Sécurité des Communications Ouvertes : Vers une Infrastructure Globale pour les Services d'Authentification

Suzan Mendes

ECOLE NATIONALE SUPERIEURE
DES MINES DE SAINT-ETIENNE

N° ORDRE: 124 ID

THESE

présentée à Saint-Etienne, le 10 Février 1995 par

SUZAN MENDES

Sécurité des Communications Ouvertes : Vers une Infrastructure Globale pour les Services d'Authentification

pour obtenir le titre de

DOCTEUR en Informatique

de l'ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE
et de l'UNIVERSITE DE SAINT-ETIENNE

COMPOSITION du JURY

Monsieur Christian HUITEMA	*président*
Monsieur Jean-Charles FABRE	*rapporteur*
Monsieur Michel HABIB	*rapporteur*
Monsieur Jean-Jacques GIRARDOT	*examinateurs*
Monsieur Alain ZAHM	
Monsieur Jean AZEMA	

ECOLE NATIONALE SUPERIEURE
DES MINES DE SAINT-ETIENNE

N° ORDRE: 124 ID

THESE

présentée à Saint-Etienne, le 10 Février 1995 par

SUZAN MENDES

Sécurité des Communications Ouvertes : Vers une Infrastructure Globale pour les Services d'Authentification

pour obtenir le titre de

DOCTEUR en Informatique

de l'ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE
et de l'UNIVERSITE DE SAINT-ETIENNE

COMPOSITION du JURY

Monsieur Christian HUITEMA	*président*
Monsieur Jean-Charles FABRE	*rapporteur*
Monsieur Michel HABIB	*rapporteur*
Monsieur Jean-Jacques GIRARDOT	*examinateurs*
Monsieur Alain ZAHM	
Monsieur Jean AZEMA	

Avant tout, je tiens à exprimer, de tout mon coeur, ma reconnaissance et ma gratitude à tous ceux qui ont collaboré et rendu possible la réalisation de ce travail, et cela dans de si bonnes conditions :

Au Créateur ; Celui qui était, qui est et qui a créé toutes choses.

A mes parents, Elmo et Eni, et à toute ma famille, Enio, Valeska, Iona, Yvonne, Ysabella et Roberto, pour leur amour et soutien émotionnel et spirituel constants, toujours. C'est à vous que je dédie ce travail.

A Michel Gien, Directeur de la Société Chorus Systèmes, mais surtout mon ami de toujours dont la rencontre karmique (si, si) m'a fait traverser l'Atlantique pour faire ce que l'on peut appeler un vrai voyage.

A Paul-Andre Pays, actuellement Directeur de la Société EdelWeb, pour avoir accepté ma candidature, m'avoir aimablement accueillie dans son équipe et ainsi avoir initié ce travail.

A Christian Huitema, Directeur de recherche a l'INRIA Sophia-Antipolis, pour avoir accepté d'encadrer et diriger ce travail, l'avoir enrichi par sa sagesse et intuition si précieuses, et finalement m'avoir fait l'honneur d'accepter la présidence de ce jury de thèse.

A Philippe Brun, actuellement Directeur de la Société GC Tech, pour m'avoir accueillie dans le Centre de Recherche & Développement de la Société E3X (actuellement Société Telis) et ainsi permettre la continuation de ce travail en milieu industriel et au bord de la mer.

A Alain Zahm, Ingénieur à la Société Telis, pour m'avoir accueillie dans son équipe, pour l'intérêt constant avec lequel il a suivi la progression de ce travail, pour sa disponibilité, ses encouragements et conseils pendant toutes ces années de thèse, et pour avoir accepté de juger ce travail.

A Séga Sako, Directeur du Département Produits et Technologies de la Société Telis, pour avoir permis que ce travail soit complété dans de si bonnes conditions.

A Jean-Jacques Girardot, Maître de Recherche à l'Ecole Nationale des Mines de Saint-Etienne, pour avoir accepté d'encadrer ce travail et pour ses encouragements pendant des moments difficiles.

A Michel Habib, Professeur au LIRM de Montpellier, pour l'intérêt porté à mes travaux et pour avoir accepté d'être rapporteur de cette thèse.

A Jean-Charles Fabre, Chargé de Recherche à l'INRIA/LAAS-CNRS de Toulouse, pour s'être intéressé à cette étude et avoir accepté d'être rapporteur de ce travail.

A Jean Azema, Maître de Conférences à l'Université Jean Monnet de Saint-Etienne pour avoir accepté de juger ce travail.

A Bernard Peroche, Directeur du Département Informatique Appliquée de l'Ecole Nationale des Mines de Saint-Etienne pour m'avoir accueillie dans son laboratoire.

Aux membres du Département Informatique Appliquée de l'Ecole Nationale des Mines de Saint-Etienne pour leur aide amicale et gentillesse, Marie-Line Barneoud, Florence, et particulièrement Elisabeth Roudier, Bernard Kaddour et Annie Bourgeat, mes anges gardiens, présents depuis le début.

Aux collaborateurs de la Société Telis Systèmes et Communication, pour leur accueil et leur aide, et spécialement aux amis Ascan Woermann, Philippe Bourdon, Ludovic Poitou et Emmanuel Kartmann pour leur affection, leur aide et soutien constants.

Aux amis Marc Dani, Véronique Duprès, Regina Del Papa, Renata Damasio, Adriana Machado, pour leur présence, leur affection et les vibrations bénéfiques qu'ils m'ont toujours transmis.

Enfin, à tous les amours de près et de loin, l'expression de ma profonde gratitude.

TABLE DES MATIERES

Page

INTRODUCTION 1

PARTIE I PRESENTATION 6

CHAPITRE 1 L'OUVERTURE ET L'INTERCONNEXION DES RESEAUX 7

1.1. Besoins de Sécurité dans les Systèmes Ouverts 9
1.2. Menaces et Attaques 9

CHAPITRE 2 SERVICES DE SECURITE 16

2.1. Authentification 17
2.2. Intégrité 17
2.3. Confidentialité 18
2.4. Non-Répudiation 18
2.5. Contrôle d'Accès 19
2.6. Analyse des Services de Sécurité 19

CHAPITRE 3 MECANISMES DE SECURITE 21

3.1. Mécanismes de Chiffrement 22
3.1.1. Crypto-systèmes à Clé Secrète 23
3.1.2. Le problème de la gestion des clés 26
3.1.3. Crypto-systèmes à Clé Publique 27
3.1.4. Utilisation des mécanismes de chiffrement 30
3.2. Mécanismes d'Authentification 32
3.2.1. Authentification à clé secrète 32
3.2.2. Authentification à clé publique 33
3.2.3. Le problème de gestion des clés publiques 34
3.2.4. Protocoles d'authentification et techniques à zero-connaissance 34
3.3. Mécanismes d'Intégrité 35
3.4. Signatures Électroniques 38

CHAPITRE 4 MOTIVATIONS ET OBJECTIFS DE CE TRAVAIL 43

PARTIE II ETAT DE L'ART 47

CHAPITRE 5 MODELES ET ARCHITECTURES DE SECURITE 48

5.1. Le Cadre d'Authentification X.509 50
5.1.1. Le Cadre de Distribution de Clés 50
5.1.1.1. Récupération et Verification de Clés Publiques 51
5.1.1.2. Gestion des Certificats 54
5.1.2. Les Protocoles d'Authentification 55

5.1.2.1. L'Authentification Simple 55
5.1.2.2. L'Authentification Forte 57
5.1.3. Lacunes du Cadre d'Authentification X.509 58
5.2. PEM **59**
5.2.1. Les Services PEM 60
5.2.2. Les Clés de Chiffrement 61
5.2.3. Les Types de Messages 61
5.2.4. La Gestion de Clés dans PEM 63
5.3. PGP **66**
5.3.1. Gestion de Clés 67
5.3.1.1. Distribution de Clés 67
5.3.1.2. Vérification de Clés Publiques 68
5.3.2. Vulnerabilités du Modèle PGP 70
5.3.2.1. Estampilles de Temps 70
5.3.2.2. Révocation de Clés Publiques 70
5.4. Le Modèle Chimæra **71**
5.4.1. Distribution des Autorités de Certification 71
5.4.2. Etablissement de Chemins de Certification 72
5.4.3. Lacunes du Modèle Chimæra 73
5.5. Les Specifications PKCS **74**
5.5.1. PKCS #1 77
5.5.2. PKCS #2 77
5.5.3. PKCS #3 77
5.5.4. PKCS #4 78
5.5.5. PKCS #5 78
5.5.6. PKCS #6 79
5.5.7. PKCS #7 79
5.5.8. PKCS #8 80
5.5.9. PKCS #9 81
5.5.10. PKCS #10 81

CHAPITRE 6 NORMALISATION DE LA SECURITE DANS OSI **82**

6.1. L'architecture de Sécurité OSI **82**
6.1.1. A propos du Placement des Services de Sécurité 84
6.1.2. Mesures Complémentaires dans les Couches Basses 85
6.2. Des Applications Normalisées **87**
6.2.1. L'Annuaire X.500 87
6.2.1.1. Aperçu General de l'Annuaire X.500 87
6.2.1.2. Le Modèle de Sécurité de l'Annuaire 88
6.2.2. La Messagerie Electronique X.400 93
6.2.2.1. Les Services de Sécurité 94
6.2.2.2. Les Mécanismes de Sécurité 96

PARTIE III EXPERIMENTATION **98**

CHAPITRE 7 LE PROJET PASSWORD **99**

7.1. L'Infrastructure de Sécurité **100**
7.1.1. Les Techniques Cryptographiques 101

7.1.2. Les Mécanismes de Base 101
7.2. Gestion de Clés et de Certificats 102
7.2.1. Le Modèle de Certification 103
7.2.1.1. Le Schéma Password 104
7.2.1.2. La Structure du Projet 106
7.3. Les Applications 108
7.3.1. L'Annuaire Réparti X.500 109
7.3.2. La Messagerie Electronique 110
7.3.2.1. Les approches 111
7.3.2.2. La messagerie X.400 111
7.3.2.3. Les Choix d'Implémentation 112
7.3.2.4. La messagerie PEM 116
7.3.3. D'autres Applications 116
7.3.3.1. ACSE 116

CHAPITRE 8 SECUCOMX : UNE ARCHITECTURE DE SECURITE POUR LES SYSTEMES OUVERTS 120

8.1. Les Services et Composants 121
8.2. La Boîte à Outils de Sécurité 121
8.3. La Gestion de Clés 122
8.3.1. La Génération de Clés 123
8.3.1.1. L'emplacement de la Génération de Clés 123
8.3.1.2. La Génération de Nombres Aléatoires 125
8.3.2. La Gestion de Clés Privées 127
8.3.2.1. Le Stockage des Clés Privées 128
8.3.2.2. L'Accès à la Clé Privée 129
8.3.3. La Gestion de Clés Publiques 133
8.3.3.1. Le Concept de CA 133
8.3.3.2. Les Services d'une CA 134
8.3.3.3. Procédures de Certification 135
8.3.3.4. La Révocation de Certificats 140
8.3.3.5. La Protection de la CA 142
8.4. L'Obtention de Données pour l'Authentification 143
8.4.1. Diversité contre Interopérabilité ? 144
8.4.2. Le DAS 146
8.4.3. Services et Modèle Fonctionnel du DAS 146
8.4.4. Clés d'Amorçage 148

CHAPITRE 9 LA SECURISATION DES APPLICATIONS 149

9.1. Un Service d'Annuaire Sécurisé 149
9.1.1. Architecture et Services 150
9.1.1.1. L'agent utilisateur d'annuaire 151
9.1.1.2. Le serveur d'annuaire 151
9.1.2. Authentification durant l'établissement de l'association 152
9.1.2.1. L'authentification simple protégée 152
9.1.2.2. L'authentification forte 153
9.1.3. Authentification des Opérations 153
9.2. Un Service de Messagerie Sécurisé 154

9.2.1. Architecture et Services 155
9.2.2. Fonctionnement de l'Impléméntation 156

CHAPITRE 10 DES PROBLEMES OUVERTS 158

10.1. Le Cadre X.509 comme base de l'Infrastructure de Sécurité 159
10.1.1. A propos du Schéma de Nommage 159
10.1.1.1. Les Noms DNS 160
10.1.1.2. Les Noms d'Annuaire 160
10.1.2. A propos du Format de Certificats 162
10.1.2.1. Un format de certificat pour divers services sécurisés 162
10.1.2.2. Un format de certificat pour divers types d'attributs 162
10.1.2.3. Un format de certificat pour divers types de CA 163
10.1.2.4. Un format de certificat pour divers types d'utilisateur 163
10.1.3. Le Format de Listes de Révocation 164
10.1.4. La Dépendance vis-à-vis de l'Annuaire X.500 165
10.2. La Sécurité de l'Annuaire X.500 167
10.2.1. Protection des Informations de Connaissance 167
10.2.2. La Confidentialité et la Protection de la DIB 167
10.3. La Sécurité de la Messagerie X.400 169
10.3.1. Des Lacunes du Modèle de Sécurité de X.400 170
10.3.2. Un Service Complémentaire à la Classe S0a 171
10.3.3. Le Service de Preuve de Remise 173
10.3.4. X.400 versus PEM 175
10.4. D'autres Applications : L'ACSE et le restant de l'association 177

PARTIE IV EVOLUTION DU CONCEPT DU DAS 180

CHAPITRE 11 INTRODUCTION 183

11.1. Motivations 183
11.1.1. X.509 et les Modèles de Confiance 183
11.1.2. Manque de Precision du Cadre X.509 184
11.1.3. Diversité, Sécurité ou Interopérabilité ? 188
11.2. Une Nouvelle Approche 189

CHAPITRE 12 REORGANISATION DU SYSTEME 191

CHAPITRE 13 DESCRIPTION DES POLITIQUES DE CERTIFICATION 194

13.2. Objectifs 195
13.3. Une Méthode de Description de Politiques de Certification 196
13.3.1. Eléments du Modèle de Description 196
13.3.2. Choix du Langage 199
13.3.3. Méthode de Description 199
13.3.4. Emplacement 204
13.4. Etude de cas et Exemples 205
13.4.1. Modèles Publics 206
13.4.2. Modèles Privés 210
13.4.3. Interopérabilité entre Modèles Hétérogènes 212
13.5. Considérations 213

CHAPITRE 14 VERIFICATION DE CLES PUBLIQUES 214

14.1. L'Evaluation de Chemins de Certification 215
14.2. Récupération de Listes de Révocation 217
14.3. Utilisation des Description de Politiques de Certification 222
14.4. Critères Locaux d'un Domaine 223
14.5. Métriques d'Evaluation 227

CHAPITRE 15 LE SERVEUR DE DONNEES POUR L'AUTHENTIFICATION 231

15.1. Rôle du DAS dans l'Architecture de Gestion de Clés 232
15.1.1. Emplacement du DAS 232
15.1.2. Analyse des Services du DAS 232
15.1.3. Analyse du Fonctionnement du DAS 235
15.2. Modèle de Fonctionnement du DAS 235
15.2.1. Environnement Distribué des DAS 236
15.2.2. Opération Distribuée des DAS 237
15.2.3. Navigation dans l'Espace de Clés Publiques 238
15.2.3.1. La Connaissance du DAS 239
15.2.3.2. La Procédure de Navigation des DAS 241
15.3. Utilisation de Caches 245
15.3.1. Les Caches du Service de Noms 245
15.3.2. Les Caches du DAS 246
15.3.2.1. Principes 246
15.3.2.2. Besoins 246
15.3.2.3. Gestion des Caches du DAS 249
15.3.2.4. Structure des Caches du DAS 251
15.4. Les Protocoles 253
15.4.1. Accès au DAS 253
15.4.2. Interaction entre DAS 259
15.5. Assurance de la Protection du Service 260
15.5.1. Usurpation de l'identité du DAS ou du Service de Noms 260
15.5.2. Protection de la Connaissance du DAS 261
15.5.3. Consistance et gestion de connaissance 261
15.5.3.1. Consistance et gestion de la connaissance interne et externe minimale 262
15.5.3.2. Consistance et gestion de la connaissance externe ajoutée 264

PARTIE V CONCLUSION 265

ANNEXE A
Le Cadre d'Authentification X.509 en ASN.1 270

ANNEXE B
Architecture SecUcomx en ASN.1 274

ANNEXE C
Définitions ASN.1 pour l'Infrastructure Global pour les Services d'Authentication 280

BIBLIOGRAPHIE 289

LISTE DES ILLUSTRATIONS 306

GLOSSAIRE 308

INTRODUCTION

"Les réseaux : la dernière utopie pirate de l'ère de l'information", publiaient les journalistes de Mondo 2000, le magazine *cyber*[1] du moment [Wiz94].

Une technologie dominante a marqué chacun des trois derniers siècles. Le dix-huitième siècle a été l'ère des grands systèmes mécaniques propres à la Révolution industrielle. Le dix-neuvième siècle a été l'âge de la machine à vapeur. Durant le vingtième siècle, la technologie clé a concerné la collecte, le traitement et la distribution de l'information. Nous avons vu, entre autres, la mise en place d'un réseau téléphonique planétaire, l'invention de la radio et de la télévision, la naissance et la croissance sans précédant de l'industrie informatique et le lancement de satellites de communication [Tan89].

Au cours des dernières années de ce siècle, ces domaines convergent et les différences entre collecte, traitement, transport et stockage de l'information s'estompent rapidement. La fusion des ordinateurs et des télécommunications a eu une profonde influence sur la structuration des systèmes. Le modèle ancien de l'ordinateur unique desservant tous les besoins de traitement d'une organisation s'est trouvé rapidement supplanté par celui de l'exécution du travail par des ordinateurs séparés mais interconnectés. De tels systèmes sont appelés *réseaux d'ordinateurs*.

---

[1]De *cybernétique* : la science du contrôle, du feed-back qui décrit les rapports entre l'homme et la machine. Mot créé par Norbert Wienner, chercheur du MIT, designer de canons antiaériens pendant la seconde guerre mondiale. Du grec kybernetes : le pilote sur un navire.

La cybernétique a influencé la psychologie (école de Palo Alto), l'intelligence artificielle et la vie politique (les sondages). Elle gouverne la robotique, l'installation des réseaux , la guerre [Act93]. Nous espérons qu'elle n'amènera pas la guerre numérique.

Mais le concept lui-même de réseau a évolué. Le domaine des réseaux d'ordinateurs a considérablement changé pendant les deux dernières décennies. Il y a vingt ans, les réseaux étaient des outils de recherche étranges à l'usage de quelques spécialistes. Les réseaux sont passés du stade de la curiosité de laboratoire à celui d'outils indispensables pour le monde académique, des affaires et les administrations.

Si dans le passé on avait des réseaux fermés, sans problèmes de sécurité, à l'heure actuelle, les organisations ont dispersé leurs sites et ont "éparpillé" leur informatique. Les réseaux de communications se sont donc étendus. Un autre facteur de croissance des besoins en sécurité est lié à la conception même des réseaux. Ils ont été créés avec une volonté d'ouverture et les standards comme UNIX et TCP/IP sont par définition des systèmes ouverts à tous. Aujourd'hui l'informatique n'est plus restreinte à ses professionnels et experts. Des enterprises offrent pour un prix assez modique, un abonnement qui permet l'accès à l'Internet. Nous sommes à l'âge de la surinformation et de la banalisation de l'informatique.

La mise en réseau planétaire est entrée dans une phase exponentielle, et la société accompagne cela en entrant dans une phase de mutation culturelle. L'explosion des capacités, des communications d'une part et, d'autre part des techniques de synthèse d'images et de multimédia a augmenté les rapports entre homme et machine, provoquant ainsi l'émergence de nouveaux concepts tels que la "réalité virtuelle" et d'une nouvelle forme de culture : la *cyber-culture*. Celle-ci amène avec elle l'exploration des univers virtuels à travers les réseaux de communication. La transition vers le XXI^{ème} siècle est donc placée sous le signe de la cyber-culture, ascendant systèmes ouverts.

Dans l'opinion du grand public, l'INTERNET signifie le réseau de tous les réseaux. Minitel mondial et tentaculaire où se connecteraient au même instant le vieux bibliothécaire gardien d'incunables d'Instambul, le boutonneux Heavy Metal de l'Arkansas, le col blanc de la Silicon Valley, le designer japonais à la recherche du satori [Wiz94].

Pour les adeptes de la cyber-culture, aller sur l'Internet, autoroute "digitale" qui bientôt traversera le monde entier, c'est prendre la voie royale de la société de l'information. L'idée leur est commune qu'il existe un monde nouveau au bout des doigts, un monde où l'on parle directement sans se connaître, où une présence sur un écran suffit à faire de chacun le membre d'une communauté élective, où toutes les informations possibles sont disponibles, où l'on peut partir d'un PC branché n'importe où pour rentrer dans les ordinateurs des gouvermenent, des banques, des centres de recherche ainsi que dans des banques de données diverses. Tout un monde, une planète de science-fiction, de milliards de kilomètres de textes et d'images accessible là, au bout de trois opérations informatiques.

Si la banalisation et la plus grande dispersion du savoir-faire informatique génèrent de nombreux avantages en termes notamment d'amélioration générale des connaissances et de diminuition des temps de formation, elles ont aussi entraîné une multiplication des risques d'intrusion et de fraude. Le tout est de récolter le plus de codes secrets possibles, pour pouvoir se promener dans les coins les plus exclusifs du "*Net*", ceux où se trament les grands débats scientifiques, politiques, moraux ou zarbis du moment [Wiz94].

Ceux qui auparavant étaient connus comme pirates informatiques, ont désormais gagné d'autres dénominations pour apparaître comme personnages de la contre-culture électronique. Ils sont les *cyberpunks*, les rebelles de la cyber-culture et s'appellent le *hacker* et le *cracker*.

Considéré auparavant comme un jeu de jeunes, dans la cyber-culture, le piratage devient plus directement politique. Les cyberpunks considèrent que l'information c'est le pouvoir. Dans l'esprit des hackers, il s'agit de lutter pour la démocratie et contre le Big Brother électronique (représenté par l'Etat et les transnationales), de récupérer de l'information sensible et la jeter à tout vent. Les crackers, les plus radicaux des cyberpunks, sont d'une nouvelle génération. Leur intention n'est pas la démocratie, mais plutôt le vandalisme pour en tirer profit. Ils veulent rentrer dans les systèmes, si possible par effraction, pour provoquer le chaos informatique, semer des virus, récupérer de l'information revendable, voire vider des comptes bancaires.

A l'heure actuelle, beaucoup d'énergie est dépensée pour parler de d'Internet et de son accès. Débat auquel s'ajoute celui de la liberté d'expression sur le réseau. L'Etat veut néanmoins garder le contrôle sur tout ce qui pourrait porter atteinte à la sécurité nationale. En effet, même si les procédés de chiffrement ne sont plus classés comme armes de guerre, la cryptologie reste un domaine sensible.

Du coté des utilisateurs, la menace d'introduction de dispositifs (du type Clipper [NIST93, Hel93] par exemple) permettant à des agents gouvernementaux divers de décoder et de surveiller des conversations électroniques codées sur le réseau, est considérée comme une atteinte majeure aux libertés individuelles, selon les *cyberpunks*, nouvelle génération des pirates informatiques, et les *cypherpunks*, adeptes du codage personnel.

Toutefois, la communication de données à distance n'est pas un fait vraiment nouveau. Voila déjà plusieurs années depuis le télégraphe optique de Georges Chappe, que les informaticiens ont pris l'habitude d'échanger des messages par l'intermédiaire de leur outil de travail, l'ordinateur [AM91].

Ce moyen de communication est resté marginal jusqu'à ce que les sociétés d'informatique et les organismes de recherche, poussés par le besoin de partage d'information à grande échelle, se lancent dans l'aventure informatique à la fin des années 1970. La surface terrestre s'est alors couverte progressivement d'un tissu de liens de télécommunication, effaçant les contraintes géographiques en reliant, plus ou moins efficacement, des centaines d'ordinateurs à travers tous les continents.

Les services offerts par ces réseaux téléinformatiques permettent à des groupes d'abonnés d'horizons divers de bénéficier des avantages de la communication électronique : rapidité et économie de papier, non nécessité de la disponibilité immédiate du correspondant.

Aujourd'hui nous disposons de nombreuses applications réseau. Citons par exemple le transfert de fichier, le "remote login" ou connexion à distance, l'impression déportée, l'annuaire et la messagerie électroniques avec des capacités multimédia, la video-conference, l'EDI (Electronic Data Interchange), l'EDT (Electronic Data Transfer), etc. Il est possible d'envoyer un message au Ministre de la Culture français ou au Président des Etats Unis, de consulter des bases de données lointaines et même de visiter le musée du Louvre à partir de n'importe quel endroit du monde.

Nous (et aussi ces cyberpunks) savons bien que le soi-disant "net" n'est que l'extension actuelle d'un réseau commencé il y a déjà vingt ans par les forces armées américaines sous le nom d'ARPAnet. Lorsqu'un ordinateur s'y connecte, il devient automatiquement vulnérable à la surveillance et aux diverses attaques : piratage de ligne, abus de confiance, fraudes et violations d'intimité. Ce qui peut devenir une vaste autoroute "digitale" de l'âge de la surinformation, la base de la "digitalisation" de l'économie, n'est jamais clos hermétiquement.

Les fraudes et catastrophes informatiques ne relèvent pas de romans de science-fiction et les accidents ou les actes de malveillance coûtent des centaines de millions de francs (6270 MF de perte en 1993 par malveillance, constituant 58% des pertes informatiques) [TS94]. Les exemples ne manquent pas. On peut citer la fraude sur les taux des devises d'une grande banque d'affaires qui provoqua une perte de 3300 millions de francs.

Nous avons évoqué la figure des *cyberpunks* seulement comme exemple (assez simpliste) pour illustrer un problème majeur, à savoir d'une part le rapport entre l'augmentation des enjeux supportés par les systèmes informatiques et la transformation de la société, et d'autre part la manque de sécurité des réseaux. Toutefois, l'errance des *crackers* et des *hackers* pose déjà des inconvénients. Par exemple, les 6000 ordinateurs contaminés sur l'Internet en 1988 et les 15000 mots de passe forcés en une nuit sur l'Internet vers le mois de février 1994.

Si des fonctionnalités de sécurité existent dans les systèmes locaux, de façon isolée, celles-ci ne garantissent plus rien dès lors qu'il s'agit d'informatique communicante et que l'ennemi vient de l'extérieur, par exemple informatique distribuée, réseaux, client-serveur, télémaintenance, connexion au monde TCP/IP-Internet, etc. Des mécanismes de filtres logiciels, voire physiques, doivent alors être mis en place. Le problème s'accentue avec l'émergence des applications réseaux qui peuvent être reparties sur plusieurs sites [Val93].

Avec l'interconnexion des ordinateurs en réseau et l'apparition de l'informatique hétérogène et distribuée, de telles protections sont cruciales. Les besoins en sécurité vont donc aller croissant. Il faudra de plus en plus s'assurer que des données confidentielles ne puissent pas être à la portée de quelconques pirates informatiques. Il suffit de penser aux problèmes qui peuvent se poser dans le cas des banques par exemple.

Un certain nombre d'études et notamment celles menées dans le cadre de l'Institut des Hautes Etudes de la Sécurité Interieure (IHESI), montrent que les menaces pesant sur les systèmes d'information augmenteront fortement d'ici à l'an 2000 et que cette progression sera amplifiée du fait des tensions nationales et interntionales [Atz94]. Cet accroissement serait entre autres dû à la complexification et l'interconnexion des systèmes, l'évolution des mentalités, la banalisation et la diversification de l'informatique, l'explosion des communications et l'augmentation des enjeux supportés par les systèmes d'information.

Cela signifie que la mise en réseau de l'économie mondiale doit être acompagnée des mécanismes de protection de la vie privée et collective à travers les réseaux. Notre travail porte donc sur la sécurité des systèmes ouverts. Celui-ci a plus specifiquement pour but de proposer des solutions aux problèmes actuels concernant la sécurité des applications ouvertes.

Les premières études dans le contexte de la sécurité des systèmes ouverts nous ont amené à réfléchir sur le problème complexe et vaste de l'établissement d'un canal de communication sécurisé, dans un environnement ouvert et hétérogène, entre deux entités quelconques sans besoin d'échange préalable. Ce problème de communication sécurisée entre applications et usagers quelconques a constitué le contexte de l'étude présentée. Ce problème a comme origine l'authentification et la gestion des clés cryptographiques.

Nous proposons donc dans cette étude une nouvelle approche pour la structuration des architectures de sécurité et une méthode originale pour la description de politiques de certification de clés publiques ayant avec pour objet de rendre viable l'établissement d'une infrastructure globale de commune pour les services d'authentification dans un environnement ouvert et hétérogène. S'appuyant sur le cadre d'authentification X.509 défini par le CCITT pour les applications ouvertes, l'implémentation d'une telle approche permettrait la construction d'architectures de sécurité homogènes du point de vue de l'utilisateur final.

Ce travail est organisé en cinq parties.

La première partie présente les deux axes généraux d'étude : les systèmes ouverts et la sécurité. Dans un premier chapitre nous aborderons rapidement les aspects d'ouverture, d'expansion et d'interconnexion des réseaux informatiques, ses applications ainsi que les besoins de sécurité qu'ils engendrent. Le deuxième chapitre aborde rapidement les services définis pour satisfaire aux besoins en sécurité des systèmes. Le troisième chapitre présente un bref aperçu des techniques de sécurité ainsi que des développements et références dans le domaine de la cryptologie. Le quatrième chapitre finalise cette partie en exposant, en plus de détail, le cadre, les motivations, les objectifs et l'organisation de la suite de notre travail.

La deuxième partie a pour objet de donner un aperçu de l'état de l'art actuel de la sécurité dans le contexte des systèmes ouverts, en abordant des modèles, architectures et recommandations proposés à ce propos.

La troisième et quatrième parties de ce travail constituent les points clé de notre étude. Cette première présente notre expérimentation dans le domaine traité. Celle-ci s'est concrétisé à travers notre participation dans le projet Password : un projet pilot européen ayant pour but d'élaborer une infrastructure de sécurité basée sur l'approche X.509 et de piloter le déploiement de services sécurisés pour les principales applications ouvertes. Ce travail a abouti à la conception d'une architecture de sécurité, appelée SecUcomx, permettant la sécurisation des échanges en milieu ouvert, ainsi qu'à l'identification de nombreux aspects négligés par l'approche X.509, qui a été utilisée comme cadre de base. La quatrième développe donc une nouvelle approche et de nouveaux concepts, allant au-delà du cadre X.509, ayant pour objectif de permettre la mise en place d'une infrastructure globale pour les services d'authentification. Ceci en respectant l'hétérogénéité du système global, les politiques de sécurité des diverses communautés et la conformité avec les standards actuels.

La cinquième partie termine donc notre étude en montrant l'intérêt, la validité et les perspectives de l'approche proposée dans le cadre de la sécurité des systèmes ouverts.

PARTIE I

PRESENTATION

---

1 . L'OUVERTURE ET L'INTERCONNEXION DES RESEAUX

2 . SERVICES DE SECURITE

3 . MECANISMES DE SECURITE

4 . MOTIVATIONS ET OBJECTIFS DE CE TRAVAIL

CHAPITRE 1
L'OUVERTURE ET L'INTERCONNEXION DES RESEAUX

Des réseaux isolés sont, de nos jours, en train de s'intégrer pour former un réseau universel, constitué d'un nombre de réseaux interconnectés. A l'heure actuelle, de nombreux organismes publics et privés sont connectés par la structure Internet. Ce réseau est actuellement basé sur les protocoles TCP/IP, initialement développés pour le réseau ARPAnet, l'un des plus anciens réseaux informatiques crée en 1969 par l'Advanced Research Projects Agency (ARPA) du Département de la Défense Américaine (DoD).

A vocation militaire au départ, ce réseau s'est ouvert aux organismes publics américains, tels que l'éducation, les différents organismes, ainsi qu'aux industriels. Maintenant la famille de protocoles TCP/IP évolue de manière autonome dans le cadre de l'IETF (Internet Engineering Task Force).

Les sites sont reliés entre eux par des lignes téléphoniques commutées, par des lignes spécialisées. Une partie du trafic utilise aussi des protocoles définis par l'ISO (X.25 [IS8208] et CLNP [IS8473]). Des mécanismes d'encapsulation du protocole TCP/IP sont d'un usage fréquent pour le transport sur d'autres réseaux.

Le service de base de l'Internet est la commutation de paquets IP. Si un organisme est connecté au réseau alors on peut toujours lui envoyer des paquets. Dans un réseau mondial tel que l'Internet, les tâches de routage adaptatif, d'équilibrage de charge et de reroutage automatique doivent être effectués à échelle mondiale. Les paquets échangés peuvent donc suivre des chemins variés. Ainsi, il est impossible de faire confiance à tous les réseaux intermédiaires traversés par ces paquets. Par conséquent, il est en pratique impossible de garantir la sécurité du système global même si les opérateurs de diverses réseaux qui le constituent appliquent les mesures adéquates pour rendre leur réseau relativement sûr.

Outre l'Internet, des ordinateurs et des stations de travail sont connectés par des réseaux locaux (LAN) englobant toute une organisation ou des parties de celle-ci. Ces réseaux locaux peuvent être connectés entre eux à travers des réseaux fédérateurs (backbones) et des passerelles.

Bien que les réseaux locaux présentent des avantages tels qu'un débit élevé, un délai court de transmission et un trafic indépendant du coût de transmission, ces propriétés posent également des menaces considérables au niveau de la sécurité.

Une station dans un réseau local peut écouter tout le trafic en transit sur le media, sans que cela ne soit détecté. De plus, une station peut se servir d'une fausse identité pour se poser comme une autre station. Avec de telles fonctions, l'une des possibilités d'attaque est l'enregistrement et le re-jeu de séquences d'authentification. Il suffirait de surveiller le trafic du réseaux en question et rechercher des séquences du type "*username*" et "*password*", enregistrer les réponses pour utiliser ultérieurement ces mots de passe pour obtenir des services sous une fausse identité.

Une autre fonctionnalité dangereuse des LAN permet à une station d'émettre des paquets avec une adresse source quelconque. Cela est notamment utilisé pour réaliser des ponts inter-réseaux. Un utilisateur malveillant peut alors s'infiltrer dans le réseau en se faisant passer pour un tiers. Cela ouvre les portes à toute sorte d'attaques. Le groupe de travail IEEE 802.10 a travaillé sur la normalisation de mécanismes de sécurité au niveau de la couche liason des données des LAN.

Outre les LAN, les réseaux de zone métropolitaine (MAN) sont en émergence. Ces réseaux engloberont des villes entières. Des réseaux locaux et ceux basés sur des commutateurs numeriques privés PBX (Private Branch eXchange) pourront se connecter à ces MAN.

Les WAN (Wide Area Networks) sont utilisés pour des services à longue distance. Les plus modernes sont basés sur des systèmes de transmission à fibre optique avec des débits d'une ordre de Gigabytes par seconde. [Par90] prévoit une augmentation des capacités d'interconnexion entre réseaux dans les prochaines années.

Les organismes internationaux de normalisation et en particulier le CCITT, conscients de la nécessité d'une architecture universelle de communication, ont donc élaboré des normes et recommandations sur l'interconnexion des systèmes informatiques (normes OSI/ISO), et à terme, toutes les machines communiqueront selon des règles communes.

En particulier, le protocole CLNP (**C**onnection**L**ess **N**etwork **P**rocotol), aussi connu sous le nom de "ISO IP" est considéré comme le successeur naturel du protocole IP, offrant pratiquement la même fonctionnalité mais de façon normalisée au niveau mondial et avec un espace d'adressage plus large et mieux structuré.

Dans un futur proche, les réseaux corporatifs, les réseaux à valeur ajoutée (VAN) et d'autres réseaux isolés ne constitueront pas des réseaux physiquement séparés mais plutôt des réseaux virtuels, i.e., des collections de points d'accès de service réseau (NSAP) formant un seul réseau logique. Il existe, en outre, des technologies émergentes de Réseau Numérique à Intégration de Services ou RNIS (ISDN - **I**ntegrated **S**ervices **D**igital **N**etwork), dont l'objectif principal est l'intégration dans un même réseau de services relatifs aux applications vocales et à d'autres formes de services de communication.

Ces évolutions ouvrent de nouvelles perpectives tant pour les utilisateurs que pour les développeurs des systèmes d'information. Par exemple, une station de travail connectée à un réseau local peut communiquer avec toute autre machine à échelle mondiale avec une haute bande passante et avec des délais réduits. Il est clair qu'un tel potentiel de connectivité amène des risques de sécurité auxquel on n'a pas pensé jusqu'à présent et qui doivent être mis sous contrôle.

L'émergence de ces nouvelles architectures de réseau présentent des challenges pour les opérateurs et les développeurs des systèmes :

- l'administration des réseaux nécessite d'être visée sur un nouveau niveau, où les réseaux virtuels peuvent être définis et gérés efficacement, afin de maîtriser l'intégration logique et variable des réseaux séparés.
- il est nécessaire de disposer d'outils et des moyens pour la mise en vigueur de politiques de sécurité déterminées par les propriétaires et par les utilisateurs de ces réseaux virtuels.

La question qui se pose est : comment se protéger des menaces inhérentes à l'interconnexion de systèmes ? Bien entendu, nous ne devons pas abandonner les possibilités offertes par l'émergence de ces nouvelles technologies. Karila dans sa thèse [kar91] nous invite à rejeter l'hypothèse des structures réseaux complètement sûres, et à développer avec la technologie disponible aujourd'hui, des systèmes dont la sécurité ne dépend pas de la sécurité des réseaux sous-jacents. A partir de la technologie cryptographique disponible aujourd'hui, ce qui est nécessaire est de travailler sur des aspects architecturaux afin de définir et de bâtir des systèmes ouverts et sécurisés.

1.1. Besoins de Sécurité dans les Systèmes Ouverts

Considérés par certains comme concepts contradictoires, connectivité et sécurité sont en fait concepts indépendants. Bien que plus on ouvre un système, plus on augmente les risques d'intrusion, l'ouverture des systèmes ne doit pas pour autant impliquer un manque de sécurité mais l'augmentation du potentiel d'interconnexion et d'interopérabilité entre systèmes provenant de divers fabricants, gérés différemment, de niveaux de complexité différents, et d'âge différent. En contrepartie, l'introduction de mesures de sécurité ne doit pas impliquer des limitations dans l'ouverture des systèmes, mais une augmentation de la qualité des services rendus.

De plus, l'ouverture des systèmes rend nécessaire la capacité d'établissement d'engagements légaux par la voie électronique. Par exemple, l'un des obstacles à la prolifération de l'EDI (Electronic Data Interchange) et d'autres applicatifs inter-corporatifs est le manque de sécurité de ces mécanismes et des systèmes distribués de nos jours.

Ainsi, outre les protocoles de communication, il faut normaliser les procédés de sécurité pour ouvrir les systèmes. Récemment, le besoin de sécurité des systèmes ouverts a été officiellement reconnu dans divers forums. Du côté des grands organismes de normalisation, comme l'AFNOR ou l'ISO, qui prônent l'ouverture des systèmes, on s'intéresse aussi aux problèmes de sécurité. Des travaux récents des organismes internationaux de normalisation (ISO et CCITT) se sont concentrés sur les divers aspects de l'interconnexion sécurisée des systèmes ouverts. Nous présenterons un aperçu des travaux récents dans ce domaine dans la Partie II.

1.2. Menaces et Attaques

Il existe de nombreuses tendances que l'adoption largement répandue de protocoles de communication normalisés intensifieront. Ces tendances accroissent les besoins en sécurité des réseaux et des systèmes :

- l'augmentation d'utilisation des réseaux pour offrir l'accès à des services distants couplée aux mesures locales de sécurité rend l'attaque à ces réseaux plus attirant aux intrus et pirates.
- la croissance de valeur et de quantité des informations conservées sous forme électronique couplée au manque actuel de sécurité des réseaux rend ceux-ci des cibles d'attaque intéressantes.
- l'interconnexion des systèmes d'information en réseau implique souvent la coopération entre ces systèmes pour permettre le partage de ressources. Comme résultat de ce partage, la sécurité de l'information dans un système peut être, dans une certaine mesure, dépendante des mesures de sécurité employées dans le réseau et dans les systèmes le constituant.
- le développement de nouvelles technologies facilite certains types d'attaque sur les systèmes de communication ; par exemple, il n'est pas très complexe pour un intrus de surveiller des transmissions par satellite ou radiophoniques.

Les attaques aux systèmes d'information sont souvent classifiées de façon générale entre les *attaques passives*, où l'intrus ne fait qu'observer des données échangées, et les *attaques actives*, où l'intrus modifie (ou affecte) en quelque sorte la nature des données. Ainsi, les attaques passives sont difficiles, sinon impossible à être détectées tandis que les attaques actives le peuvent.

Toutefois, il est important de noter que les réseaux de communication ouvrent deux types de menaces fondamentalement différentes aux ressources et aux informations. D'une part, les réseaux sont un moyen d'accès aux systèmes locaux, d'autre part les réseaux constituent eux-mêmes le lieu des communications.

Ainsi, non seulement le danger de manipulation des systèmes locaux et des informations conservées augmente considérablement avec l'interconnexion des systèmes, mais aussi les ressources des systèmes de communication et l'information transmise par les réseaux comme acte de communication sont des cibles de manipulation abusive. Par conséquent, nous pouvons considérer comme cibles sujettes à des attaques potentielles :

- les ressources et informations locales à travers le réseau,
- les réseaux eux-mêmes.

Dans la technologie classique de sécurité, seules les menaces intentionnelles étaient considérées, à savoir l'espionnage et le sabotage. L'espionnage comprend toutes les attaques passives telle que l'observation non autorisée d'informations. Le sabotage comprend toutes les attaques actives, i.e., de tous les types de manipulation non autorisée des données, des ressources et des systèmes, jusqu'au vol des supports materiels et logiciels. Non seulement les utilisateurs humains sont affectés par le manque de sécurité de même que les composants des systèmes de communication.

Aujourd'hui, les menaces accidentelles sont également considérées comme une atteinte à la sécurité des systèmes. Par exemple, la mauvaise maintenance d'un mécanisme de routage peut amener à une interruption des services dans le réseau. Du point de vue de l'utilisateur final, il est indifférent que le problème ait été provoqué par des défauts d'administration ou par une attaque intentionnelle. Du point de vue de l'administrateur, cela n'est pas indifférent. Celui-ci doit sélectionner des mesures adéquates contre les deux sources de problème.

Dans notre étude nous préférons classifier les attaques en rapport avec la sécurité des communications. Nous pouvons donc identifier deux types différents d'attaques :

(1) un type d'attaque est celui contre les réseaux et les informations qui y transitent.

(2) l'autre type d'attaque est celui qui utilise les réseaux comme un moyen d'accéder aux systèmes locaux.

Parmi les attaques du type (1) ci-dessus nous pouvons citer :

- **Interception d'identité**

  Il s'agit d'intercepter l'identité d'un ou de plusieurs des utilisateurs intervenant dans une communication afin de se faire passer pour l'un de ces utilisateurs.

- **Re-jeu**

  Il s'agit de l'enregistrement et de la répétition ultérieure d'une communication.

- **Interception de données**

  Il s'agit de l'observation de données au cours d'une communication par un utilisateur non autorisé.

- **Manipulation de données**

  Il s'agit de tout type de modification non autorisée des données en violant leur intégrité, i.e., remplacement, insertion, suppression ou mise en désordre de données au cours d'une communication par un utilisateur non autorisé.

- **Rejet ou Répudiation**

  Il s'agit de la dénégation d'avoir participé à une communication.

- **Refus de service**

  Il s'agit de l'empêchement ou de l'interruption d'une communication ou encore retard d'opérations urgentes ou limitées dans le temps.

- **Acheminement erroné**

  Il s'agit de l'acheminement erroné d'un itinéraire de communication prévu vers un autre.

- **Analyse de trafic**

  Il s'agit de l'observation de l'information relative à une communication entre deux entités (par exemple, absence, présence, fréquence, sens, séquence, type, volume, etc.).

  Même si un intrus n'est pas capable de lire des messages, il peut déduire des informations utiles (pour lui) en observant les adresses source et destination des messages, leur taille, la date d'emission, etc, telle que l'observation malicieuse d'une facture téléphonique détaillée.

A titre d'exemple, les noeuds fonctionnant comme passerelle entre deux réseaux, peuvent intercepter le trafic et lancer tout type d'attaque active contre l'intégrité des communications. Concrètement, les attaques ci-dessus peuvent servir à, par exemple :

- violer la confidentialité des informations,
- accéder illégalement à des informations,

- acquérir des privilèges supplémentaires,
- générer et/ou expédier de l'information frauduleusement,
- autoriser ou endosser des transactions frauduleusement,
- impliquer d'autres frauduleusement, par exemple en modifiant des informations liées à ces parties,
- désavouer ou renier la responsabilité liée à des transactions effectuées ou à des informations expédiées ou reçues,
- prétendre avoir reçu d'un autre utilisateur des informations générées et/ou expédiées par un imposteur, i. e., l'attribution frauduleuse de responsabilité,
- prétendre avoir envoyé à un destinataire (à une date et heure spécifiques) des informations qui n'ont pas été envoyées (ou qui ont été envoyées à une date et/ou heure différentes),
- empêcher la communication entre certaines parties.

Considérons maintenant le cas où les réseaux sont utilisés comme un moyen d'attaquer des systèmes locaux. La connexion à distance non autorisée et la propagation de virus en sont les versions les plus courantes de ces types d'attaque.

- **Pirates informatiques**

  Ceux-ci ont, de nos jours, gagné des noms spéciaux. Ils s'appellent *hackers* (les pirates) lorsqu'ils ne réalisent pas de modifications sur les systèmes auxquels ils accèdent. Sinon, ils sont appelés *crackers* (les casseurs).

  Bien que parfois considéré comme un jeu inoffensif des jeunes, le piratage (ou *hacking*) est en fait bien plus important que cela : il constitue une méthode pour tester la qualité des mécanismes de contrôle d'accès d'un site ou d'une machine.

- **Programmes malicieux**

  De même que les chevaux de Troie, d'autres instances des dits *programmes malicieux* ou *logiques malicieuses* existent [Den88]. Les programmes malicieux utilisent les droits d'un utilisateur légitime pour exécuter ses fonctions et donc exploitent non seulement les défaillances des systèmes mais aussi le fait que ni les systèmes d'exploitation ni les mécanismes de protection ne peuvent pas distinguer les actions intentionnelles des involontaires. Normalement, toutes les classes de programmes malicieux exploitent les mêmes vulnérabilités dans leurs attaques.

  Un **worm**[2] est un programme capable d'être exécuté indépendamment et d'en propager une version exécutable vers d'autres machines d'un réseau. Lorsqu'il réussit à s'introduire dans une machine, le worm s'exécute localement et puis se transfère à travers un port de connexion vers une autre machine cible, normalement sans laisser de traces. Plus de détails sur les origines du terme peuvent être trouvés dans [Spa91].

---

[2]En français : ver.

Différent des virus, les worms sont considérés comme des programmes indépendants et n'attaquent pas les structures matérielles et logicielles des machines. Ils suivent seulement leur "route".

Bien qu'un worm ne soit pas typiquement considéré comme un programme vraiment malin, car il ne détruit pas les structures des systèmes, un tel programme peut tout à fait être utilisé pour introduire des virus ou d'autres programmes malicieux dans les machines pénétrées.

Deux worms sont devenus célèbres dans les dernières années. L'un a été connu sous le nom de "*Christmas tree*" et est apparu sur l'Earn-Bitnet en 1987. L'autre est le "*Morris worm*" apparu en 1988 sur l'Internet. Apparemment, leur intention a été un mélange d'envie d'amusement et de montrer le manque de sécurité des systèmes. Leur effet a été un important refus de service car ils ont surchargé les machines et les lignes de communication affectées.

Le "Christmas tree" a réussi car de nombreuses personnes ont aveuglément fait confiance à des messages anonymes qui, pendant l'époque de Noël, leur a demandé de les lancer comme un programme. Ce programme apparemment inoffensif, avait pour objet et pour effet indirect d'envoyer des ces mêmes messages à toutes les adresses trouvées dans les calepins locaux.

Le "Morris worm" a été exécuté dans certaines machines et s'est propagé rapidement, en collectant dans chaque machine pénétrée des informations sur des utilisateurs (nom de login et mots de passe), d'autres machines (adresses) afin de s'introduire dans d'autres machines.

Ce programme n'impliquait pas les utilisateurs directement. Il a exploité des faiblesses présentes dans des logiciels standard des systèmes UNIX version BSD 4, tels que *sendmail*, *fingerd* et *gets*. Une autre vulnérabilité exploitée est le fait que le fichier abritant les mots de passe chiffrés des utilisateurs ait des droits publics de lecture. Le programme a donc pu réussir des attaques par dictionnaire[3] sur certains mots de passe et les utiliser pour pénétrer d'autres machines. [Spa88] fournit une étude détaillée de ces faiblesses, ainsi qu'une description sur la façon dont ce worm a utilisé ces faiblesses pour envahir les systèmes.

Ce worm se propageait à travers l'Internet et reportait toute pénétration réussie, afin de donner une preuve de succès à son auteur. Cependant, l'auteur a oublié de donner une direction à son programme. Une fois le système pénétré, le programme re-entrait un nombre croissant de fois, paralysant rapidement le système. En quelques heures, ce programme a rendu inutilisable plusieurs milliers de machines. [Spa91] présente une chronologie de la diffusion et de l'éradication de ce programme.

Un **virus** est une portion de code qui s'attache à d'autres logiciels, y compris les systèmes d'exploitation, afin de produire des dégâts fonctionnels. Un de ses effets indirects consiste à ajouter le virus à d'autres programmes. Les virus agissent localement mais peuvent se propager à travers les réseaux comme les worms. Ses conséquences peuvent être désastreuses car ils peuvent détruire complètement la structure logicielle d'une machine affectée.

---

[3] Ce type d'attaque est classique et consiste à essayer d'estimer le texte en claire des données chiffrés.

A la différence des worms, les virus ne s'exécutent pas indépendamment, ils nécessitent qu'un programme "hôte" soit exécuté pour qu'il puisse être activé. Un virus est une séquence d'instructions qui se copie dans d'autres programmes de façon à ce que l'exécution de ce dernier active l'exécution de cette séquence. Ainsi, en analogie avec les virus biologiques, les "virus informatiques" ne sont pas considérés "vivants" dans le sens usuel du terme ; en fait ils envahissent les programmes des machines pour les corrompre et éventuellement pour produire d'autres virus.

Un virus n'apparaît pas spontanément. Un opposant doit trouver de l'introduire un dans le système cible, normalement en usant de la ruse, pour amener un utilisateur légitime à placer le virus dans ce système. Cela peut être fait en utilisant une application du type **cheval de Troie**, par exemple.

Selon Fred Cohen, le premier à utiliser le terme virus informatique de façon formelle, ce terme désigne un programme capable d'infecter d'autres programmes en les modifiant pour y inclure une copie probablement évoluée de lui-même[4] [Coh84].

Parmi d'autres exemples de programmes malicieux nous pouvons citer les **bactéries**, qui se reproduisent jusqu'à ce que toutes les ressources disponibles dans la machine cible soient absorbées et les **bombes logiques**, qui s'exécutent lorsque des conditions spécifiques sont satisfaites, par exemple, lorsqu'une certaine date est atteinte.

Il existe déjà une vaste littérature concernant les virus informatiques dans le contexte de machines personnelles, par exemple [Dav88, Hig88a, Hig88b, SHF89]. Pour une analyse de l'impact des virus informatiques dans l'environnement de recherche et de développement, voir [Bis91]. Ce travail présente également une breve histoire des attaques par virus et des autres programmes malicieux dans cet environnement. Pour plus des détails sur la théorie et l'expérimentation des virus informatiques, voir l'ouvrage de Cohen : [Coh90, Coh87, Coh88, Coh89]. Adleman a développé dans [Ade88] une théorie abstraite des virus informatiques. Pour les programmes malicieux en général voir [Den90].

Si avant ces types d'attaques (piratage, programme malicieux) étaient lancés pour satisfaire un challenge intellectuel ou par souhait de démontrer les limitations des mécanismes de sécurité mis en place ou pour attirer l'attention et faire de la publicité, ou même par une fausse manipulation ou erreur, aujourd'hui ces mêmes attaques peuvent être utilisées pour dévoiler ou pour détruire des informations confidentielles ou sensibles. Les cibles et les conséquences de telles attaques s'accentuent avec l'accroissement de l'interconnexion de systèmes et de la diversité de services accédés à travers les réseaux.

Des programmes envoyés ou récupérés par messagerie électronique ou à partir des BBS (Bulletin Board Systems) peuvent être altérés durant leur transmission, sur un des noeuds intermédiaires ou même dans le site source. A titre d'exemple, au début des années 80, un programme posté sur le réseau USENET contenait une commande pour supprimer tous les fichiers du système où le programme était exécuté.

---

[4] Cohen prétend que le premier virus informatique a été écrit par lui-même pour présentation lors d'un cours en Novembre 1983. Dans sa thèse de doctorat, il désigne son superviseur, L. Adleman, comme l'auteur du terme.

Vis-à-vis de cette discussion, notre vision globale du réseau doit être telle que ce système consiste d'un nombre de réseaux interconnectés dont la sécurité n'est pas garantie. Le système peut accidentellement ou volontairement enregistrer, perdre, doubler, générer ou modifier des messages qui y transitent. Lorsqu'un nouveau système est connecté au réseau, tout un ensemble de menaces et d'attaques provenant du réseau doit être pris en compte.

CHAPITRE 2
SERVICES DE SECURITE

L'objectif de l'OSI est de faciliter l'interconnexion des systèmes hétérogènes afin de permettre la communication entre des processus d'application [X.800]. Des contrôles de sécurité doivent donc être établis afin de protéger l'échange d'information entre ces processus. De tels contrôles doivent rendre le coût d'obtention ou de manipulation non autorisée des données plus important que la valeur potentielle de ces données, ou rendre le temps d'obtention ou de manipulation des données tellement important que la valeur des données soit perdue.

Du point de vue de l'utilisateur, lorsqu'il se sert d'un service à travers le réseau, il peut avoir besoin de s'assurer d'un ou de plusieurs des aspects suivants :

(1) que l'identité de la partie communicante est celle qu'elle prétend avoir et que celle-ci reste la même tout au long de la session,

(2) que personne d'autre n'est capable d'observer (et comprendre) les informations échangées durant la session,

(3) que personne n'est capable de modifier les informations échangées durant la session sans que cela soit détecté,

(4) que les engagements faits durant la session peuvent être prouvés postérieurement par un arbitre impartial.

De même, le fournisseur du service en question peut souhaiter s'assurer des aspects ci-dessus plus les suivants :

(5) que personne d'autre que les utilisateurs légitimes ne soit capable d'accéder au service,

(6) qu'il peut, si nécessaire, prouver à un arbitre impartial que l'utilisateur a véritablement utilisé ses services.

Dans la terminologie OSI [X.800], les services de sécurité nécessaires pour répondre aux besoins de (1) à (6) ci-dessus sont respectivement : l'authentification, la confidentialité, l'intégrité, la non-répudiation et le contrôle d'accès. Le service de non-répudiation répond aux besoins (4) et (6).

2.1. Authentification

L'identification consiste à associer une identité à chaque entité du système pour permettre de distinguer les entités entre elles de façon non ambiguë. L'*authentification* est la procédure de confirmation d'une identité. Lorsqu'un utilisateur utilise un service à travers le réseau, il doit être capable d'authentifier de façon fiable la partie communicante, qu'il soit un autre utilisateur ou un service. De même, la partie communicante doit être capable de s'assurer de l'identité de l'utilisateur initiateur de la communication.

Nous pouvons distinguer deux classes (ou modes) d'authentification :

(1) L'authentification des parties communicantes (*peer entity authentication*) consiste en la confirmation de l'identité de chaque entité participant à une communication. Ce mode d'authentification a normalement lieu lors de l'établissement d'une association dans des communications en mode connecté. Ainsi, ce mode d'authentification ne peut pas garantir que l'autre partie reste la même tout au long de l'association.

(2) L'authentification de l'origine des données (*data origin authentication*) consiste à associer des données à l'identité de son expéditeur, permettant ainsi au destinataire de déterminer si ces données proviennent de l'utilisateur prétendant être l'expéditeur.

Dans certaines circonstances, outre l'authentification des entités communicantes, il peut être aussi nécessaire d'authentifier le système à partir duquel l'utilisateur accède au service pour garantir :

- qu'un utilisateur légitime n'accède pas au service à partir d'une machine non autorisée. Par exemple, lorsqu'un utilisateur est menacé chez l'ennemi et obligé d'accéder à un service pour récupérer des informations ou obtenir des privilèges au profit de ce dernier.

  Un autre exemple est celui d'un utilisateur légitime qui accède volontairement à un service mais à partir d'un système dont la sécurité ne peut pas être garantie.
- qu'un utilisateur non autorisé accède au service ayant découvert des données nécessaires à l'authentification appartenant à un utilisateur autorisé.

2.2. Intégrité

L'*intégrité* des données est d'une importance primordiale dans tout service d'information. Toute information sensible doit être protégée contre la falsification et la modification accidentelle ou préméditée. Même les informations considérées au premier abord comme sans valeur, peuvent être manipulées, systématiquement ou aléatoirement, par exemple par un concurrent ou par un pirate casuel.

Deux types de service d'intégrité existent : ceux désignés pour protéger contre des modifications aléatoires et accidentelles, et ceux désignés pour protéger contre des modifications tramées intentionnellement pour détourner le mécanisme d'intégrité.

Le premier type de service d'intégrité vise à détecter tout changement accidentel des données transférées, avec recouvrement d'erreurs lorsque possible. Cette fonctionnalité est normalement offerte par les couches OSI de liaison de données et de transport.

Le deuxième type de service d'intégrité vise à détecter non seulement des modifications accidentelles des données mais aussi celles occasionnées par une attaque active. Dans un tel cas, deux types d'intégrité des données peuvent être considérées :

- *intégrité de messages* (ou de contenus) : garantit qu'un message ne peut pas être modifié sans que cela ne soit détecté,
- *intégrité de séquence de messages* : garantit qu'aucun message d'une séquence n'est supprimé, ni dupliqué, et que l'ordre originel de la séquence ne peut être modifié sans que cela ne soit détecté.

2.3. Confidentialité

La *confidentialité* constitue un aspect normalement moins critique que l'intégrité des données. La plupart du trafic transitant à travers un réseau n'a pas d'intérêt pour un intrus, et toute la masse d'information englobant ces données rend difficile à trouver les portions importantes d'information. Toutefois, une partie de l'information transférée par le réseau est sensible (i.e., non publique ou d'accès restreint) et ne doit être révélée qu'aux destinataires autorisés.

Parmi les portions d'information les plus critiques, on peut citer les clés de chiffrement distribuées à travers le réseau. Parfois, des documents entiers sont confidentiels et nécessitent d'être protégés, tels que transactions bancaires, offres, actes ou rapports de vente.

Le service de confidentialité consiste à rendre illisible une portion de données à des utilisateurs non autorisés ou à rendre l'accès non autorisé à ces données plus coûteux que la valeur de découverte de l'information.

2.4. Non-Répudiation

Pour que les transactions puissent être effectivement réalisées "électroniquement" entre systèmes d'information, il est essentiel de fournir des moyens pour l'établissement d'engagements par la voie électronique. Le service de *non-répudiation* consiste à fournir la preuve :

- de l'origine des données, afin de protéger le destinataire contre une fausse dénégation de la part de l'expéditeur d'avoir envoyé ces données,
- de la réception des données, afin de protéger l'expéditeur contre une fausse dénégation de la part du destinataire d'avoir reçu ces données.

Dans le cadre de services existants, la non-répudiation est très importante pour les services du type EDI, où des factures, des ordres de payement, etc., sont manipulées électroniquement.

La réalisation de ce service exige l'existence d'une tierce partie impartiale dont le rôle est de juger les cas de litige. Ainsi, la non-répudiation constitue un domaine complexe à cause des aspects

légaux impliqués. Avant que des engagements électroniques puissent avoir lieu et être pris en considération devant un tribunal, un effort législatif non négligeable doit être mené, surtout en ce qui concerne les engagements inter-frontières.

2.5. Contrôle d'Accès

L'élaboration d'un système distribué ou ouvert doit définir précisément quels usagers et quels systèmes ont droit à chaque service du système. Il est aussi nécessaire d'avoir les moyens de mise en vigueur cette politique. Le contrôle d'accès sert à "filtrer" l'accès à un service de façon à ce que seulement les utilisateurs autorisés puissent y accéder.

La fonction de contrôle d'accès doit donc vérifier que les accès aux divers services et données d'un système sont réservés aux entités autorisées. Cette fonction contrôle également l'accès aux divers types d'opérations d'un service donné. Ce service peut être établi, d'une manière simple, par une matrice dont les lignes représentent les entités du système et les colonnes représentent les ressources. Chaque case de cette matrice indique les opérations autorisées à l'entité de la ligne sur le ressource de la colonne.

Ce service n'est pas demandé par les utilisateurs d'un service mais imposé par le fournisseur du service. En effet, du point de vue de l'utilisateur, le contrôle d'accès représente plutôt une restriction qu'un service. Ainsi, le contrôle d'accès est un service appartenant aux domaines des applications et de l'administration. Basé sur l'authentification, ce service vérifie les droits et privilèges d'un utilisateur par rapport à un certain service ou information. Par conséquent, le fonctionnement du service de contrôle d'accès dépend fortement du service d'authentification sous-jacent et de la protection de la matrice mentionnée ci-dessus.

2.6. Analyse des Services de Sécurité

Bien qu'il soit difficile de prioriser les services de sécurité dans le cadre des systèmes ouverts, car les priorités dépendent de l'usager, du type d'application et de son utilisation, nous pouvons tout de même faire quelques observations :

(1) Le service intégrité est inutile si appliqué de façon isolée. Si l'on ne peut pas être sûr de l'identité de l'expéditeur de l'information, comment savoir si l'identificatfion n'a pas été falsifiée ? Ainsi, le service d'intégrité doit être combiné avec le service d'authentification pour être de réelle valeur. De même, l'authentification exige l'intégrité dans la mesure où une procédure d'authentification fiable ne peut pas employer des données d'authentification falsifiables.

Il est possible d'annuler la fonctionnalité d'un de ces deux services s'ils ne sont pas combinés. Considérons une attaque lancée par un ennemi E contre la sécurité de la communication entre deux parties A et B. E est tel qu'il est capable d'intercepter la communication entre A et B. Par exemple, E peut être l'opérateur d'une passerelle entre deux réseaux.

Supposons que E intercepte un message entre A et B, contenant des donnés d'authentification et un contenu quelconque. Si une fonction d'intégrité n'est pas appliquée à tout le message, E peut modifier à son choix le contenu du message sans toucher les

données d'authentification. En réception du message, *B* vérifiera les données d'authentification et considérera que ce message provient de *A* alors qu'en fait il provient de *E*.

Si l'intégrité du message est garanti mais l'authentification ne l'est pas, *E* peut intercepter le message et substituer l'identification de *A* par la sienne. En réception du message, *B* vérifiera les données d'authentification et considérera que ce message provient de *E* alors qu'en fait il provient de *A*. Si un tel message constitue, par exemple, une demande de brevet pour une nouvelle invention, les conséquences peuvent être très graves.

Par conséquent, les services d'authentification et d'intégrité doivent être toujours combinés pour éviter les types d'attaques décrits ci-dessus.

(2) Bien que la confidentialité implique l'intégrité, elle dépend également de l'authentification. Si une information secrète n'est pas strictement liée à l'identité de l'expéditeur, un attaquant pourra substituer cette information secrète par une autre également protégée sans que le destinataire en tienne compte.

(3) En ce qui concerne le service de non-répudiation, tandis que l'authentification permet de déterminer "qui" est la partie communicante ou de "qui" provient une certaine portion de données, la non-répudiation permet de prouver ces faits à un juge impartial.

Ainsi, si d'une part la non-répudiation dépend de l'authentification, d'autre part la non-répudiation peut être vue comme une forme plus forte d'authentification.

(4) Le contrôle d'accès est aussi fortement dépendant de l'authentification. Nous ne pouvons pas appliquer un mécanisme de contrôle d'accès pour un service quelconque sans s'être assuré de l'identité de la partie requérante au préalable.

Comme conséquence des aspects (1) à (4) ci-dessus nous concluons que :

- les services de sécurité sont très liés entre eux,
- le service d'*authentification* est à la base de tous les services de sécurité.

Avec la connectivité disponible à nos jours et la croissance à venir, le besoin le plus urgent est la disponibilité de procédures fiables d'authentification. Avec la pratique courante d'utilisation de mots de passe et de transferts en clair de données d'identification, il devient facile d'accéder à des services sous une fausse identité.

Dans le chapitre suivant nous discuterons des mécanismes utilisés pour réaliser ces services de sécurité.

CHAPITRE 3
MECANISMES DE SECURITE

Les mécanismes de sécurité constituent les moyens de réalisation des services de sécurité. Si traditionnellement les techniques de sécurité des communications se limitaient à offrir une protection contre les attaques passives, l'approche aujourd'hui est d'utiliser des techniques d'authentification et d'intégrité pour offrir une granularité plus fine pour la protection contre les attaques actives. En fait, ces deux types de technique doivent être utilisés en conjonction avec des mesures de sécurité physique et des contrôles personnels.

Dans le cas des systèmes locaux, la sécurité peut être, en grande partie, réalisée par des mesures physiques de sécurité. En contrepartie, dans le contexte de réseaux interconnectés, il ne serait pas possible de garantir la sécurité des communications au moyen de telles mesures. Dans un tel cas, nous employons de techniques *cryptographiques*.

Selon Desmedt [Des92], la *cryptographie* est la science et l'étude de la protection des données dans leur forme brute contre l'action malicieuse. Casser les méthodes cryptographiques constitue toute une science, appelée *cryptanalyse*. L'ensemble constitué par la conception des méthodes de chiffrement (cryptographie) et les recherches pour les casser (cryptanalyse) est généralement appelé la *cryptologie*.

Beaucoup de travail a été fait dans le domaine de la cryptologie dans le dernier millenaire. Cependant, le progrès le plus signifiant a eu lieu après l'avènement de l'informatique et du calcul électronique. Aujourd'hui nous disposons de nombreux algorithmes, techniques et mécanismes cryptographiques pouvant être utilisés en conjonction avec les architectures de communication ouvertes. Nous discuterons ici les classes principales de mécanismes cryptographiques, à savoir : les mécanismes de chiffrement, d'authentification, d'intégrité, de signature numérique.

Notre étude ne concerne pas les aspects théoriques de la cryptologie. Notre motivation ici est de donner un aperçu des résultats importants dans ce domaine et de montrer que nous ne pouvons rien attendre de la nature des algorithmes de chiffrement, mais qu'il est important de permettre une flexibilité suffisante afin d'accommoder des nouveaux algorithmes qu'inévitablement surgiront.

En outre, il est tout de même important de discuter les mécanismes de sécurité dans un niveau de détail suffisant pour montrer la viabilité d'implémentation des solutions qui seront proposées dans ce travail, et pour lier celui-ci plus fermement à la réalité.

Pour avoir une vue exhaustive de l'histoire de cryptographie, il est recommandé de se plonger dans l'ouvrage de Kahn [Kah67]. Celui de Kranakis [Kra86] traite les aspects théoriques. [Bra88] présente une introduction à la cryptologie moderne.

3.1. Mécanismes de Chiffrement

L'objectif d'un *mécanisme de chiffrement* ou *crypto-système* est rendre l'information illisible à toute partie non autorisée. Les messages à chiffrer, appelés *texte en clair* (plaintext), sont transformés grâce à une fonction paramétrée par une *clé*. La sortie du processus de chiffrement est appelée *texte chiffré* (ciphertext) ou *cryptogramme*, i.e., une forme inintelligible à toute partie n'ayant pas connaissance de la fonction de déchiffrement.

Un mécanisme de chiffrement est basé sur un algorithme public[5] paramétrisé par au moins une *clé*[6] choisie aléatoirement parmi un ensemble très vaste de valeurs. Une clé de chiffrement est une portion d'information secrète ou publique pouvant être représentée numériquement et utilisée en conjonction avec un algorithme public pour former un algorithme cryptographique.

Du point de vue d'un espion (ou cryptanalyste), le problème de la cryptanalyse se pose à trois niveaux :

- S'il ne possède que des textes chiffrés et jamais de textes en clair, il est confronté à un problème dit de *texte chiffré seulement* (*ciphertext only*). Les cryptogrammes publiés sous la rubrique "puzzle" des journaux posent typiquement ce genre de problème.
- Si l'espion dispose de certains textes chiffrés et des textes en clair correspondants, le problème devient un problème de *texte en clair connu* (*known plaintext*).
- S'il a la capacité de chiffrer des éléments de texte en clair de son choix, il a un problème du type *texte en clair choisi* (*chosen plaintext*).

Bien qu'intuitivement, on puisse croire qu'une méthode de chiffrement est sûre si elle résiste à la cryptanalyse du type "texte chiffré seulement", en fait le travail de cryptanalyse devient plus facile dès lors que l'espion dispose de plusieurs paires texte en clair/texte chiffré. Pour que la sécurité soit réelle, le cryptographe doit être sûr que la méthode de chiffrement est incassable

---

[5]Selon le principe de Kerckhoffs [Mas88], seul un algorithme qui a été ouvert à la critique publique et à la cryptanalyse pour une période raisonnable de temps pour être considéré comme un algorithme sûr. Les algorithmes maintenus secrets peuvent, intentionnellement ou non, comporter de *trapdoors* et d'autres défauts.

[6]A la différence des algorithmes de chiffrement, qui n'évoluent qu'à l'échelle des années, la clé peut changer aussi souvent qu'il le faut. Ainsi, le modèle de chiffrement est une méthode générale, stable et bien connue, paramétrisable avec une clé que l'on peut changer facilement.

même si l'espion est capable de chiffrer lui-même une quantité quelconque de textes en clair de son choix.

Historiquement, on peut diviser les méthodes de chiffrement traditionnelles en deux catégories : les méthodes de *substitution* et les méthodes de *transposition*. Dans un chiffrement par substitution, chaque lettre ou groupe de lettres est remplacée par une autre lettre ou groupe de lettres. Ainsi, ce type de méthode conserve l'ordre des caractères du texte en clair qu'il se contente de déguiser. Les méthodes de chiffrement par transposition, au contraire, changent l'ordre des caractères sans les déguiser.

La cryptographie moderne utilise les idées de base de la cryptographie traditionnelle, à savoir substitution et transposition, mais pas de la même manière. La cryptographie traditionnelle mettait en œuvre des algorithmes relativement simples et des clés plutôt longues pour assurer la sécurité des crypto-systèmes. Aujourd'hui, c'est l'inverse : on conçoit des algorithmes complexes, souvent basés sur un problème mathématique difficile à résoudre, de telle sorte que le cryptanalyste ne peut rien faire même s'il a accès à une grande quantité de textes chiffrés.

Ainsi, si à l'époque de Jules César[7] on utilisait des alphabets pour effectuer les substitutions et les transpositions lettre par lettre, aujourd'hui celles-ci sont mises en œuvre bit à bit par un nombre raisonnable des circuits appelés *boîtes-S* (S pour substituition) et *boîtes-P* (P pour permutation). Ces différentes boîtes sont employées en cascade de sorte que la sortie soit une fonction non linéaire de l'entrée.

3.1.1. Crypto-systèmes à Clé Secrète

La cryptographie conventionnelle est basée sur le partage d'une clé secrète, en sorte qu'une seule clé soit utilisée pour le chiffrement et le déchiffrement. Par exemple, l'expéditeur d'un message utilise une clé pour le chiffrer, et le destinataire utilise la même clé pour le déchiffrer, comme montre la figure 1. Cette méthode est connue comme *cryptographie à clé secrète* ou *symétrique*. La cryptographie symétrique date des temps anciens mais a été développée comme science exacte depuis 1945 à partir du célèbre article de Shannon, publié d'abord comme rapport secret, plus tard déclassifié et rendu public en 1949 [Sha49].

---

[7]La plus ancienne méthode de chiffrement connue est attribuée à Jules César.

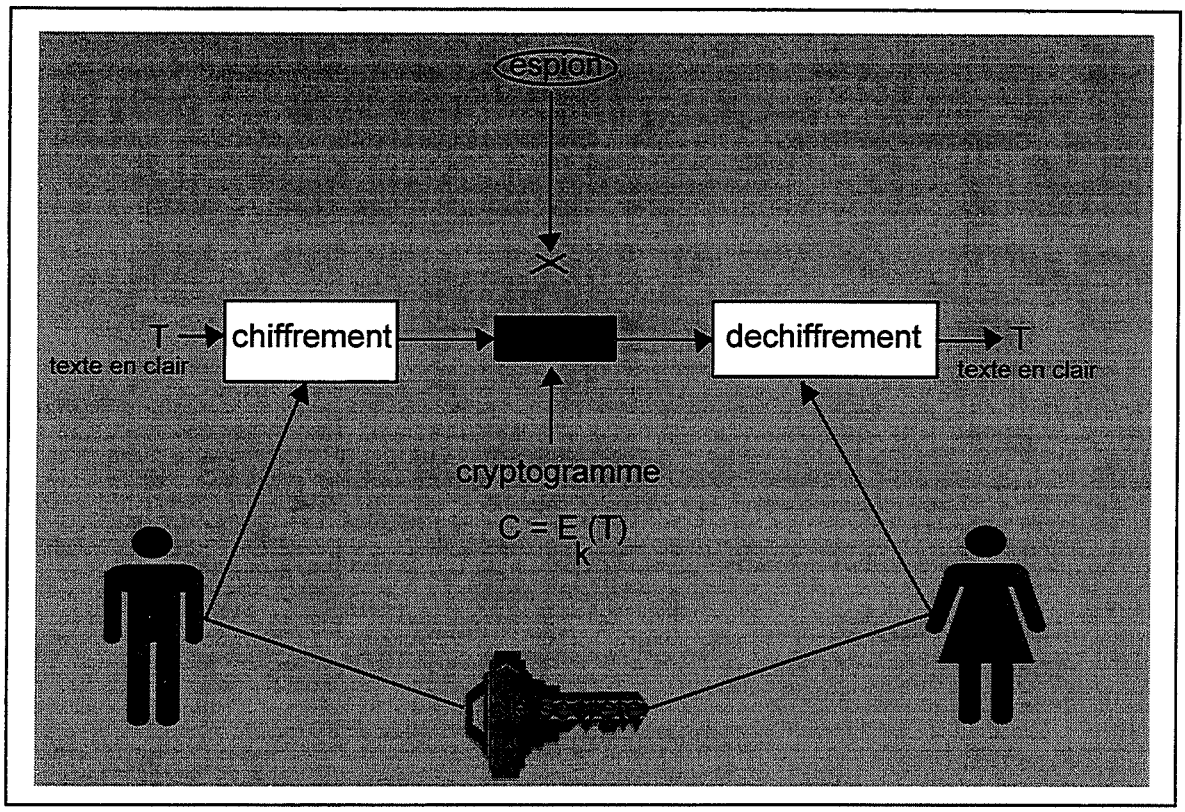

Figure 1. Modèle de Chiffrement Symétrique

L'algorithme symétrique le plus connu et utilisé de nos jours est l'algorithme DES (Data Encryption Standard), développé par la société IBM par sollicitation (faite en 1973) par le NBS (National Bureau of Standards des Etats-Unis).

DES est un algorithme de chiffrement de blocs de données (*block cipher*), i.e., chaque bloc d'un message est chiffré independament des autres. L'algorithme utilise une clé de 56 bits pour chiffrer des blocs de 64 bits, fournissant un cryptogramme de 64 bits. Les standards DES définissent la procédure standard de chiffrement ainsi que quatre modes d'opération[8], à savoir, ECB, CBC, CFB et OFB. L'algorithme DES et ses modes d'opération peut être trouvé dans [ANSI81, ANSI83].

Le mode **E**lectronic **C**ode **B**ook (ECB) constitue le mode de base, chiffrant independament chaque bloc de 64 bits l'un après l'autre. Les autres modes d'opération transforment cet algorithme en une méthode de chiffrement de chaînes (*stream cipher*). Avec ce type de méthode, chaque bloc chiffré dépend du texte en clair précédant. Ainsi, une séquence répétée dans le texte en clair ne générera pas une séquence répétée dans le texte en chiffré, ce qui rend difficile la cryptanalyse.

Dans les modes **C**ipher **B**lock **C**haining (CBC), **C**ipher **F**eed**B**ack (CFB) et **O**utput **F**eed**B**ack (OFB), avant de chiffrer chaque bloc de données on calcule le OU EXCLUSIF entre celui-ci et le bloc chiffré précédemment. Ainsi, le chiffrement d'un même bloc peut résulter en des différents cryptogrammes selon le contexte global du message.

---

[8]ces modes sont applicables à tout algorithme de chiffrement de blocs de données.

En particulier, le mode CFB permet le chiffrement de blocs de données inférieurs à 64 bits. Les modes CBC et OFB sont appropriés pour le calcul de *check-sums* cryptographiques (valeurs de vérification d'intégrité de données).

Des définitions similaires existent sous la forme de standards internationaux. La norme [IS8372] définit les modes d'opération pour le chiffrement de blocs de 64 bits et la norme [IS10116] pour des blocs d'une taille arbitraire.

Parmi les bonnes propriétés de DES nous pouvons citer les suivantes :

- DES est un algorithme largement accepté avec de nombreuses implémentations efficaces en support matériel,
- il est raisonnablement léger au niveau du calcul,
- il dispose de plusieurs modes d'opération normalisés et appropriés à plusieurs types d'usage,

Malgré sa popularité, l'utilisation de DES doit être découragée dans un futur proche. Le gouvernement américan avait déjà annoncé l'annulation de la certification du DES en 1988 [Hel87], ce qui n'a pas eu lieu. De plus, apparement cet algorithme n'est plus utilisé par ses solliciteurs pour la protection de ses données sensibles [Zim93]. Cela probablement à cause de la taille reduite des clés de 56 bits, ce qui est suffisament court pour réussir une attaque par force brute utilisant une machine de propos specifique ayant un nombre massif de puces DES.

Outre l'algorithme DES, d'autres crypto-systèmes symétriques sont utilisés aujourd'hui, parmi lesquels nous citons l'algorithme FEAL (**F**ast **E**ncryption **AL**gorithm) [SM87] pour lequel des attaques réussies ont été publiéés par Biham et Shamir [BS91], bien que des nouvelles versions aient été proposées.

Ron Rivest a développé les algorithmes symétriques RC2 et RC4. En principe, ces crypto-systèmes peuvent être rendus aussi sûrs que nécessaire puisqu'ils utilisent des clés de taille variable. Cependant, ces algorithmes sont propriétaires à RSA Security Inc. et ses particularités n'ont pas encore été publiées.

Un probable successeur de DES peut être l'algorithme IDEA (**I**nternational **D**ata **E**ncryption **A**lgorithm), un schéma publié en 1990 par Lai et Massey [LM90]. Cet algorithme est en fait une amélioration de sa version précédente qui a été publiée sous le nom de IPES (**I**mproved **P**roposed **E**ncryption **S**tandard). IDEA chiffre des blocs de 64 bits, ayant des clés de 128 bits et donnant comme résultat des cryptogrammes de 64 bits. IDEA peut également être utilisé en mode CBC et CFB.

Basé sur un nouveau concept qui mélange des opérations de différents groupes algébriques, cet algorithme est plus efficace au niveau du calcul que DES dans des implémentations logicielles, et semble plus fort que ce dernier. Jusqu'à maintenant, IDEA a mieux résisté aux attaques cryptanalytiques que d'autres crypto-systèmes symétriques tels que le DES, FEAL [BS90, BS91]. Biham et Shamir ont examiné cet algorithme pour des faiblesses sans succès [Zim93].

Bien que l'algorithme IDEA semble prometteur, il n'a pas été suffisamment apprécié pour que sa sécurité soit confirmée. Toutefois, ce crypto-système connaît, à l'heure actuelle, un grand intérêt

de la part des cryptanalystes et est donc exposé aux tentatives d'attaque. Sa résistance accroît sa popularité et l'évidence de sa sécurité.

3.1.2. Le problème de la gestion des clés

L'utilisation d'un algorithme de chiffrement à clé secrète pose un problème : il nécessite que le récepteur utilise, pour déchiffrer le message, la même clé que l'émetteur. Se pose alors la question : comment mettre la clé secrète à disposition des parties communicantes sans qu'une partie non autorisée puisse y accéder ? Il faut alors règler le problème de la distribution sûre de clés.

Traditionnellement, une clé secrète est fabriquée par un "office central de distribution de clés". Deux exemplaires de la clé sont ensuite transmis aux parties communicantes par courrier personnel. Pour une banque ou tout autre système qui utilise les clés par centaines ou milliers, cette technique ne peut évidement pas convenir, surtout si des impératifs de sécurité imposent de changer les clés tous les jours.

La méthode la plus commode est alors de transmettre les clés par le réseau lui-même. Bien sûr, l'échange des clés doit être chiffré et déchiffré au moyen d'une clé qui ne soit pas déjà compromise, ce qui constitue un problème circulaire.

En fait tout crypto-système doit être accompagné des mécanismes qui garantissent la génération, la distribution et le stockage des clés de chiffrement. Ces mécanismes constituent la *gestion des clés*. La sécurité de ces mécanismes est très importante. En pratique, la plupart des attaques viseront plutôt ces mécanismes que les crypto-systèmes eux-mêmes.

Puisque toute partie ayant accès à une clé secrète peut lire toute information chiffrée au moyen de cette clé, la cryptographie symétrique complique particulièrement la gestion des clés : si deux personnes étrangères l'une à l'autre doivent communiquer secrètement en utilisant une technique cryptographique, elles doivent disposer d'un moyen sûr pour se mettre d'accord sur une clé n'étant connue que de ces deux personnes.

Un autre problème qui se posait est celui des *signatures*. Pourrait-on développer une méthode permettant de fournir, au destinataire d'un message électronique, un moyen de démontrer à d'autres parties que ce message a été émis par un expéditeur particulier, telle qu'une signature manuscrite ?

Face aux problèmes de distribution des clés et de signature, Diffie et Hellman ont en 1976 introduit le concept de *cryptographie à clé publique* ou *asymétrique* [DH76]. Dans ces nouveaux crypto-systèmes, chaque partie possède une paire de clés complémentaires, une clé de chiffrement (*la clé publique*) et une autre pour le déchiffrement (*la clé privée*), tel qu'il doit être impossible de calculer la dernière à partir de la première.

Plus précisément, la fonction de chiffrement doit être une fonction à sens unique du type *trapdoor*[9]. Une fonction à sens unique est une fonction facile à calculer mais difficile à inverser.

---

[9]Il s'agit d'une "astuce" ou "piège" permettant, à la partie en ayant connaissance, de casser un schéma.

La propriété de "trapdoor" permet que la partie ayant connaissance de *l'astuce* soit capable d'inverser la fonction.

Avec le concept de cryptographie asymétrique, Diffie et Hellman ont proposé une méthode de distribution de clés symétriques. Normalement, pour que n parties puissent se communiquer confidentiellement, la distribution de $n(n-1)/2$ clés secrètes exigerait la génération de ce nombre de clés au préalable et ensuite la remise sécurisée des n clés secrètes. Le fait que les clés publiques puissent être largement divulguées et que les secrets (clés privées) ne soient pas partagés, simplifie considérablement la gestion des clés.

Au moyen de l'algorithme Diffie-Hellman, deux parties A et B peuvent établir une clé de session symétrique K_{AB} à travers l'échange de paramètres publics K_A et K_B. La clé K_{AB} reste inconnue à d'autres parties et peut être utilisée pour l'échange de données confidentielles entre A et B. Grâce à ce schéma, n parties peuvent s'accorder confidentiellement sur $n(n-1)/2$ clés sécretes en utilisant n paramètres publics dont les besoins en sécurité sont l'authenticité et l'intégrité. Les clés secrètes sont générées localement à partir de ces paramètres publics.

Cette technique est toujours considérée comme l'une des meilleurs méthodes pour l'échange de clés secrètes [Fah92]. L'un des problèmes associés à ce schéma découle du fait que les parties communicantes doivent utiliser certains paramètres communs à savoir, le *générateur* et le *modulus* du cryptosystème. Outre cela, chaque utilisateur doit disposer d'un copie intégrale du paramètre publique de chaque autre utilisateur avec lequel ce premier souhaite établir un canal sécurisé. Cela exige la distribution préalable de ces paramètres en préservant leur authenticité et intégrité.

Pour que cela soit viable à grande échelle, il est nécessaire d'avoir une autorité globale capable de générer et distribuer largement ces paramètres. De plus, ce modulus doit être suffisamment grand car la compromission de celui-ci met en danger tous les utilisateurs du groupe. Outre cela, du fait que la clé secrète résultante est partagée par les parties communicantes, la méthode ne se prête pas au service de non-répudiation sans l'emploi d'une tierce partie de confiance.

Cet ensemble de restrictions rendent l'algorithme Diffie-Hellman convenable seulement dans des environnements où il est possible l'accord sur les paramètres communs ainsi que la maintenance et la distribution des copies intégrales des paramètres publiques de tous les utilisateurs.

Malgré ces inconvénients, l'algorithme Diffie-Hellman est la base de la cryptographie à clé publique et a été utilisé comme base pour la conception d'autres schémas de distribution de clés [OT89, Roe93] et est encore en utilisation aujourd'hui [Fah92].

L'algorithme Diffie-Hellman est la base de la cryptographie à clé publique. Cet algorithme a été utilisé comme base pour la conception d'autres schémas de distribution de clés [OT89, Roe93] et est encore en pratique aujourd'hui [Fah92].

3.1.3. Crypto-systèmes à Clé Publique

A partir du concept de cryptographie asymétrique, chacun peut divulguer sa clé publique et doit maintenir secrète la clé privée. Le besoin de partager une clé secrète est alors éliminé : toute

communication sécurisée emploie seulement les clés publiques, et les clés privées ne sont jamais transmises ni partagées.

Dans ces termes, un algorithme asymétrique permet de réaliser le service de confidentialité car seulement la partie ayant la clé privée est capable de déchiffrer le message. L'utilisateur E souhaitant envoyer un message secret à D, obtient la clé publique de B et l'utilise pour chiffrer le message. En réception, D utilise sa clé privée pour déchiffrer le message. Ainsi, toute partie peut envoyer une information confidentielle à D et aucune autre partie observant la communication ne peut déchiffrer cette information. Bien entendu, l'une des exigences d'un tel crypto-système est que l'on ne puisse pas calculer la clé privée à partir de la clé publique et que la clés privée soit maintenue secrète par son proprietaire.

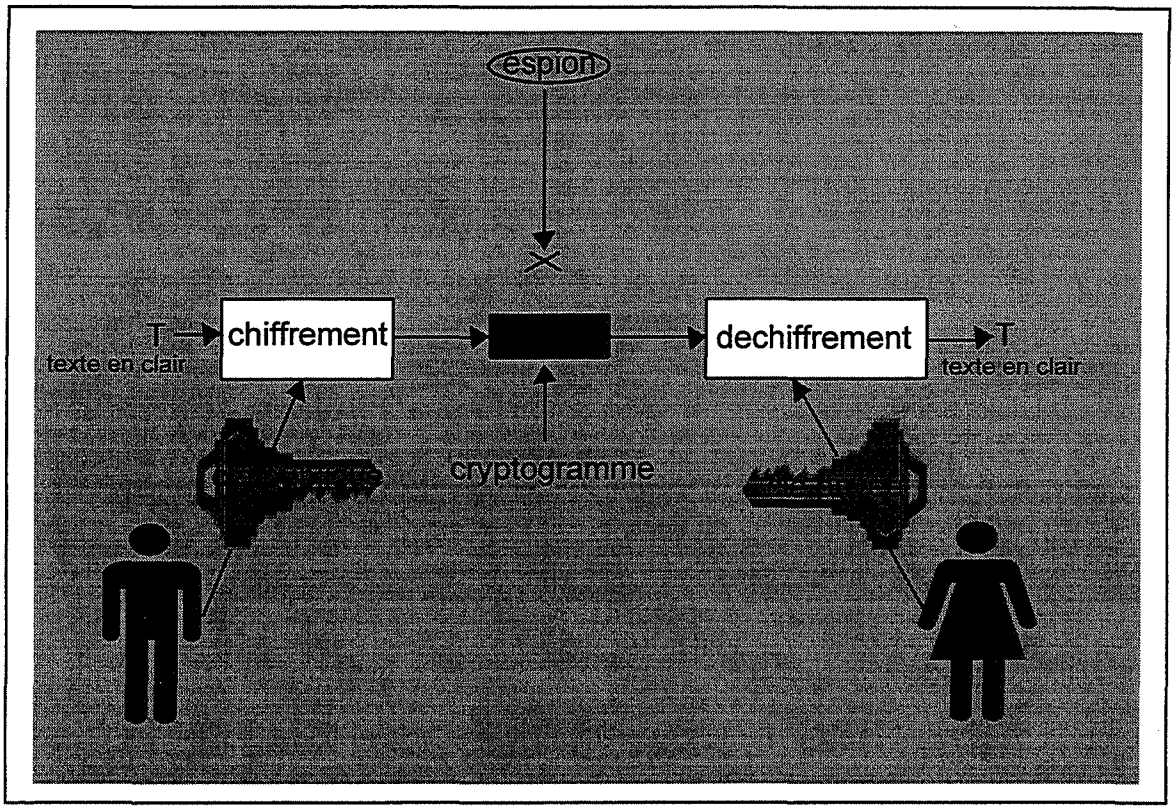

Figure 2. Modèle de Chiffrement Asymétrique

Après le mécanisme de Diffie et Hellman, nombre de crypto-systèmes à clé publique ont été proposés. Les premiers crypto-systèmes asymétriques sont les *trapdoors knapsacks* inventés par Merkle et Hellman [MH78] et le RSA [RSA78]. Le *knapsack* est une classe des problèmes mathématiques NP-complets. Ce problème a été utilisé comme base d'un nombre de crypto-systèmes, mais ceux-ci ont perdu leur crédibilité car plusieurs versions ont été cassées. Malgré cela, d'autres versions améliorées sont encore considérées comme assez sûres [LLH89].

L'algorithme RSA a été inventé en 1977 par Ron Rivest, Adi Shamir et Leonard Adleman [RSA78]. Cet algorithme est probablement le crypto-système à clé publique le plus utilisé (et utilisable) aujourd'hui. Etant basé sur l'exponentiation modulaire de nombres premiers, le fonctionnement de RSA est le suivant :

Prenons deux grands nombres premiers p et q, et trouvons leur produit $n = pq$; n est appelé *modulus*. Choisissons un nombre e, tel que $e < n$ et que e soit relativement prime à $(p-1)(q-1)$. Trouvons son inverse d, $mod\ (p-1)(q-1)$, ce qui signifie :

$$ed = 1\ mod\ (p-1)(q-1).$$

Les nombres e et d sont appelés exposants public et privé, respectivement. La clé publique est le pair (n,e) et la clé privée est d.

Lorsqu'un utilisateur A envoie un message confidentiel m à l'utilisateur B, A calcule le cryptogramme c, tel que :

$c = m^e\ mod\ n$, où e et n constituent la clé publique de B.

Pour déchiffrer le message c et obtenir le information originelle m, B calcule :

$m = c^d\ mod\ n$, où d est la clé privée de B.

De cette sorte, toute la sécurité de RSA repose sur la difficulté de factorisation. Casser cet algorithme est équivalent à *factoriser* un grand nombre entier. A l'heure actuelle, des clés d'une taille de l'ordre 500 bits sont considérés raisonnablement sûre et des clés d'une taille d'environ 1000 bits sont considérés sûres

Casser cet algorithme est équivalent à décomposer un grand nombre entier en facteurs premiers. A l'heure actuelle, des clés d'une taille de l'ordre 500 bits sont considérées raisonnablement sûres et des clés d'une taille d'environ 1000 bits sont considérées sûres

Cet algorithme a fait l'objet de plus d'études, d'applications et de tentatives d'attaque que tout autre crypto-système à clé publique. Les derniers 15 ans d'expérimentation de RSA ont renforcé sa popularité et l'évidence empirique de son assurance. Parmi ces inconvénients nous pouvons citer :

- la taille de blocs à chiffrer dépend de la taille des clés,
- inefficacité au niveau du calcul.

En des implémentations logicielles, le DES est 100 fois plus rapide que le RSA. Dans des implémentations en support matériel, le DES est entre 1000 et 10000 fois plus rapide que le RSA, selon l'implémentation. Les implémentations de RSA en support matériel les plus rapides aujourd'hui atteignent un débit supérieur à 64 Kbits par seconde avec un modulus de 512 bits. Des spécialistes prévoient des pastilles pouvant atteindre un débit égal ou supérieur à 1 Mbits par seconde avec un modulus de 512 bits [Fah92].

Puisque casser l'algorithme RSA est équivalent à la factorisation d'un grand nombre entier, les avances dans le domaine de la factorisation exigent l'augmentation de la taille des clés. En considérant la possibilité d'une avancée définitive dans ce domaine, nous ne devons pas baser la totalité des développements sur cet algorithme mais nous devons être capables d'y adapter rapidement des nouveaux algorithmes à clé publique.

Un autre crypto-système à clé publique conçu par ElGamal [ElG85], a été la base d'autres schémas de chiffrement, tel que celui proposé par Schnorr [Sch89]. En comparaison avec RSA,

l'algorithme ElGamal est plus lent pour le chiffrement/déchiffrement et génère des signatures plus longues.

D'autres crypto-systèmes proposés sont basés sur l'exponentiation discrète sur le champ fini $GF(2^n)$. Leur avantage est l'efficacité d'implémentations sur support matériel. Cependant, ces algorithmes ont été questionnés car le problème de base peut être plus facile à résoudre que la factorisation [Odl84, GM92].

Goldwasser et Micali [GM84] ainsi que Blum et Goldwasser [BG84] ont proposé des méthodes intéressantes de *chiffrement probabilistique*. Avec ces algorithmes, la même portion de données étant chiffrée deux fois au moyen d'une même clé donnera deux résultats différents. Ces méthodes ont donc l'avantage d'être résistantes aux attaques par dictionnaire (*guessed ciphertext attack*), mais ceci aux prix d'expansion des données [Fah92].

Une ligne récente de recherche en cryptologie est celle de *crypto-systèmes à courbe elliptique*. Ces crypto-systèmes sont basés sur des opérations mathématiques sur un groupe de points d'une courbe elliptique. Proposé en 1985 par V. Miller de la Société IBM et développé plus tard par N. Koblitz et d'autres, ce type de crypto-système a connu beaucoup de succès dans les dernières années. Il est attendu que ces méthodes deviennent des crypto-systèmes à clé publique plus sûrs que le RSA dans un futur proche. Ces crypto-systèmes sont encore considérés coûteux mais les progrès en VLSI aideront l'implémentation efficace de ces algorithmes. Koblitz [Kob87] et Miller [Mil85] présentent des introductions à ce sujet.

Un bon aperçu de l'histoire de la cryptographie à clé publique, décrit par l'un de ses inventeurs, peut être trouvé dans [Dif88].

3.1.4. Utilisation des mécanismes de chiffrement

L'avantage principal de la cryptographie à clé publique réside sur le niveau de sécurité : cette méthode n'exige ni le transfert ni le partage des clés privées. Dans les crypto-systèmes symétriques il existe toujours la possibilité de découverte de la clé secrète par un intrus pendant sa transmission.

Le désavantage principal des crypto-systèmes asymétriques est le niveau de performance. A cause de leur complexité, ces crypto-systèmes sont plus coûteux que les crypto-systèmes symétriques. Certaines méthodes symétriques de chiffrement sont sensiblement plus efficaces au niveau de la bande passante que les méthodes asymétriques existantes.

Ce problème de performance est particulièrement remarqué lorsque des données confidentielles sont envoyées à plusieurs destinations. Pour cela, les données doivent être chiffrées au moyen de la clé publique de chaque destinataire. Ainsi, pour un nombre n de destinataires, les mêmes données sont chiffrées n fois, ce qui peut être fort laborieux selon la quantité de données à échanger et le nombre de destinataires.

La solution courante est donc de combiner les deux types de crypto-systèmes afin de profiter des avantages des crypto-systèmes asymétriques (sécurité) et des crypto-systèmes symétriques (efficacité).

Puisque les crypto-systèmes symétriques sont beaucoup moins coûteux que les asymétriques, il est préférable d'utiliser ces crypto-systèmes pour le chiffrement de grandes quantités de données et ensuite d'utiliser un crypto-système asymétrique pour chiffrer la clé utilisée pour chiffrer ces données. Ainsi, nous utilisons la clé publique de chaque destinataire seulement pour chiffrer la clé symétrique utilisée pour chiffrer les données proprement dites, qui sont chiffrées une seule fois.

Cela montre que la cryptographie à clé publique n'est pas considérée comme remplaçante de la cryptographie à clé secrète, mais plutôt comme complément pour rendre celle-ci plus sûre. Par ailleurs, la première utilisation de la cryptographie à clé publique a été la distribution sécurisée de clés symétriques sans besoin d'un échange préalable de secrets; ceci reste encore l'une des fonctions principales. La cryptographie à clé secrète reste extrêmement utile et fait encore l'objet de travaux de recherche.

Bien que les crypto-systèmes soient très complexes, l'existence d'algorithmes cryptographiques forts ne suffit pas à elle seule à résoudre les problèmes de sécurité pour lesquels ces algorithmes ont été conçus. L'algorithme cryptographique doit être utilisé dans un ensemble de règles et procédures pour la manipulation des données cryptographiques, à savoir un *protocole* qui assure que l'algorithme cryptographique offre bien le niveau et le service de sécurité exigé par le système. Cela est particulièrement noté dans la mise en oeuvre de mécanismes d'authentification, comme nous le remarquerons tout au long de ce travail.

La conception d'un protocole de sécurité inclut la spécification des caractéristiques des algorithmes cryptographiques pouvant être utilisées dans le protocole afin de rendre le service souhaité sans dégradation de la sécurité du système. Cette tâche peut nécessiter d'une logique permettant d'établir formellement la fonctionnalité et le niveau d'assurance du protocole.

Dans les sections suivantes nous présenterons un aperçu de l'utilisation des techniques cryptographiques pour la réalisation des principaux services de sécurité.

3.2. Mécanismes d'Authentification

Les mécanismes d'authentification visent à confirmer l'identité d'une entité. Ces mécanismes reposent sur la démonstration de possession d'un secret ou d'une caractéristique associée à un utilisateur, par exemple :

- un mot de passe,
- une clé de chiffrement secrète,
- une carte magnétique munie d'un code secret,
- une empreinte digitale ou rétinienne.

Nous prenons en considération deux types de procédure d'authentification :

- les procédures d'authentification simple, qui peut faire usage de mots de passe,
- les procédures d'authentification forte, qui font usage de techniques cryptographiques.

Puisque basées sur l'utilisation de mots de passe, les procédures d'authentification simple peuvent n'offrir aucune protection (lorsque les mots de passe ne sont pas utilisés) ou offrir une protection faible ou inadéquate (par exemple, l'usage de mots de passe non protégés). De plus, ces procédures sont normalement vulnérables au *re-jeu* (voir Section 1.2). Malgré ces inconvenients, les procédures d'authentification simple sont utilisées dans la plupart des systèmes actuels (par exemple, pratiquement dans tous les systèmes locaux et systèmes d'exploitation).

Les procédures d'authentification forte peuvent être réalisées au moyen de crypto-systèmes symétriques ou asymétriques.

3.2.1. Authentification à clé secrète

En utilisant des crypto-systèmes symétriques, l'authentification repose sur le partage d'une clé secrète entre les paires de parties communicantes. Plus le nombre n de parties communicantes s'élève, plus un tel système devient impraticable car le nombre nécessaire de clés est trop élevé pour être généré, distribué et conservé de façon sûre, chaque partie devant conserver un nombre $n-1$ de clés.

En outre, ces algorithmes ne peuvent pas réaliser le service de non-répudiation. Si une clé symétrique doit être partagée par au moins deux parties, comment prouver à un arbitre neutre, l'origine de données authentifiées au moyen de cette clé ? Si un utilisateur A produit un message authentifié au moyen d'une clé secrète partagée avec un autre utilisateur B, A pourra toujours prétendre que ce message a été créé par B puisque celui-ci est aussi en possession de la clé secrète. Dans un tel cas, rien ne peut être prouvé au juge.

L'utilisation de crypto-systèmes symétriques pour l'authentification requiert donc la disponibilité d'un *serveur de clés* ou *d'authentification* de haute fiabilité et confiance, servant, à la fois à associer les clés sécrètes à chaque utilisateur et à établir des clés de session entre ces utilisateurs pour le service de confidentialité. Il doit être noté qu'un tel serveur a le potentiel de lancer toute

sorte d'attaque passive ou active contre ces utilisateurs puisqu'il a accès aux secrets des utilisateurs. L'exemple d'un tel serveur est présenté dans la Partie II de ce travail.

Un serveur d'authentification peut fonctionner comme une sorte de "notaire électronique" pour qu'un utilisateur ne puisse pas nier sa participation dans une opération en prétendant que le secret partagé a fait l'objet de compromission ou a été utilisé par une autre partie ayant accès légitime au secret.

L'utilisation de crypto-systèmes asymétriques élimine les limitations imposées pour les crypto-systèmes symétriques mais pose d'autres problèmes, comme nous le verrons dans les paragraphes suivants.

3.2.2. Authentification à clé publique

Bien que les algorithmes asymétriques puissent être utilisés pour réaliser le service de confidentialité, ceux-ci ne permettent pas, en principe, d'offrir le service d'authentification car toute partie ayant la clé publique aurait pu chiffrer le message. Toutefois, certains crypto-systèmes asymétriques ont la propriété de la *permutabilité* des clés, i.e., de permettre l'usage des deux clés de la paire pour le chiffrement et le déchiffrement. Autrement dit,

$$X_p \cdot X_S = X_p \cdot X_S,$$

où, X_p est la fonction de chiffrement et X_S la fonction de déchiffrement. Ces crypto-systèmes[10] peuvent donc être utilisés pour l'authentification. Avec de tels crypto-systèmes asymétriques, un utilisateur peut prouver son identité sans révéler ses données secrètes d'identification[11]. L'utilisateur A souhaitant prouver son identité, chiffre une portion d'information au moyen de sa clé privée. Puisque A est le seul à avoir accès à sa clé privée, toute partie ayant accès à la clé publique de A peut vérifier l'origine de cette information en la déchiffrant. En outre, puisqu'une clé privée n'est connue que par son propriétaire, les algorithmes asymétriques permettent de réaliser le service de non-répudiation. En particulier, un autre avantage de RSA sur d'autres crypto-systèmes asymétriques repose sur cette propriété.

---

[10]Les cryptosystèmes asymétriques ayant la propriété de la permutabilité sont aussi appelés*mécanismes de signature à clé publique.*

[11]Le besoin d'authentifier un utilisateur, sans que ceci ait à reveler son information secrète d'identification, a été soulevé dans [Sim89], qui présente un schéma de vérification d'identité capable de fournir des accusés de transaction infalsifiables pour utilisation dans un arbitrage postérieur.

3.2.3. Le problème de gestion des clés publiques

Bien que le concept d'authentification à clé publique soit plutôt élégant, il pose des difficultés de réalisation qui exigent la conception de mécanismes de gestion.

Tout d'abord, un destinataire D ne peut prétendre qu'un message lui a été envoyé par un utilisateur S, et par lui seul, tant que la clé privée de S reste secrète. Si S ne maintient plus cette clé secrète, cette argumentation n'est plus valable : n'importe qui pourrait avoir envoyé le message, même D.

Une telle situation peut apparaître, par exemple, si S et D sont des sociétés commerciales. L'équipe de direction de S peut, à un moment donné, s'apercevoir qu'avoir envoyé un tel message était une erreur commerciale. Pour renier le message envoyé, la société S peut rendre publique sa clé privée et appeler la police en prétendant qu'il vient d'y avoir un cambriolage et que la clé privée a été volée. Selon les lois locales en vigueur, la société peut être tenue responsable ou non de l'usage qui a été fait de la clé volée.

Un autre problème se pose lorsque S veut changer sa paire de clés. Une telle opération est de toute évidence légale. Le changement de clés est même une opération courante. Cela peut être nécessaire, par exemple, pour faire face à l'évolution des capacités de calcul et de la cryptanalyse ou en cas de perte de la clé privée.

Si un cas de litige se produit peu après, le juge appliquera la nouvelle clé publique de S pour vérifier l'ancienne signature et découvrira qu'il n'obtient pas le résultat correct. Quel recours D aura-t-il alors ? Ainsi, il est nécessaire de disposer d'une "haute autorité" pour enregistrer toutes les modifications de clés et leurs dates.

Puisque les clés privées ne doivent pas être partagées, les fonctions de cette autorité doivent être restreintes au niveau des clés publiques. D'autant plus que l'utilisation de clés publiques pour la confidentialité est sujette à la menace d'usurpation d'identité dans la mesure où un attaquant peut essayer de remplacer la clé publique d'un utilisateur par la sienne, de manière à pouvoir intercepter toute information confidentielle envoyée à cet utilisateur.

Cette autorité doit donc associer l'identité de chaque utilisateur à sa clé publique en garantissant l'authenticité et l'intégrité de ces données, de telle sorte que la validité d'utilisation de la clé privée correspondante dépend de l'aval donné par l'autorité.

Une telle autorité est dite *Autorité de Certification* car elle produit des *certificats* portant garant de la validité et de la propriété des clés publiques. Le certificat sert de lettre d'introduction et permet aux parties communicantes d'échanger leurs clés publiques afin, par la suite, de communiquer en sécurité.

La Recommandation X.509 [X.509] définit les bases pour l'implémentation d'une telle approche. Cependant, nous allons montrer que cette norme ne couvre pas tous les aspects de la certification et que la gestion de clés reste un problème très délicat, surtout dans un environnement ouvert et hétérogène. Cela constitue l'un des points clé de notre travail.

3.2.4. Protocoles d'authentification et techniques à zero-connaissance

L'échange d'information nécessaire à l'authentification d'entités est communément appelé un *protocole d'authentification.* Outre la démonstration de la possession d'un secret, ces protocoles comprennent normalement l'échange de *challenges* et/ou d'estampilles de temps, de façon à ce que chaque partie s'assure que l'information reçue ne constitue pas un re-jeu de données d'authentification déjà utilisées.

Les protocoles d'authentification ont été largement étudiés [NS78], [DS81], [NS87], [DH79], [Moo88], [IM90], [BAN89]. Dans la Partie II de ce travail nous décrirons les protocoles d'authentification définis dans le cadre d'authentification X.509 qui ont été utilisés comme base des procédures d'authentification de l'annuaire X.500 et de la messagerie X.400.

Un domaine de recherche récent et qui connait un intérêt croissant est celui des *protocoles à zero-connaissance*. Ceux-ci constituent la classe de schémas d'authentification où une entité peut prouver son identité sans révéler un seul bit de son information secrète d'authentification. Dans ces schémas, le vérificateur (ou un observateur/pirate) ne peut rien apprendre qui pourrait l'aider à usurper l'identité de la partie authentifiée. Ainsi, ces techniques peuvent être utilisées pour des fins de non-répudiation.

Dans ces techniques, l'information secrète d'authentification de chaque partie joue un rôle très semblable à celui des clés privées des crypto-systèmes asymétriques mais ne peut pas toujours être utilisée pour la confidentialité. L'une des plus importantes propriétés de ces techniques est que celles-ci éliminent le besoin de changement périodique de clés afin de les protéger contre la cryptanalyse.

Avec une telle méthode, la partie vérificatrice envoie des challenges choisis aléatoirement au prétendant. Celui-ci applique ses données privées d'authentification pour calculer un résultat capable de prouver son identité. A son tour, le vérificateur peut vérifier ce résultat sans obtenir aucune information à propos du secret du prétendant et sans avoir à maintenir aucune information sur ce dernier.

Ces techniques sont plus souvent moins complexes au niveau du calcul que les crypto-systèmes à clé publique habituels. En contrepartie, ces schémas emploient souvent plusieurs cycles de challenge/réponse entre les deux parties afin d'atteindre le niveau souhaité de confiance *sur* l'identité de la partie authentifiée. Le niveau de sécurité de la procédure d'authentification dépend donc du nombre de cycles du protocole de base [FP90].

Certaines techniques à zero-connaissance simplifient la gestion de clés puisqu'elles éliminent le besoin de clés publiques différentes pour chaque utilisateur. Un inconvénient de ces techniques, en comparaison avec les crypto-systèmes à clé publique, est que celles-ci exigent que les clés privées soient générées par une tierce partie (i.e., une autorité de confiance) et qu'elles ne peuvent pas toujours être utilisées pour la confidentialité [FS86].

La première technique à zero-connaissance a été proposée par Shamir en 1984 [Sha84]. Une autre approche importante est présentée dans [FFS87]. [QG89] donne une très agréable introduction aux techniques à zero-connaissance.

3.3. Mécanismes d'Intégrité

Traditionnellement, dans les réseaux de communication de données, l'intégrité d'un message est assurée (avec une certaine probabilité) en ajoutant à la fin du message une sorte de "bloc (ou total) de contrôle" (*block check*), qui est normalement un CRC (**C**yclic **R**edundancy **C**heck). Cette technique est suffisante pour détecter une modification accidentelle sur un message. Cependant, une telle technique n'est pas suffisante pour protéger contre une modification malveillante. En fait, un attaquant peut intercepter un message, le modifier, recalculer son bloc de contrôle et remettre le message sans que le changement ne soit détecté.

Il est donc nécessaire de protéger ce bloc de contrôle contre des modifications malveillantes. Cela demande que le bloc de contrôle soit incalculable par une tierce partie, ce qui peut être réalisé en utilisant des techniques cryptographiques.

Afin de minimiser la bande passante, les techniques cryptographiques d'intégrité de données utilisent souvent une *fonction de scellement* pour le calcul du bloc de contrôle (similaire au CRC), appelé MDC (**M**odification **D**etection **C**ode) ou ICV (**I**ntegrity **C**heck **V**ector), qui est ensuite chiffré pour former un bloc de données appelé *Contrôle d'Intégrité de Message* ou MIC (**M**essage **I**ntegrity **C**heck).

Ainsi, une tierce partie n'ayant pas connaissance de la clé de chiffrement ne peut pas modifier le MIC sans que cela soit détecté par le récepteur du message en question. Le destinataire d'un tel message peut recalculer le MDC et, soit le chiffrer avec la même clé symétrique utilisée par l'expéditeur, soit déchiffrer le MIC reçu avec sa clé publique, pour comparer le résultat avec le MIC ou le MDC originel. En particulier, l'inclusion de numéros de séquence ou de segment dans chaque message permet la protection du trafic contre l'insertion, suppression ou mise en désordre des messages dans une séquence.

Une fonction de scellement (ou de *hash*) est une fonction qui prend comme entrée des données de taille variable et qui retourne un résultat de taille fixe. Bien que les fonctions de scellement aient diverses applications dans la programmation, en cryptographie ces fonctions sont utilisées pour générer une portion réduite de données pouvant représenter une portion de données beaucoup plus longue (par exemple, un fichier quelconque). Un MIC peut ainsi être calculé sur une portion réduite de données alors que les données originelles sont d'une taille quelconque.

Pour qu'une fonction de scellement soit de haute assurance pour l'utilisation en cryptographie, il ne doit pas être possible d'obtenir facilement le même résultat de scellement à partir de combinaisons différentes d'un message d'entrée. Une telle fonction[12] devra satisfaire aux exigences suivantes :

- la fonction doit être à sens unique (non inversable ou difficile à inverser), i.e., quel que soit le résultat de scellement possible, il doit être impossible par calcul de construire un message d'entrée qui est réduit à ce résultat,
- la fonction doit être exempte de collision, i.e., il doit être impossible par calcul de construire deux messages d'entrée distincts qui se réduisent au même résultat.

---

[12]Des telles fonctions sont aussi appelées *message-digest*.

Ainsi, un résultat de scellement peut être rendu public sans révéler les données originelles. L'absence de ces propriété ci-dessus permettrait à un attaquant de remplacer un message par un faux message sans que cela soit détecté. Des exemples de fonctions de scellement à sens unique ayant ces propriétés sont MD4, MD5 et SHS. [Mer89] discute l'utilisation de l'algorithme de chiffrement DES pour le calcul de fonctions de scellement à sens unique.

MD2, MD4, et MD5 sont des fonctions de scellement conçues par Ron Rivest. Largement utilisées aujourd'hui, ces fonctions produisent des résultats de 128 bits et sont considérées comme assez sûres. Il est supposé qu'un effort d'un ordre de 2^{128} opérations est nécessaire pour trouver un message dont la fonction de scellement donne un résultat donné, et un effort d'un ordre de 2^{64} opérations pour trouver deux messages ayant un même résultat de scellement.

MD2 [Kal92] est la fonction la plus lente parmi les trois, MD4 [Riv90] est la plus rapide et MD5 [Riv92] est considérée comme la plus sûre mais 33% plus lente que MD4. Cependant, il existe une extension de MD4 produisant des résultats de 256 bits [Riv90].

La fonction MD5 est la plus recommandée pour le calcul de signatures. En particulier, cette fonction fait partie du protocole SNMP (**S**ecure **N**etwork **M**anagement **P**rotocol) de l'Internet [CFSD90]. L'application de cryptanalyse différentielle par Biham et Shamir [BS91] à l'algorithme MD5 n'a révélé aucune attaque viable à celle-ci. En contrepartie, cette expérimentation a révélé des faiblesses dans deux autres fonctions de scellement, à savoir N-hash [MOI90] et Snefru [Mer89].

SHS (Secure Hash Standard) [NIST92b] est une fonction de scellement proposée par le NIST[13]. Cette fonction produit des résultats de 160 bits et est structurellement similaire à MD4 et à MD5. Cependant, SHS est 25% plus lente que MD5 mais peut être plus sûre car elle produit des résultats 25% plus longs que les fonctions MD. SHS n'a pas encore été formellement adoptée par le NIST comme standard gouvernemental.

Des organismes de normalisation étudient la normalisation d'algorithmes pour le calcul des MIC. Par exemple, la norme ISO8731-1 suggère la transmission des messages accompagnés d'un MIC constitué du dernier bloc du résultat de chiffrement du message avec DES en mode CBC. En réception, le destinataire chiffre le message et compare le dernier bloc du résultat avec le MIC reçu.

Il doit être noté que dès qu'une technique d'intégrité utilise un algorithme de chiffrement symétrique, il est nécessaire que les parties communicantes se fassent confiance mutuellement. Puisque ces parties ont accès au message originel et à la clé de chiffrement commune utilisée pour le calcul du MIC, ces parties sont capables de falsifier le message (et recalculer le MIC) après la transmission, de nier l'envoi ou la réception du message originel et même d'accuser l'autre partie de falsification.

Ainsi, un mécanisme d'intégrité convenable doit être accompagné d'un moyen d'assurer l'origine du message. Avec les crypto-systèmes symétriques une telle fonctionnalité ne serait pas possible sans l'existence d'une tierce partie de confiance globale, une sorte de "notaire électronique" ayant la tâche d'arbitrer les transactions en estampillant les messages. Cette fonctionnalité peut être

---

[13]SHS a été en principe designée pour utilisation dans le cadre de DSS (**D**igital **S**ignature **S**tandard).

réalisée en utilisant des crypto-systèmes asymétriques, sans besoin d'une source externe de confiance, à travers les *signatures numériques* ou *électroniques*. Nous décrirons cette technique dans la section suivante.

3.4. Signatures Électroniques

Dans le monde réel, les procédés légaux font une grande différence entre les documents originels et les copies. Par exemple, les photocopies n'ont pas de valeur légale. Un autre aspect lié concerne les signatures manuscrites. L'authentification de documents légaux, fiscaux ou administratifs est déterminée par la présence ou l'absence de signatures manuscrites dûment mandatées. Dans le cas des transactions électroniques, il est nécessaire de trouver des solutions permettant de viabiliser ces mêmes procédures.

La conception d'un mécanisme remplaçant la signature manuscrite est particulièrement complexe. Dans un système où l'un des correspondants peut envoyer un message "signé" à l'autre, il faut essentiellement que :

- le destinataire puisse vérifier l'identité prétendue de l'expéditeur,
- la signature ne soit pas falsifiable,
- l'expéditeur ne puisse nier plus tard avoir envoyé le message.

La différence majeure entre la signature manuscrite et électronique est que cette dernière ne peut pas être constante ; elle doit être une fonction du message signé. En ayant cette propriété, une même signature ne pourra pas être attachée à n'importe quel message, et la modification d'un seul bit du message produira une signature différente. Ainsi, un message signé ne peut être falsifié sans que cela soit détecté.

Le concept de signature électronique ou numérique a été introduit par Diffie et Hellman [DH76]. D'autres études dans le domaine peuvent être trouvées dans [Akl83], [PK79], [Rab78]. Le principal désavantage des schémas conventionnels de signature électronique est que ceux-ci sont normalement non réutilisables. Typiquement, la signature est générée aléatoirement pour un message donné et utilise une grande quantité de matériel cryptographique. De plus, la résolution postérieure de disputes requiert des agréments par écrit et la maintenance d'une quantité considérable d'information de la part de l'expéditeur et du destinataire, ce qui rend le jugement difficile. L'exemple d'un tel schéma est montré dans [DH76] et amélioré dans [Mer82]. D'autres schémas conventionnels sont basés sur l'algorithme DES.

Les systèmes de cryptographie à clé publique peuvent être d'un grand secours pour résoudre ce problème. Pour qu'un crypto-système asymétrique puisse être utilisé pour le calcul des signatures électroniques, il doit avoir la propriété de la permutabilité, i.e., il doit aussi permettre le chiffrement au moyen de la clé privée (voir 3.2.2).

Lorsque réalisée au moyen d'un crypto-système asymétrique, une signature électronique d'un message est typiquement une portion redondante d'information dépendante des contenus du message. Ceci de façon à ce que seule la partie possédant la clé privée puisse produire la signature, mais qu'elle puisse être vérifiée par tous au moyen de clé publique correspondante. Cette signature prouve donc l'origine et l'intégrité du message et permet la réalisation du service de non-répudiation d'origine.

Un schéma de signature fiable doit avoir les propriétés suivantes :

- il doit être impossible de calculer une signature valide sans la connaissance de la clé privée,
- il doit être impossible de générer deux messages différents avec la même valeur de signature.

Pour signer une information (info) on lui ajoute un résumé chiffré de l'information. Ce résumé est produit au moyen d'une fonction de scellement à sens unique *h*, tandis que le chiffrement est effectué au moyen de la clé privée du signataire X_S :

$$X\{info\} = Info, [h(info)]X_S$$

L'information *info* elle-même peut, soit rester en clair si elle n'est pas confidentielle, soit être chiffrée. Si l'information n'est pas confidentielle, les noeuds intermédiaires peuvent lire son contenu sans le besoin de déchiffrer la signature.

Le chiffrement de la signature utilisant la clé privée garantit que la signature ne peut pas être falsifiée. Du fait que la fonction de scellement est non inversable, il devient impossible de produire une information truquée de façon à obtenir le même résultat de scellement (et ainsi la signature) afin de falsifier l'information signée.

Le destinataire de l'information signée vérifie la signature en :

(1) appliquant la fonction de scellement à l'information,

(2) comparant le résultat avec celui obtenu par déchiffrement de la signature au moyen de clé publique du signataire.

Si les résultats ne sont pas égaux, soit l'information a été modifiée, soit la clé publique utilisée pour la vérification ne correspond pas à la clé privée utilisée pour la génération de la signature. Le mécanisme est illustré par la figure 3.

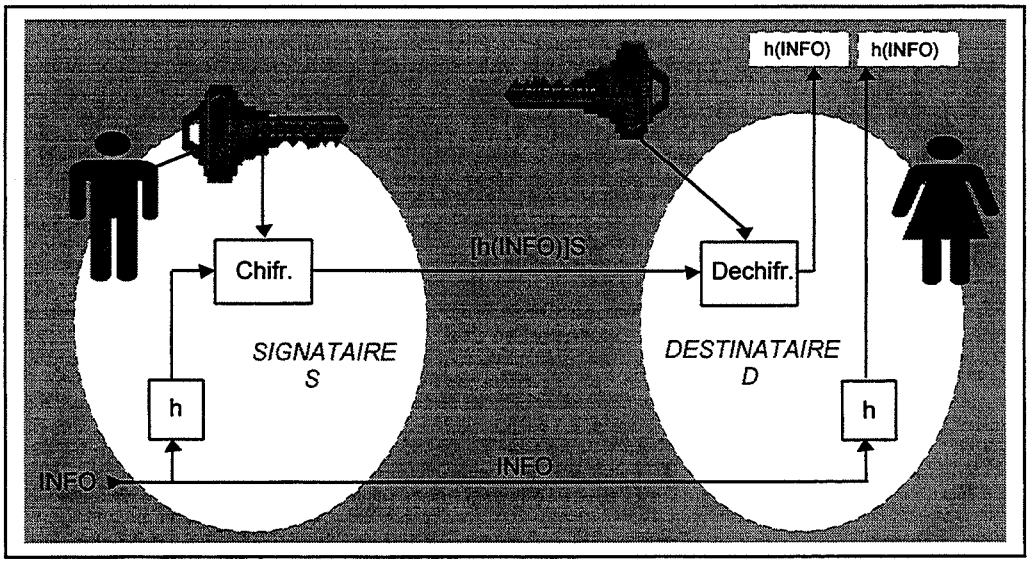

Figure 3. Signatures Numériques

Si un désaccord survient ultérieurement, le destinataire D peut produire au tribunal les trois pièces justificatives qui permettront aux juges de vérifier que c'est bien le signataire S qui a envoyé le message. En effet, même si D avait fabriqué lui-même le message *info*, il aurait pu calculer *h(info)*, mais en aucun cas il ne pourrait reproduire $[h(info)]X_S$, puisqu'il n'a pas accès à la clé privée de S.

Ce mécanisme est particulièrement intéressant du fait qu'un résultat de scellement très court suffit quelle que soit la longueur du message, évitant ainsi l'inconvénient au niveau du coût des crypto-systèmes asymétriques.

Il doit être noté l'importance de la fonction à sens unique *h*. S'il était possible de trouver un message M correspondant à un h(M) donné, alors D pourrait tricher en générant un autre message M' tel que h(M') = h(M) pour fournir aux juges M', h(M) et $[h(M)]X_S$.

Les signatures numériques peuvent ainsi être utilisées pour réaliser plusieurs services de sécurité, tels que l'authentification, la non-répudiation et l'intégrité.

Un schéma de signature numérique peut être cassé par attaque soit à l'algorithme de chiffrement, soit à la fonction de scellement. Ainsi, il est recommandé de choisir un algorithme de chiffrement et une fonction de scellement ayant des niveaux d'assurance comparables, car la cible principale d'attaque sera toujours le composant le plus faible.

Par exemple, les fonctions MD peuvent être attaquées avec 2^{64} opérations, ce qui est comparable à l'effort nécessaire pour casser une clé RSA ayant un modulus de 512 bits. En pratique l'attaque aux fonctions MD est plus difficile que cela car elles exigent 2^{64} unités de mémoire et la capacité d'amener l'utilisateur à signer un message de son choix.

Ainsi, le couplage de la fonction MD5 à une clé RSA ayant un modulus de 512 bits est typiquement considéré comme un bon choix pour le calcul de signatures. Toutefois, pour les utilisateurs nécessitant un niveau de sécurité plus élevé, il est préférable d'utiliser un modulus RSA plus long (par exemple, 1024 bits) et une fonction de scellement qui produise des résultats plus longs, telle que la version de MD4 produisant des résultats de 256 bits ou la fonction SHS.

Rabin [Rab79] a proposé un schéma de signature numérique de fiabilité équivalente à la factorisation. Un autre schéma proposé par Fiat et Shamir [FS86] est basé sur des protocoles interactifs à zero-connaissance mais peut être adapté pour permettre le calcul de signatures. Ce schéma est plus efficace que le RSA et du même niveau d'assurance, mais les signatures sont plus longues que celles générées avec RSA ; bien que d'autres variations de ce schéma réduisent la taille des signatures. Ces méthodes peuvent être plus convenables aux applications du type carte à puce que pour l'authentification dans les systèmes ouverts [BDB92].

Le NIST a récemment proposé un standard de signature numérique pour le gouvernement des Etats-Unis appelé DSS (**D**igital **S**ignature **S**tandard). DSS spécifie un algorithme de signature appelé DSA (**D**igital **S**ignature **A**lgorithm). DSS est un dérivé du crypto-système à clé publique développé par Schnorr [Sch89] qui à son tour est inspiré du crypto-système proposé par ElGamal [ElG85]. DSS est donc basé sur le problème des logarithmes discrets. Une description détaillée de DSS se trouve dans [NIST92a].

DSS a été globalement reçu avec méfiance par les spécialistes en cryptographie et de l'industrie informatique [Mes92]. Les critiques les plus sérieuses concernent la sécurité de DSS et se sont concentrées sur les aspects suivants :

- le manque d'un schéma de gestion de clés,
- la taille de clés fixée à 512 bits,
- le manque de directives d'implémentation,
- l'état récent du crypto-système de base,
- la procédure à travers laquelle le NIST a choisi DSS a été considérée comme dissimulée, arbitraire et trop influencée par le NSA (National Security Agency).

Bien qu'une telle taille de clés soit suffisante pour des applications ordinaires, une clé de 512 bits n'est pas assez sûre pour des utilisateurs/applications qui nécessitent un niveau d'assurance plus élevé. De plus, dans quelques années cette taille de clés pourra ne plus être considérée sûre, même pour les besoins d'utilisateurs ordinaires. Ainsi, il est nécessaire que de tels algorithmes supportent une taille variable de clés de façon à ce que chaque utilisateur puisse choisir la taille de clés la plus appropriée à ses besoins. En réponse à cette critique, le NIST a annoncé que DSS sera révisé pour permettre des tailles de clés jusqu'à 1024 bits.

DSS n'a pas encore été suffisamment exposé à l'expérimentation et aux attaques. Bien que le problème de logarithmes discrets ne soit pas récent, la forme particulière du problème utilisée dans DSS a été déjà proposée pour l'utilisation en cryptographie en 1989 [Sch89] et n'a guère fait l'objet d'études.

De plus, des spécialistes ont soulevé l'existence de certains nombres premiers du type "*trapdoor*" dans DSS, i.e., des nombres permettant de casser facilement une clé lorsque utilisés pour la génération de celle-ci. Cependant, des tels nombres premiers sont rares et peuvent être évités si chaque utilisateur est capable de générer et de vérifier la qualité ses propres clés. Si les clés sont générées par une autorité centrale, chaque utilisateur doit disposer des moyens de vérifier la qualité de ses clés [SB88]. Une discussion détaillée des critiques autour de DSS peut être trouvée dans [NIST92c].

En ce qui concerne l'application des signatures numériques, celles-ci peuvent être utilisées, non seulement pour renforcer la sécurité des communications, mais aussi pour protéger des ressources locales de façon à prévenir contre le type d'attaque (2) mentionné dans la Section 1.2.

De la même façon que les mécanismes d'intégrité détectent des erreurs de transmission, ils peuvent être utilisés pour détecter des altérations dans un fichier conservé sur disque. Puisque la majorité des programmes malicieux, par exemple les virus, agissent en altérant ou remplaçant des fichiers, les mécanismes d'intégrité peuvent aider à protéger contre ce type d'attaque.

Selon Bishop[14] [Bis91], les programmes malicieux utilisent nombre de techniques bien connues des informaticiens mais dans un ordre inhabituel. Ainsi, bien que les mécanismes de sécurité

---

[14][Bis91] étudie l'impact des techniques conventionnelles de sécurité sur le traitement de virus en les analysant dans un cadre général.

existants ne soient pas spécifiquement conçus pour parer à ce type de problème, ces mécanismes ont été conçus pour traiter des techniques utilisées par les programmes malicieux.

Dans ces termes, des mécanismes de sécurité existants peuvent empêcher ou rendre difficile l'introduction et la prolifération de programmes malicieux, de la même façon que ces mécanismes peuvent réduire les probabilités ou limiter les dégâts causés par une attaque réussie contre l'intégrité des communications.

Une des approches est d'appliquer une signature numérique à chaque fichier, après s'être assuré de "l'immunité" de ce fichier. Ainsi, chaque utilisation du fichier peut être précédée de la vérification de sa signature. Si la vérification échoue, ceci indique la modification du fichier, probablement par un virus ou la substitution du fichier par une attaque du type cheval de Troie. Bien sûr, ce fichier aurait pu être modifié pour d'autres raisons (par exemple, la recompilation des sources ou par des problèmes physiques sur le disque). Si la modification semblait suspecte, l'utilisateur procéderait à l'exécution d'un logiciel spécifique de détection et d'élimination du problème en question.

Bien entendu, pour une attaque plus ingénieuse, le programme malicieux pourrait essayer de changer la clé publique et/ou le logiciel de vérification pour échapper à la détection. Ainsi, une méthode plus sûre doit incorporer le logiciel de vérification dans le système d'exploitation et fournir des mécanismes de protection à ce système d'exploitation.

CHAPITRE 4
MOTIVATIONS ET OBJECTIFS DE CE TRAVAIL

Au cours de cette partie, nous avons vu que l'accroissement constant des réseaux et des télécommunications dans les systèmes dits ouverts pose aujourd'hui un problème important : l'ouverture même de ces systèmes implique une réduction de sécurité. Dans le Chapitre 1, nous avons montré les menaces auxquelles ces systèmes sont sujets. Dans le Chapitre 2, nous avons décrit les services susceptibles de parer à ces menaces ; et dans le Chapitre 3 les techniques utilisées pour réaliser ces services. Il nous reste maintenant à utiliser ces mécanismes pour renforcer la sécurité des systèmes ouverts.

En particulier, nous avons aussi vu l'importance de méthodes fiables d'authentification et que ces méthodes nécessitent des mécanismes de gestion de clés publiques. La définition et l'implémentation consciencieuses de systèmes ouverts sécurisés est donc vitale pour l'avenir des réseaux de télécommunications.

Des travaux approfondis ont été menés dans le domaine de la cryptologie, ce qui a permis la conception des mécanismes cryptographiques utilisables, tels que les crypto-systèmes symétriques et asymétriques, les fonctions de scellement et les schémas de signature numérique. Aussi dans le domaine de la sécurité, le respect des normes facilite l'implémentation de solutions communes, minimise les investissements et permet de réutiliser le savoir-faire acquis. Nous allons voir dans la partie suivante de ce travail, que le modèle de référence OSI a été étendu pour inclure un cadre de sécurité [X.800]. L'activité croissante des organismes de normalisation dans le domaine de la sécurité est démontrée par le nombre de nouveaux standards dans le sujet [X.50993, X.500, X.40088].

Toutefois, peu de recherche a été menée entre ces deux extrêmes. Le modèle OSI se situe dans un très haut niveau d'abstraction, ce qui nous offre un cadre conceptuel mais peu de direction vers la construction de systèmes réels. A l'autre extrême, la plupart des travaux dans le domaine de la cryptologie se concentrent sur les propriétés mathématiques des crypto-systèmes particuliers. Tout au plus, des protocoles d'authentification ont été analysés sans égard pour leur position dans le modèle OSI et leur rapport avec les composants des systèmes ouverts [DM83]. Des groupes de

travail, comme le projet Password (CEC) et les groupes PEM (IETF) et WG-SEC (RARE) commencent à apparaître dans le domaine d'architectures pour la sécurité des systèmes ouverts.

Il existe donc un besoin pressant de travaux combinant ces deux lignes de recherche dans le domaine des communications sécurisées. A partir de cadres et de modèles génériques, des services et des mécanismes de sécurité doivent être définis et intégrés dans leurs propres places dans les systèmes ouverts. Ces mécanismes doivent être intégrés dans les services et protocoles existants, et notamment dans des outils d'administration. Finalement, des crypto-systèmes adéquats doivent être choisis pour répondre aux besoins en sécurité de ces services, protocoles et outils. Comme résultat de tels travaux, des solutions implémentables et interopérables aux problèmes de sécurité existeront. Ce travail vise ainsi à combiner l'expérience acquise, d'une part dans le domaine des protocoles de communication et d'interconnexion de systèmes ouverts, et d'autre part dans de domaine des techniques de sécurité. Puisque ce travail vise la mise en œuvre concrète de systèmes ouverts sécurisés, les aspects liés à l'implémentation seront pris en compte tout au long de ce travail.

La Partie II de ce travail présentera un aperçu de l'état de l'art dans le domaine de la sécurité pour les systèmes ouverts. Dans le premier chapitre de cette deuxième partie, nous décrirons divers schémas et modèles proposés pour la sécurité des réseaux et des applications ouvertes. Dans le deuxième chapitre de cette partie nous aborderons les aspects de normalisation des mésures de sécurité dans OSI. En particulier, nous présenterons dans cette partie, le cadre d'authentification X.509 [X509], cadre de base sur lequel notre étude s'appuie. Cette recommandation propose un cadre pour la distribution de clés publiques à travers l'annuaire X.500 [X500], et des protocoles d'authentification. Du fait du domaine étendu recouvert par l'annuaire, ce cadre a été désigné pour être utilisé par toute une gamme d'applications, voire faciliter la mise en place d'une infrastructure commune pour le services d'authentification dans le monde OSI. En tant que cadre de distribution de clés publiques, X.509 normalise le concept d'Autorité de Certification, spécifie la forme sous laquelle les données publiques nécessaires à l'authentification sont conservés dans l'annuaire et établit des hypothèses au sujet des procédures permettant de constituer ces informations .

La Partie III portera sur notre expérimentation dans le domaine de la sécurité pour les systèmes ouverts. Notre participation dans un projet européen de recherche, à savoir le projet *Password*, nous a permis de prendre connaissance des besoins en sécurité des applications ouvertes et des mécanismes pour y répondre. Nous présenterons rapidement, dans le premier chapitre de cette partie, les aspects abordés dans ce projet. Dans les deux chapitres qui suivent, nous exposerons notre contribution personelle dans le cadre de ce projet, à savoir la conception et le développement d'une architecture de sécurité, baptisée *SecUcomx*. Basée sur l'approche X.509, cette architecture permet la sécurisation des échanges de données dans un réseau ouvert et le développement d'applications sécurisées. Un autre aspect de notre contribution personelle dans Password est l'utilisation des modules de cette architecture pour la sécurisation des applications. Ceci fait également l'objet d'un chapitre de la Partie III.

Etant donné le besoin pressant de mécanismes d'authentification dans un environnement ouvert et hétérogène, nous nous sommes concentrés notamment sur les aspects de gestion des clés publiques et d'obtention de données publiques pour l'authentification dans un tel contexte. Nous avons donc étudié le rôle, la gestion et les fonctionnalités des Autorités de Certification ainsi que leur rapport avec l'annuaire X.500. En particulier, il est attendu que le problème d'accès aux données publiques necessaires à l'authentification devienne une question suscitant un interet

croissant. Pour répondre à ces besoins, nous avons crée un nouveau concept, à savoir le *DAS* (**D**ata **A**uthentication **S**erver). Celui-ci constitue un module independant de l'architecture SecUcomx dont le rôle est d'établir les informations publiques d'authentification. Le DAS constitue l'un des éléments clé de notre travail, comme l'on remarquera tout au long de la Partie IV.

L'étude et l'expérimentation de l'approche X.509 nous ont permis d'en détecter les obstacles à l'implémentation d'une infrastructure globale et commune pour les services d'authentification. Dans le dernier chapitre de la Partie III, nous soulèverons ces problèmes ainsi que des lacunes des modèles de sécurité de l'annuaire X.500 et de la messagerie X.400. Dans ce même chapitre, nous essayerons egalement de proposer des solutions à certains aspects non traités dans le cadre du projet Password.

Comme nous l'avons déjà mentionnée, les premières études dans le domaine de la sécurité pour les systèmes ouverts nous ont amené à réfléchir sur le problème complexe de l'établissement d'un canal de communication sécurisé entre deux entités quelconques, dans un environnement ouvert et hétérogène.

Si d'une part l'évolution de la technologie de réseaux de télécommunication permet aujourd'hui de créer un canal de communication entre deux utilisateurs quelconques indépendamment de la distance les séparant et des différences logicielles ou matérielles ; d'autre part cette capacité de connectivité et d'harmonisation n'a pas de correspondance au niveau de la sécurité, ce qui est quelque peu décevant, surtout du point de vue de l'utilisateur final. Ce problème, ayant comme origine l'authentification et la gestion des clés cryptographiques, constitue la motivation majeure de notre travail.

Nous avons constaté que si un nombre de modèles et schémas de sécurité ont été proposés, la plupart d'entre eux sont dédiés à une gamme spécifique d'applications ou d'utilisateurs. Ils ne constituent donc pas moins de mondes isolés. L'utilisateur final est ainsi confronté à tout un ensemble de solutions incompatibles entre elles. D'un autre point de vue, le cadre X.509, bien qu'il ait été désigné pour faciliter la mise en place d'une infrastructure commune pour l'authentification et qu'il constitue la base de nombre des schémas et modèles, il ne prend pas en compte l'hétérogénéité de l'environnement dans lequel il se situe et contient de nombreux problèmes ouverts. Or, la mise en réseau de l'économie mondiale doit être accompagnée des mécanismes de sécurité capables de soutenir les nouveaux enjeux supportés par les systèmes d'information.

En tenant compte de l'importance des services d'authentification pour la sécurité des systèmes ouverts et prenant en considération l'ensemble de problèmes et besoins exposés, la Partie IV de notre travail développe une nouvelle approche et de nouveaux concepts, allant au-delà du cadre X.509 et ayant pour objectif de rendre viable la mise en place d'une infrastructure globale pour les services d'authentification. Ceci en respectant l'hétérogénéité du système global, les politiques de sécurité des diverses communautés et la conformité avec les standards actuels. Cette approche a été conçue autour de l'evolution du concept du DAS, qui nous avons créé initialement pour l'architecture SecUcomx.

Dans le premier chapitre de la Partie IV, nous exposerons le problème traité. Nous partons, en fait, de deux problèmes centraux de gestion de clés, à savoir la distribution initiale de clés et l'obtention de clés entre les parties communicantes. Ceux-ci sont considérés en fait comme des

réquisits par les services de sécurité. Ces problèmes s'aggravent dans un environnement hétérogène car la gestion de clés doit aussi permettre l'authentification d'entités pouvant être des diverses catégories d'utilisateur et des ressources telles que des serveurs et des applications. Notre approche a pour but d'offrir ainsi, aux procédures d'authentification, des mécanismes spéciaux permettant, à la fois : le respect de l'hétérogénéité du système et des politiques de sécurité des diverses communautés d'utilisateurs, ainsi que l'indépendance vis-à-vis des modèles de certification et des services de noms.

Dans le deuxième chapitre de la Partie IV, nous proposerons une réorganisation modulaire des architectures de sécurité pour permettre d'accommoder différentes solutions et politiques, tout en conservant l'interopérabilité et la sécurité des mécanismes. Dans le troisième chapitre, nous aborderons le problème de vérification de l'information publique d'authentification. Nous proposerons donc une méthode originale pour la description de politiques de certification de clés publiques permettant de rendre explicite l'hétérogénéité des divers modèles de certification et de dériver des critères et métriques de validation de clés publiques pour aboutir à un mécanisme modulaire d'évaluation des données publiques nécessaires à l'authentification. Finalement, le quatrième chapitre approfondit et évolue le rôle et les fonctionnalités du DAS qui nous proposerons comme un nouvel élément, complémentaire aux architectures de gestion de clés, permettant d'alléger les applications, d'être indépendant de services de noms, des formats des données. Cela constitue en fait, un moyen de permettre la réalisation de toute l'approche, bien que chaque module de celle-ci soit indépendant l'un de l'autre.

L'implémentation d'une telle approche permettrait la construction d'architectures de sécurité homogènes du point de vue de l'utilisateur final. En effet, si de telles architectures deviennent viables, celles-ci permettront l'établissement d'un canal sécurisé entre deux entités quelconques. Par conséquent, la quasi-majorité sinon tous les utilisateurs souhaitera profiter des services sécurisés.

Enfin, dans la Partie V nous présenterons une conclusion sur l'apport de ce travail ainsi que des directions pour des travaux futurs dans le domaine traité.

Cette étude vise, en principe, le lecteur intéressé aux aspects de recherche et de développement de la sécurité pour les systèmes ouverts. Le modèle OSI est utilisé comme cadre général de référence, bien qu'une vision critique de ce modèle soit maintenue. Malgré cela, ce travail n'est pas dépendant ou exclusif au modèle et aux applications OSI. En fait, les solutions proposées dans ce travail peuvent être appliquées à toute architecture de communication, par exemple l'architecture Internet.

PARTIE II

ETAT DE L'ART

---

5 . MODELES ET ARCHITECTURES DE SECURITE

6. NORMALISATION DE LA SECURITE DANS OSI

CHAPITRE 5
MODELES ET ARCHITECTURES DE SECURITE

Après la création de l'ARPAnet, le monde de l'informatique a connu une grande diffusion des réseaux. Encore aujourd'hui, ces réseaux ne présentent pas des mesures adéquates pour garantir leur sécurité. Pour faire face aux menaces, nombre de recommandations modèles, schémas et recommandations ont été proposés. Le département de défense américain (DoD) a été le premier à publier une norme concernant les critères d'évaluation de la sécurité des systèmes informatiques [DoD85]. L'organisation européenne ECMA a aussi présenté une norme concernant la définition des services pour la sécurité des systèmes ouverts [ECMA89].

En ce qui concerne les modèles et schémas de sécurité, la plupart d'entre eux sont dédiées à une gamme précise d'applications. En fait, les premières études portaient essentiellement sur la confidentialité des communications dans les réseaux. L'ouvrage de Wen-Pai Lu [Lu86], par exemple, aborde le problème de la distribution de clés pour la confidentialité des communications sur le réseau Internet.

D'autres modèles ont été proposés pour le contrôle d'accès. Parmi ces modèles se trouve le projet UCLA [PKK+79], qui est une réalisation d'un système sécurisé offrant une interface Unix selon modèle Bell-LaPadula. Ce modèle est détaillé dans [LS89], et des implémentations de cette approche peuvent être trouvées dans [MD79] et [Fra83].

En ce qui concerne l'authentification dans un environnement distribué, le résultat le plus important est celui de *Kerberos*, le service d'authentification développé au MIT (Massachusetts Institute of Technology), dans le cadre de la sécurité de son projet d'informatique distribuée *Athena*. Kerberos est aujourd'hui dans sa version cinq[11].

[11]Les trois premières versions de Kerberos sont des versions de développement internes à ce projet, la quatrième a été rendue publique et diffusée à grande échelle dans les milieux industriels et universitaires. Puisque cette version a été conçue pour répondre aux besoins particuliers du projet Athena, elle ne convenait pas à d'autres types et architectures de réseau. La version cinq a pris en compte des nouveaux aspects et des nouvelles suggestions émises pendant la longue expérience avec la version précédante.

Le modèle de Kerberos est inspiré de celui proposé par Needham et Schroeder [NS78], et modifié par Denning et Sacco [DS81]. Ce modèle permet aux entités du système de s'authentifier les unes auprès des autres à travers une tierce partie de confiance, à savoir un serveur d'authentification, en utilisant la cryptographie conventionnelle[12]. Selon le modèle Needham-Schroeder, ce serveur est responsable du stockage de toutes les preuves d'identité secrètes de toutes les entités du système. La confiance dans ce serveur est la brique de base du modèle et le jugement des diverses entités du réseau sur l'authenticité des messages échangés repose sur cette confiance. Ce modèle offre également les services d'intégrité et de confidentialité à travers les clés symétriques partagées entre les parties communicantes. Le modèle de fonctionnement de Kerberos peut être trouvé dans [KN93].

Bien que Kerberos présente un modèle capable d'améliorer considérablement la sécurité des systèmes distribués, ce modèle a ses limitations et ses points faibles (voir [BM90]). Kerberos ne désigne pas un modèle de sécurité convenable aux systèmes ouverts. La plupart de ces limitations sont dues à l'environnement du projet Athena pour lequel Kerberos a été développé. Si les hypothèses de base de cet environnement ne sont pas vérifiées dans un autre environnement, il est alors nécessaire de modifier le protocole d'authentification.

Certains problèmes sont liés à l'architecture globale. En fait, ce projet considère le réseau comme une interconnexion d'un ensemble de principautés (*realms*), où chaque principauté a l'autonomie de l'administration de ces ressources. Le problème d'authentification croisée entre des entités appartenant à des principautés différentes n'est toujours pas traité efficacement. L'authentification de deux entités appartenant à deux principautés différentes nécessite un accord préalable entre les administrateurs de ces principautés. Un autre point faible est que les clés secrètes sont dérivées des mots de passe, ce qui pose le risque d'attaques dites par dictionnaire. Pour avoir une vue des limitations de Kerberos, voir [BM90]. Les différences entre les versions quatre et cinq sont détaillées dans [KNT90].

Dans ce chapitre nous essayons d'exposer les plus récents modèles dirigés vers la sécurité des systèmes ouverts. Les approches choisies sont celles les plus importantes et qui ont un trait en commun avec notre travail, à savoir, celles qui s'attaquent aux problèmes de gestion de clés publiques et que présentent un intérêt pour les environnements ouverts de nos jours.

La suite de ce chapitre détaille le cadre d'authentification X.509, le schéma de sécurité PEM proposé à l'origine pour le système de messagerie Internet, le modèle Chimaera, l'approche PGP et les spécifications PKCS. En particulier, l'approche X.509 est utilisée comme base de notre travail.

---

[12]L'authentification à clé secrète, telle que décrit dans la Partie I, Section 1.2.

5.1. Le Cadre d'Authentification X.509

La Recommandation X.509 du CCITT [X509] définit un cadre pour la prestation des services d'authentification à travers l'annuaire réparti X.500 [X500]. Cette recommandation propose un cadre pour la distribution de clés publiques et des protocoles d'authentification. Dans tous les cas, l'authentification repose, entre autres, sur la possession d'un nom d'annuaire unique pour chaque utilisateur.

L'architecture de gestion de clés spécifie la forme sous laquelle l'information d'authentification est conservée par l'annuaire, décrit la façon dont on peut obtenir cette information et établit les hypothèses faites au sujet des procédures permettant de constituer cette information.

La Recommandation X.509 propose deux types de protocoles d'authentification, l'authentification simple, basée sur l'utilisation des mots de passe pour la vérification de l'identité et l'authentification forte, qui fait intervenir des jetons établis au moyen de techniques cryptographiques. Puisque l'authentification simple n'offre qu'une protection limitée contre l'accès non autorisé, seulement l'authentification forte devra servir de base à la prestation de services sûrs.

Les utilisateurs du cadre X.509 comprennent l'annuaire lui-même ainsi que d'autres applications et services. L'annuaire peut contribuer utilement à répondre à leurs besoins en authentification et autres services de sécurité car c'est un emplacement adéquat à partir duquel les parties communicantes peuvent obtenir l'information d'authentification des uns et des autres, connaissance sur laquelle repose l'authentification. En raison du domaine étendu recouvert par l'annuaire, ce cadre d'authentification pourrait être utilisé par toute une gamme d'applications.

5.1.1. Le Cadre de Distribution de Clés

Les procédures d'authentification forte définies par la norme X.509 reposent sur des systèmes cryptographiques asymétriques (ou à clé publique). Ainsi, l'authentification repose sur la confirmation de l'identité par démonstration de la possession d'une clé privée.

En effet, pour qu'un système cryptographique soit utilisable dans ce cadre d'authentification, il doit avoir la propriété de la permutabilité, i.e., il doit permettre l'usage des deux clés de la paire pour le chiffrement; la clé privée est utilisée pour le déchiffrement si la clé publique à été utilisée pour le chiffrement et vice-versa.

La Recommandation X.509 définit un cadre de distribution de clés qui repose sur le fait que chaque utilisateur possède un nom spécifique (ou nom d'annuaire) unique dont l'attribution est à la charge d'une autorité de désignation. Dans ce cadre, chaque utilisateur est identifié par la possession d'une clé privée. Un deuxième utilisateur est capable de déterminer si un partenaire d'une communication est en possession de sa clé privée et peut utiliser ce renseignement pour confirmer que ce partenaire est bien l'utilisateur qu'il prétend être. Bien entendu, la validité de cette confirmation dépend de la mesure dans laquelle la clé privée demeure confidentielle au niveau de l'utilisateur.

Pour qu'un utilisateur puisse déterminer si un partenaire est en possession de sa clé privée, il doit lui-même avoir accès à la clé publique de cet utilisateur. Pour que les utilisateurs puissent obtenir

les clés publiques des autres utilisateurs en sécurité, la norme X.509 préconise l'utilisation de *Certificats*.

Un certificat associe l'identification d'une entité à sa clé publique de façon infalsifiable et pour cela il doit être émis par une entité de confiance. Pour concrétiser cela, la norme X.509 utilise le concept d'*Autorité de Certification* (CA). Celle-ci constitue alors l'entité de confiance dont le rôle est de certifier les clés publiques des utilisateurs en produisant des certificats.

Une autorité de certification produit les certificats en signant un recueil d'informations comprenant le nom spécifique d'utilisateur et sa clé publique. Plus précisément, le certificat d'un utilisateur A produit par autorité de certification CA prend la forme :

$$CA\langle\langle A\rangle\rangle = CA\{V, SN, AI, CA, UCA, A, UA, Ap, TA\}$$

Où, *V* indique la version du format de certificat, *SN* son numéro de série, *AI* est l'identificateur de l'algorithme utilisé pour signer le certificat, *Ap* est la clé publique de l'utilisateur A, et *TA* indique la période de validité du certificat et comprend deux dates, correspondant au début et la fin de cette validité.

Les éléments *UCA* et *UA* représentent une évolution du cadre X.509. Les nouvelles spécifications de la norme (X.500 1994) définissent une nouvelle version de certificat qui permet aux CA d'attribuer des identificateurs uniques à la CA (UCA) et à l'utilisateur (UA) de manière à permettre la réutilisation de noms d'annuaire sans que cela entraîne des conséquences sur l'authentification. Les CA peuvent alors s'en servir pour faire la différence entre des instances réutilisées d'un même nom d'annuaire.

Puisqu'un certificat est signé au moyen de la clé privée d'une autorité de certification, il possède les propriétés suivantes :

(1) tout utilisateur ayant accès à la clé publique de l'autorité de certification peut vérifier la clé publique qui est certifiée,

(2) aucune partie autre que l'autorité de certification ne peut modifier le certificat sans que cela ne soit détecté, i.e., les certificats sont infalsifiables.

Du fait que les certificats sont infalsifiables, on peut les publier, en les introduisant dans l'annuaire, sans que ce dernier ait à prendre des dispositions particulières pour assurer leur protection.

5.1.1.1. Récupération et Verification de Clés Publiques

Le certificat d'un utilisateur A peut être conservé dans son entrée d'annuaire. Ce certificat est produit par une autorité de certification de A, qui possède également une entrée d'annuaire et un certificat contenant sa clé publique. La clé publique de A peut ainsi être vérifiée par tout utilisateur qui a vérifié la clé publique de la CA de A. De la propriété (1) ci-dessus, il s'ensuit que la découverte de clés publiques est récursive.

En effet, la procédure d'obtention et validation d'une clé publique ne peut remplir son rôle que s'il existe une chaîne continue de points de confiance entre les utilisateurs qui souhaitent s'authentifier.

Si l'utilisateur A, qui essaie d'obtenir la clé publique d'un utilisateur B, a déjà obtenu la clé publique de sa CA, le processus est alors achevé. Afin de permettre l'obtention et la validation de clés publiques, l'entrée d'annuaire de chaque autorité de certification X peut contenir plusieurs certificats, qui sont de deux types. Le premier type correspond aux certificats de X produits par d'autres autorités de certification (*forward certificates*). Le deuxième type correspond aux certificats "inverses" produits par X, qui contiennent les clés publiques certifiées d'autres autorités de certification (*reverse certificates*).

Ces deux types de certificats permettront l'élaboration de chaînes de certificats contenant les clés publiques des points de confiance entre deux utilisateurs. La vérification récursive des certificats de cette chaîne assure la vérification de la clé d'un utilisateur final. La liste de certificats nécessaires pour permettre à un utilisateur de verifier la clé publique d'un autre utilisateur est appelée *Chemin de Certification*. Chaque élément de la liste est un certificat d'une CA permettant de vérifier le suivant. Ainsi, un chemin de certification de A vers B (indiqué par A ➞ B) débute avec un certificat produit par la CA de A, CA(A), à savoir CA(A) $\langle\langle X_1 \rangle\rangle$, où X_1 est l'autorité suivante dans le chemin. Il continue ensuite avec d'autres certificats du type X_i $\langle\langle X_{i+1} \rangle\rangle$, jusqu'au certificat de B, X_n <<B>>.

La méthode précise utilisée pour obtenir des chemins de certification peut varier et dépend de la distribution des CA. Une façon d'y parvenir plus facilement consiste à établir une distribution hiérarchique des CA. Cette hiérarchie peut coïncider, ou non, avec une partie ou la totalité de la hiérarchie du DIT. L'avantage d'une telle topologie est de pouvoir établir des chemins de certification entre deux utilisateurs sans information préalable. Pour cela, chaque CA peut stocker un certificat et un certificat inverse désigné comme correspondant à sa CA supérieure.

Pour constituer le chemin, on peut alors identifier un point de confiance commun dans la hiérarchie. Ce point commun doit être relié à chaque utilisateur par une chaîne continue de points de confiance. En adoptant une disposition hiérarchique des CA, un utilisateur A pourrait mémoriser les certificats et les certificats inverses de toutes les CA comprises entre lui-même et la racine de son arbre de certification. Lors de l'authentification d'un autre utilisateur B, A n'aurait besoin que d'obtenir les certificats entre le point commun de confiance et B.

La figure suivante représente un exemple fictif de distribution hiérarchique de CA. Dans un tel scénario, l'utilisateur A doit acquérir, de l'annuaire, les certificats suivants pour établir un itinéraire de certification vers B :

X<<W>>, W<<T>>, T<<Y>>, Y<<Z>>, Z<<B>>

Lorsque A a obtenu ces certificats, et admettant qu'il a obtenu la clé publique de sa CA, *Xp*, en sûreté, il peut vérifier en séquence l'itinéraire de certification pour vérifier le certificat de B, y compris sa clé publique *Bp* :

Bp = Xp . X<<W>> W<<T>> T<<Y>> Y<<Z>> Z<<B>>

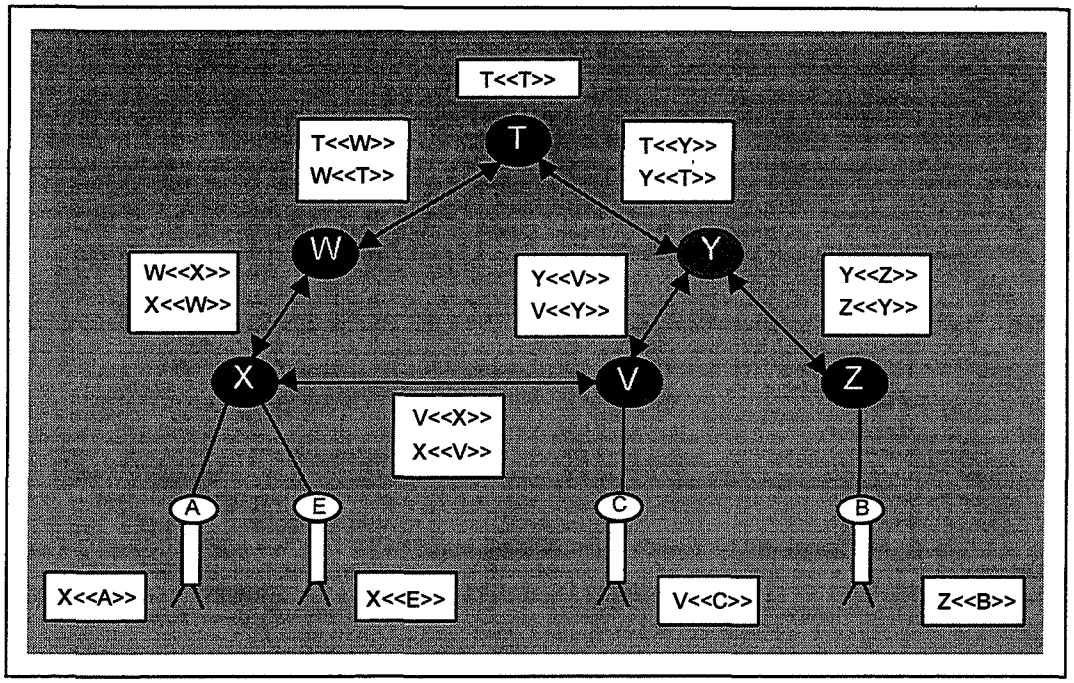

Figure 4. Exemple Fictif de Hiérarchie de CA

Le cadre d'authentification X.509 n'impose pas un mode spécifique de distribution des CA, la disposition hiérarchique est seulement suggérée. Comme exemple de modèle hiérarchique, nous pouvons citer celui conçu au sein du groupe de travail PEM [Ken93a] ou celui conçu au sein du groupe de travail PASSWORD [Roe92]. Les deux seront décrits dans la suite de ce travail.

Dans un cas général, les autorités de certification ne sont pas supposées avoir de relations hiérarchiques entre elles. Dans un tel contexte, plusieurs méthodes peuvent être utilisées pour établir un chemin de certification entre deux utilisateurs. La plus générale consiste à considérer les utilisateurs et les CA comme les noeuds et les certificats comme les arcs dans un graphe orienté. Dans ces conditions, en prenant la figure suivante comme exemple, A doit rechercher dans le graphe un chemin de U vers B, qui pourra être U<<V>>, V<<W>>, W<<B>>. Une fois que cet itinéraire a été établi, l'itinéraire inverse W<<V>>, V<<U>>, U<<A>> peut aussi être constitué, considérant que les certificats inverses existent.

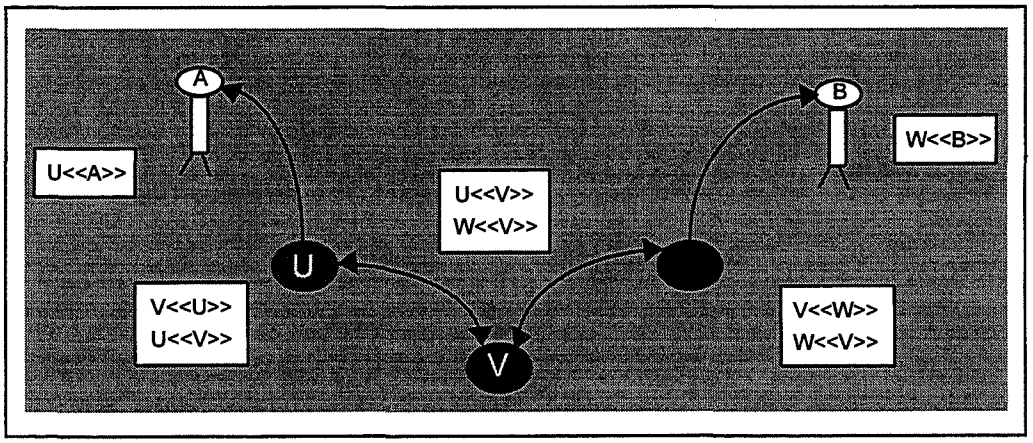

Figure 5. Exemple de Distribution non Hiérarchique de CA

Altarah a soulevé l'analogie entre la topologie des réseaux et la distribution generique des CA pour permettre la construction de chemins de certification à partir d'une disposition quelconque de ces autorités [Alt92]. Nous décrivons cette approche dans la Section 5.4.

5.1.1.2. Gestion des Certificats

Le cadre X.509 prescrit les lignes générales pour la gestion des certificats sans pourtant proposer une procédure spécifique. Chaque architecture particulière de gestion de clés doit spécifier les procédures qui satisfont à sa politique de sécurité. Une telle approche permet l'existence de CA de différents niveaux d'assurance et de fonctionnalité.

L'aspect concernant la génération de clés est considéré comme en dehors du cadre d'authentification. Cependant, il est capital pour la sécurité effective de ce cadre que la connaissance des clés privées reste limitée à l'utilisateur auquel ses clés appartiennent.

En ce qui concerne le rapport entre la production de clés et la certification des clés publiques, plusieurs approches peuvent être envisagées. Par exemple,

- L'utilisateur peut génèrer lui-même sa propre paire de clés.

  Cette méthode a l'avantage de ne jamais livrer une clé privée à une entité autre que son propriétaire. En contrepartie, une telle méthode exige une certaine capacité de calcul de la part du système de l'utilisateur.

- La CA ou une tierce partie peut génèrer la paire de clés de l'utilisateur.

  Dans ce cas, la CA (ou la tierce partie) doit livrer la clé privée à l'utilisateur de manière sûre, puis détruire résolument toute information relative à la création de la paire de clés, y compris les clés elles-mêmes.

  Un tel choix a l'avantage de ne pas nécessiter un transfert sécurisé de la clé publique à la CA pour la certification. En contrepartie, cette approche implique une augmentation de la confiance accordée à la CA qui aura le potentiel d'usurper l'identité des utilisateurs et d'accéder aux ses informations confidentielles.

La procédure de certification mérite également des mécanismes spéciaux. Un certificat lie la clé publique d'un utilisateur à son identification. Ainsi, il est important que les CA établissent les moyens de s'assurer de l'identité d'un utilisateur avant de créer un certificat à son intention. De même, les CA ne doivent pas livrer de certificats à deux utilisateurs qui possèdent le même nom. Par conséquent, et pour réduire les risques d'attaque à la clé privée de la CA, la production d'un certificat doit être effectuée comme une opération indépendante et ne doit pas être effectuée à travers des mécanismes du type question/réponse automatique. L'avantage d'un tel mécanisme de certification est que la clé privée de l'autorité de certification n'est jamais connue si ce n'est de la CA isolée et physiquement sûre, le secret de la clé privée ne peut être percé que par une attaque contre la CA elle-même.

Il convient de prendre des mesures de sécurité appropriées pour qu'un utilisateur soumettre sa clé publique à une CA en vue de sa certification. La sécurité de ce cadre serait mise en compromis si la CA livrait un certificat à un utilisateur ayant une clé publique qui aurait été falsifiée. Ainsi, les exigences de sécurité de la certification sont l'authencité et l'intégrité des clés publiques.

Par exemple, un intrus X qui réussi à certifier sa clé publique sous le nom d'un autre utilisateur Y, non seulement aurait accès à toute information confidentielle chiffrée au moyen de la clé publique contenue dans ce certificat. Cet intrus pourrait également se faire passer par Y dans tout mécanisme d'authentification basé sur ce certificat. Tout mécanisme de certification doit prendre en considération ces aspects.

Les certificats ont une durée de vie qui leur est associée et qu'à la fin de laquelle leur validité expirera. Pour assurer la continuité du service, la CA doit procéder de telle sorte que des certificats de remplacement soient disponibles à temps. La validité des certificats peut être prévue de manière que chacun devienne valide au moment où expire la validité du précédent. Les certificats périmés ne sont plus utilisables et peuvent être enlevés de l'annuaire. Il incombe à la CA de conserver les anciens certificats pendant un certain temps si un service de non-répudiation doit être fourni.

Les certificats peuvent être révoqués avant d'être périmés, par exemple si l'on suppose que la clé privée d'un utilisateur fait l'objet d'une compromission, ou si l'utilisateur change son nom d'annuaire (par la suite d'un changement d'employeur), ou si l'utilisateur n'est plus certifié par la CA.

L'annulation des certificats d'utilisateur ou de CA est annoncée à travers la publication de listes de révocation. Le cadre d'authentification X.509 préconise que chaque CA doit conserver deux listes de révocation datées. La première concerne les certificats qu'elle a émis et qui ont été annulés, la deuxième concerne les certificats annulés de CA.

Ces listes sont signées par la clé privée de la CA pour assurer leur intégrité et doivent être publiées même si elles sont vides, i.e., si aucun certificat n'a été révoqué. Ces listes sont conservées sous la forme d'attributs d'annuaire, à savoir, *CertificationRevocationList* et *AuthorityRevocationList*, dont la spécification ASN.1 est présentée dans l'Annexe A.

Finalement, le cadre X.509 fournit des moyens pour la représentation de données cryptographiques à travers la définition de macros ASN.1. Ces macros permettent d'identifier des algorithmes cryptographiques, mais aussi de représenter, de façon normalisée, des signatures et des portions d'information signées pour leur transfert sur le réseau. Une spécification complète des types et macros X.509 est présentée dans l'Annexe A.

5.1.2. Les Protocoles d'Authentification

Le cadre X.509 prescrit deux niveaux d'authentification : l'authentification simple qui utilise un mot de passe pour la vérification de l'identité et l'authentification forte qui fait intervenir des jetons établis au moyen de techniques cryptographiques. Puisque l'authentification simple n'offre qu'une protection limitée contre l'accès non autorisé, seule l'authentification forte devra servir de base à la prestation de services sécurisés.

5.1.2.1. L'Authentification Simple

L'authentification simple vise à fournir une autorisation locale reposant sur un nom spécifique d'utilisateur et un mot de passe. Le protocole d'authentification simple peut être réalisé de trois manières :

(a) transfert du nom spécifique de l'utilisateur et optionnellement du mot de passe en clair, donc sans aucune protection, au destinataire pour évaluation,

(b) transfert du nom spécifique de l'utilisateur, du mot de passe et d'un nombre aléatoire et/ou d'une indication horaire, le tout protégé par application d'une fonction de scellement à sens unique,

(c) transfert de l'information protégé décrite dans *(b)* ainsi qu'un nombre aléatoire et/ou une indication horaire, le tout protégé par application d'une fonction de scellement à sens unique.

La modalité d'emploi et de traitement du mot de passe dans un domaine donné doit être défini par accord bilatéral. Lorsque la modalité *(a)* est employée, un niveau de sécurité minimal est assuré pour empêcher un accès non autorisé. Ce type de mécanisme ne doit pas être considéré comme une base pour des services sécurisés. La protection du nom spécifique et du mot de passe de l'utilisateur assure une sécurité plus importante.

Le protocole d'authentification simple suit quatre étapes :

(1) L'utilisateur A (le postulant) envoie son nom spécifique et son mot de passe à un utilisateur destinataire B (le vérificateur),

(2) B envoie le nom et le mot de passe de A à l'annuaire, pour comparaison avec celui qui détient l'attribut "mot de passe" de l'entrée d'annuaire concernant A,

(3) l'annuaire confirme ou dément à B que ces accréditations sont valides,

(4) B communique le succès ou l'échec de l'authentification à l'utilisateur A.

Deux méthodes peuvent être utilisées pour protéger l'information d'identification, comme le montre la figure 6. Dans la figure, *f1* et *f2* sont des fonctions à sens unique, identiques ou différentes. Les indications horaires (*t1A* et *t2A*) et les nombres aléatoires (*q1A* et *q2A*) sont facultatifs et dépendent d'accords bilatéraux.

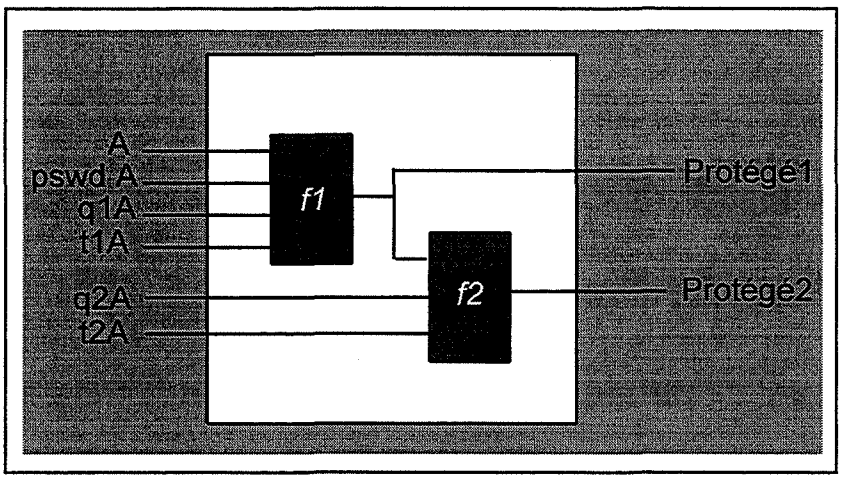

Figure 6. Authentification Simple Protégée

Le protocole d'authentification simple protégée comporte les étapes suivantes :

(1) L'utilisateur A envoie à l'utilisateur B son information d'identification protégée (*authentificateur1*). La protection est assurée par application de la fonction à sens unique *f1* sous la forme :

protégé1 = f1 (t1A, q1A, A, pswdA)

Où, *pswdA* représente le mot de passe de A et l'indication horaire *t1A* et le numéro aléatoire *q1A* ont pour objet de réduire les répétitions et de cacher le mot de passe. L'information transmise à B prend la forme :

authentificateur1 = t1A, q1A, A, protégé1

(2) L'utilisateur B vérifie l'information protégée envoyée par A en produisant une copie locale de *protégé1* en utilisant les informations envoyées en clair dans *authentificateur1* et une copie locale du mot de passe de A. L'authentification est considérée comme réussie si la valeur calculée est égale à la valeur envoyée par A.

Cette procédure peut être modifiée afin d'assurer une plus grande protection du mot de passe. Cette protection supplémentaire est obtenue en appliquant une autre fonction de scellement à sens unique *f2* à la portion de données *protégé1* (une autre indication horaire et/ou numéro aléatoire en peuvent être ajoutés pour "brouiller les pistes"). La nouvelle information d'identification protégée prend la forme :

protégé2 = f2 (t2A, q2A, protégé1)

L'information transmise à B prend la forme :

authentificateur2 = t1A, t2A, q1A, q2A, A, protégé2

A la réception de *authentificateur2*, B procède à la vérification telle que présentée en (2) ci-dessus.

Les protocoles d'authentification sont présentés en fonction de A et B, mais par exemple, dans le cas de l'authentification appliquée à l'annuaire, A et B seraient, respectivement, un DUA et un DSA ou deux DSA.

5.1.2.2. L'Authentification Forte

La recommandation X.509 définit trois protocoles d'authentification forte, basés sur des techniques cryptographiques asymétriques, où la confirmation de l'identité est déterminée par démonstration de la possession d'un secret, à savoir, la clé privée.

Les trois procédures font intervenir différents nombres d'échanges d'information d'authentification et, en conséquence, offrent différents niveaux d'assurance à leurs participants. Elles ont été conçues de manière à satisfaire à toute une gamme d'applications. Le choix de la procédure est laissé à chaque application qui doit déterminer celle qui satisfait à sa politique de sécurité.

Dans chaque cas d'authentification forte, l'utilisateur A doit obtenir et valider la clé publique de B et le chemin de certification de B vers A (représenté par B ➞ A) avant tout échange d'information. Ces protocoles sont décrits ci-dessous.

(a) ***Authentification à sens unique*** : Cette procédure fait intervenir un transfert unique de l'utilisateur A vers l'utilisateur B sous la forme suivante :

(1) L'utilisateur A génère un nombre aléatoire non-récurrent *rA* (qui est utilisé pour détecter des attaques par re-jeu et pour empêcher la falsification) et envoie le message suivant à B :

B ➞ A, A{ tA, rA, B, sgnData, Bp[encData] }

où, *tA* comprend une ou deux indications horaires, la date-heure de production du jeton (facultatif) et sa date d'expiration ; *sgnData* est facultatif et contient des données signées par A pour assurer leur origine et leur intégrité ; *Bp[encData]* est facultatif et contient des données chiffrées avec la clé publique de B pour assurer la confidentialité. Ces données peuvent être, par exemple, une clé de session à utiliser pour chiffrer la suite de la communication.

(2) L'utilisateur B obtient la clé publique *Ap* de A à partir de B ➞ A et vérifie la signature du jeton. Ensuite, il vérifie si B est bien le destinataire prévu, si l'indication horaire est à jour, et optionnellement, si le nombre aléatoire n'a pas été réutilisé pendant la période de validité spécifiée par *tA*.

(b) ***Authentification à double sens*** : Cette procédure fait intervenir, en plus de l'authentification à sens unique, une réponse de B vers A. Les deux premières étapes sont identiques à celles spécifiées ci-dessus. Ensuite, B envoie le message suivant à A :

B{ tB, rB, A, rA, sgnData, Ap[encData] }

La signification des champs est la même que celle de la procédure d'authentification à sens unique. A la réception, A procède à la vérification des données reçues comme décrit en (2) ci-dessus.

(c) ***Authentification à trois échanges*** : Cette procédure comprend, en plus de l'authentification à double sens, un transfert supplémentaire de A vers B. Les quatre premières étapes sont identiques à celles spécifiées ci-dessus. Ensuite, A vérifie que *rA* reçu est identique au *rA* émis et envoie à B le jeton suivant :

A{ rB, B }

A la réception, B vérifie la signature du jeton, et par conséquent l'origine et intégrité de l'information signée. Puis, il vérifie que *rB* reçu est identique au *rB* émis par B.

5.1.3. Lacunes du Cadre d'Authentification X.509

Le cadre d'authentification défini par la Recommandation X.509 a apporté une grande contribution à l'implantation de mécanismes de sécurité dans les systèmes ouverts. Cependant ce

cadre, tel qu'il se présente aujourd'hui, ne viabilise pas la mise en place d'une infrastructure globale d'authentification, ceci dû aux raisons suivantes :

D'abord, afin de donner une plus grande liberté aux applications, la recommandation laisse des points importants sans traitement spécifique, comme par exemple la procédure de construction de chemins de certification et la définition des politiques de certification adoptées par les CA et leur juridiction dans l'espace de noms.

De plus, ce cadre impose pratiquement l'utilisation de l'algorithme RSA dû à la propriété de la permutabilité textuellement imposée aux algorithmes utilisables dans ce cadre. En outre, le format de certificats ne permet pas que deux paires de clés différentes, une pour la confidentialité et une autre pour l'authentification, soient utilisées. D'autres déficiences au niveau des formats de certificat et de listes sont traitées dans la Partie III, Section 10.1.3 de ce travail.

La Recommandation X.509 ne définit pas seulement le format de certificats, mais tout un cadre d'authentification dans lequel les certificats jouent un rôle essentiel. Ce cadre est très général et impose très peu de restrictions sur le système de certification résultant. Cela peut entraîner l'élaboration de tout un spectre de systèmes de certification. A une extrémité de ce spectre, il est possible d'avoir des systèmes dans lesquels il n'existe pas un modèle de sémantiques globales sur le plan de la certification, i.e., la sémantique et la confiance sur la certification sont définies localement par chaque utilisateur.

A l'autre extrémité de ce spectre, il se trouve des systèmes dans lesquels il existe un ensemble de politiques globales, largement disponibles et définies à travers l'accord dans une communauté d'utilisateurs. Le cadre X.509 sert de support à ces deux extrémités.

En raison du caractère général du cadre de distribution de clés, la tendance est que chaque communauté ou groupe d'utilisateur élabore un système de certification convenant le mieux à l'organisation de son environnement.

Au niveau des protocoles d'authentification, des lacunes ont été déjà publiées par Mitchell [IM90] et prouvées formellement par Toussaint dans [Tou93]. Dans sa publication, Mitchell montre une possibilité d'attaque sur l'échange de données secrètes par les protocoles X.509 et une défaillance au niveau du protocole à trois échanges. Cette dernière a été corrigée dans la nouvelle version de la norme X.509, tandis que la première reste inchangée.

5.2. PEM

Privacy **E**nhanced **M**ail (PEM) [Lin93, Ken93a, Bal93, Kal93] constitue l'approche développée pour répondre aux besoins de sécurité du système de courrier électronique du monde Internet. Les travaux sur PEM ont débuté en 1985 comme l'une des activités du groupe de travail PSRG (**P**rivacy and **S**ecurity **R**esearch **G**roup), sous l'égide de l'IAB (**I**nternet **A**rchitecture **B**oard).

Cette approche est définie à travers une série de documents, en forme de RFC (**R**equest **f**or **C**omments), qui spécifient des mécanismes de sécurité pour la messagerie électronique tout en préservant la structure de messagerie et de communication existante. Ainsi, PEM désigne une architecture qui permet aux utilisateurs finals de renforcer la sécurité de leur courrier

électronique, sans que les systèmes intermédiaires aient à fournir la moindre mesure ou protection.

L'existence de divers agents et protocoles de transfert de message suggère que, pour être largement utilisé, un protocole pour le renforcement de la sécurité du courrier électronique ne doit pas demander la reconstruction des protocoles et des systèmes existants. L'effort de coordination d'un changement global de protocoles à travers l'Internet serait très important, voire impraticable.

Ainsi, PEM ne fourni pas un renforcement de la sécurité de la structure de messagerie elle-même, mais ce service est fourni au niveau du document. La sécurité de la messagerie elle-même n'est pas fournie car cela demande des changements au niveau du système de transfert de message. Dans PEM, tant le texte chiffré que l'information de contrôle cryptographique, sont représentés en format textuel. Ceci permet aux messages PEM d'être véhiculés par n'importe quel système de messagerie.

Le RFC 1421 [Lin93] définit les procédures de chiffrement et d'authentification nécessaires à la fourniture du service PEM. Ces procédures sont désignées pour être compatibles avec toute une gamme d'approches d'architectures de gestion de clé. Ces procédures font usage d'algorithmes symétriques et asymétriques. Le RFC 1423 [Bal93] spécifie les algorithmes, les modalités d'utilisation et les identificateurs associés à cette approche.

5.2.1. Les Services PEM

L'approche PEM a été conçue pour offrir les services suivants :

* confidentialité de messages,
* intégrité de messages,
* authentification de l'origine de messages,
* non-répudiation de l'origine de message.

Tous les messages PEM incorporent les services d'intégrité et d'authentification, tandis que le service de confidentialité est optionnel. Le service de non-repudiation d'origine ne peut être fourni si des mécanismes d'authentification à clé publique sont utilisés.

En principe, PEM est orienté vers l'utilisation dans l'environnement de messagerie classique Internet, caractérisé par deux standards Internet, à savoir, le RFC-822 [Cro82] et SMTP [Pos82]. Le premier définit la syntaxe des messages et la sémantique des en-têtes de messages. Le dernier définit un protocole de transfert de messages. Des travaux sont menés actuellement, de façon à pouvoir déployer PEM dans l'environnement MIME (**M**ultipurpose **I**nternet **M**ail **E**xtensions) [BF92]. Une telle extension conjuguera la sécurité de la messagerie aux capacités du multimédia.

Bien que PEM soit conçu pour être utilisé sur les protocoles de messagerie Internet, cette approche peut être employée dans une gamme variée d'environnements de messagerie. Par exemple, le NIST Open System Implementation Workshop Security Working Group a défini une partie de corps X.400 pour transférer des messages PEM. Ceci permet que des passerelles SMTP-X400 puissent livrer des messages PEM dans un message X.400.

Il doit être noté que l'opération inverse n'est pas si triviale, car les éléments qui implémentent la sécurité dans X.400 sont placés dans l'enveloppe du message. Le groupe de travail Password est en train d'étudier ce sujet pour concevoir une passerelle X400(88)-PEM.

Un schéma de codage spécifique à PEM est aussi utilisé pour maximiser la possibilité que des messages PEM puissent transiter par des passerelles entre la messagerie Internet et d'autres systèmes de courrier électronique, étant donné que beaucoup d'entre elles ne font pas la conversion des formats de façon transparente au niveau des contenus de messages.

5.2.2. Les Clés de Chiffrement

PEM utilise deux types de clés pour la transmission d'un message sécurisé, à savoir, les **DEK** (**D**ata **E**ncrypting **K**ey) et les **IK** (**I**nterchange **K**ey). Les DEK sont utilisées, principalement, pour le chiffrement du texte des messages confidentiels. Les IK sont utilisées pour chiffrer les DEK, afin d'assurer la sécurité de leur transmission sur le réseau.

Les clés du type DEK sont générées individuellement pour chaque message PEM. Aucune distribution préalable de DEK est nécessaire au service PEM. Dans les systèmes conventionnels cela était assuré par les serveurs de clés symétriques. Le besoin de tels serveurs est éliminé grâce au mécanisme de distribution de clés asymétriques encapsulées dans des certificats.

Les DEK peuvent aussi être utilisées, avec certains choix parmi un ensemble d'algorithmes alternatifs, au calcul des dits MIC (**M**essage **I**ntegrity **C**heck). Les DEK peuvent aussi être utilisées pour chiffrer les MIC (déjà signées au moyen d'une clé asymétrique), lorsque leur confidentialité doit être assurée.

Chaque message PEM inclut une représentation de la DEK (ou des DEK) utilisée pour le chiffrement du message et/ou pour le calcul du MIC, chiffrée avec une IK différente par chaque destinataire du message. Cette représentation est associée aux champs "*Originator-ID*" et "*Recipient-ID*" de l'en-tête, pour permettre à chaque destinataire d'identifier la IK utilisée pour le chiffrement des DEK (et/ou du MIC). Avec la IK, chaque destinataire peut alors déchiffrer la DEK et ensuite l'utiliser pour le déchiffrement du message (et/ou pour la validation du MIC).

Une seule clé IK sera utilisée entre un émetteur et un destinataire pour une période de temps spécifiant la validité de cette clé. La définition d'une IK et sa période de validité diffèrent selon le type de système cryptographique utilisé pour le chiffrement de la DEK. Si un système cryptographique symétrique est utilisé, la IK est une clé symétrique partagée entre l'expéditeur et le destinataire. Dans ce cas, la même IK est utilisée pour chiffrer la DEK et le MIC.

Si un système cryptographique asymétrique est utilisé, la IK employée pour le chiffrement de la DEK est la clé publique (contenue dans le certificat) du destinataire et la IK utilisée pour le chiffrement du MIC est la clé privée de l'expéditeur. Ceci constitue l'approche la plus courante.

5.2.3. Les Types de Messages

Dans la messagerie Internet, l'en-tête du message est séparé du contenu par une ligne blanche. Dans un message traité par des procédures PEM, une marque explicite est insérée après ce séparateur pour identifier le début d'un message PEM :

---- BEGIN PRIVACY-ENHANCED MESSAGE ----

Ensuite, il vient l'ensemble d'informations qui représente l'en-tête PEM. Ces informations sont utilisées par chaque destinataire du message pour vérifier l'intégrité et l'origine du message, et aussi pour déchiffrer un message confidentiel.

Apres l'en-tête du message, se trouve le contenu proprement dit du message PEM, et finalement une marque d'identification de fin :

---- END PRIVACY-ENHANCED MESSAGE ----

PEM définit trois types principaux de message pour fournir différentes combinaisons de services, à savoir, MIC-CLEAR, MIC-ONLY et ENCRYPTED. Un message du type MIC-CLEAR et MIC-ONLY assure les services intégrité et d'authentification de l'expéditeur du message à travers la présence d'un MIC (Message Integrity Code). La différence entre ces deux types de message concerne uniquement le choix de representation des données sous une forme textuelle, i.e., le codage particulier n'est pas appliqué aux messages du type MIC-CLEAR.

L'avantage du message MIC-CLEAR est qu'il peut être envoyé à des utilisateurs qui ne disposent pas d'une implémentation PEM. Par exemple, un tel message peut être envoyé à une liste de distribution qui contient des utilisateurs disposant ou non de PEM. Tous les utilisateurs pourront lire le texte du message, mais seuls ceux qui disposent des procédures PEM pourront vérifier l'intégrité et l'origine du message.

En contrepartie, l'application du codage particulier, utilisant un code de 7 bits, assure la diffusion d'un message PEM à travers diverses passerelles et tous les types de modem sans que les conversions invalident la vérification de l'intégrité et de l'origine du message.

Le message du type ENCRYPTED assure, en plus des services d'authentification et d'integrité, le service de confidentialité à travers le chiffrement. Le champ "*Proc-type*" de l'en-tête indique le type de message PEM.

Le traitement de messages PEM comprend deux transformations principales sur le plan de la sécurité, à savoir, le calcul du MIC et le chiffrement (optionnel) du message. Ces deux étapes sont précédées par une étape canonique propre aux procédures SMTP.

PEM considère le choix de l'algorithme de calcul du MIC comme un paramètre qui peut varier parmi des communautés d'utilisateurs et qui peut changer avec l'évolution de la technologie. L'algorithme utilisé pour calculer le MIC est indiqué dans le champ "*MIC-Info*" de l'en-tête d'un message PEM. Les procédures PEM utilisent des fonctions de scellement à sens unique considérées de haute assurance pour le calcul des MIC, par exemple, l'algorithme MD2 [Kal92].

Pour assurer, à la fois, les services d'intégrité, d'authentification et non-répudiation d'origine, le MIC doit être protégé de façon à être lié à l'identité de l'expéditeur. Si un système cryptographique asymétrique est utilisé, le MIC est protégé par le composant privé de la paire de

clés de l'expéditeur. Ceci résulte en une signature numérique du message qui peut être vérifiée par n'importe quel utilisateur.

Pour qu'un destinataire puisse établir facilement la liaison entre la valeur du MIC et l'identité de l'expéditeur, le composant "*Originator-Certificate*" de l'en-tête des messages PEM permet d'identifier l'expéditeur par inclusion de son certificat. La clé publique encapsulée dans ce certificat permettra aux destinataires de vérifier le MIC et de prouver l'origine du message. PEM permet également l'inclusion de l'itinéraire de certification qui permet de valider la clé publique de l'expéditeur à travers le champ "*Issuer-Certificate*" de l'en-tête.

Dans les cas où la confidentialité du message doit être assurée, PEM chiffre également la valeur signée du MIC (en utilisant la même clé employée pour le chiffrement du contenu du message). Cela est fait pour protéger le message des attaques par déduction (guess attacks). Bien qu'il ne soit pas possible de déterminer le contenu du message à partir de la valeur du MIC, un intrus pourrait faire des estimations à propos du contenu du message (chiffré) et les tester contre la valeur du MIC (déchiffrée avec la clé publique de l'expéditeur).

L'algorithme de chiffrement employé et sa modalité d'utilisation sont indiqués dans le champ "*DEK-Info*" de l'en-tête PEM du message. Si l'algorithme en question requiert des paramètres, cela est également indiqué dans ce paramètre.

Un message confidentiel PEM est chiffré exactement une fois, indépendamment du nombre de destinataires. Seulement une copie de ce message chiffré est envoyée. En utilisant un système cryptographique asymétrique pour la distribution de clés, une copie de la DEK est chiffrée utilisant la clé publique de chaque destinataire. Si un système cryptographique symétrique est utilisé, la clé de chiffrement de la DEK est définie par accord bilatéral avec chaque destinataire. A travers un tel mécanisme, chaque copie de la DEK est protégée et déchiffrable par exactement un destinataire.

Chaque copie protégée de la DEK est placée dans le champ "*Key-Info*" de l'en-tête PEM, suivie de l'algorithme utilisé pour le chiffrement. Chaque champ "*Key-Info*" est précédé par le champ "*Recipient-ID-Asymetric*" qui contient le nom d'annuaire de son autorité de certification et le numéro de série du certificat du destinataire en question. Cette combinaison identifie uniquement ce destinataire et lui permet de déterminer la clé nécessaire pour déchiffrer la DEK.

5.2.4. La Gestion de Clés dans PEM

L'approche PEM repose sur une architecture de gestion de clés inspirée du cadre proposé par la Recommandation X.509. Le RFC 1422 [Ken93a] spécifie les mécanismes de gestion de clé basés sur de certificats X.509. Le RFC 1424 [Kal93] fournit les détails sur les formats de message (électronique ou non) et les procédures pour l'établissement de l'infrastructure de gestion de clés nécessaire au support de ces services PEM. Cela inclut les formats de requêtes pour l'émission, la révocation d'un certificat et la récupération des listes de révocation.

Comme cela a été déjà mentionné, les spécifications PEM encouragent l'utilisation de la cryptographie à clé publique pour l'authentification de l'expéditeur, l'intégrité de messages et pour la distribution de clés de chiffrement symétriques. La façon d'aborder la certification dans l'environnent PEM reflète la topologie de l'environnement dans lequel il s'insère : l'Internet

Society. Cette approche a été conçue avec l'intention d'accommoder toute une gamme de politiques de certification sous l'égide d'une politique commune et globale. La politique globale a comme objectif d'imposer certaines restrictions sur le système de façon à permettre :

- ✓ l'association d'une sémantique d'authentification compatible avec la validation automatique de certificats,
- ✓ permettre l'élaboration de procédures uniformes de validation de certificats,
- ✓ et assurer que les utilisateurs pourront facilement interpréter les résultats de ces procédures.

L'architecture PEM propose un système de certification hiérarchique, sous forme d'arbre dont la racine est unique et représentée par l'IPRA (**I**nternet **P**olicy **R**egistration **A**uthority). L'IPRA est une autorité de certification chargée d'établir la politique générale et commune, applicable à tous les certificats sous cette hiérarchie. Cette entité fournit un point de référence à partir duquel tous les certificats de la hiérarchie de certification peuvent être validés. L'IPRA certifie les entités appartenant au deuxième tiers de la hiérarchie, à savoir, les PCA (**P**olicy **CA**).

Pour accepter une PCA dans le système de certification, l'IPRA entre en accord avec la PCA en exigeant que celle-ci obéisse à la politique générale du système et qu'elle publie sa politique spécifique dans un format commun. Cette politique spécifique est soumise à l'IPRA pour approbation avant l'acceptation de la PCA dans le système.

Le système de certification PEM associe une sémantique spéciale au second tiers de la hiérarchie. Au-dessous de chaque PCA, se trouvent les CA ordinaires représentant des organisations, des régions géopolitiques, etc. En vue de sa certification par une PCA, la CA doit se soumettre à la politique générale du système et à la politique spécifique de la PCA en question.

Les CA sont chargées de certifier, en principe, les utilisateurs finaux, mais elles peuvent certifier également d'autres CA subordonnées qui représentent des unités d'organisation, des rôles organisationnels ou des régions géopolitiques. Trois types de CA ont été identifiés dans le système de certification PEM:

- Les ***CA Organisationnelles*** - Les CA de ce type sont chargées de certifier des utilisateurs affiliés à une organisation.
- Les ***CA Résidentielles*** - Les CA de ce type sont désignées pour certifier des utilisateurs ayant besoin de s'authentifier en tant qu'individu non affilié à une organisation, ou indépendamment de cette affiliation.
- Les ***CA "Persona"*** - Ce type de CA répond au besoin des utilisateurs qui ne souhaitent pas révéler leur identité réelle. Les certificats persona permettront une utilisation "anonyme" de services de sécurité tout en préservant la continuité de l'authentification. Ainsi, bien que l'on ne connaisse pas la véritable identité du détenteur d'un certificat persona, on serait capable de déterminer si une série de messages émis sous cette identité appartient vraiment à un même utilisateur.

Les certificats d'utilisateur représentent les feuilles de l'arbre de certification. Dans le sens du système de certification PEM, ces utilisateurs peuvent être des individus, des organisations (ou unités d'organisation), des rôles organisationnels, des listes de distribution. La figure 7 illustre le schéma de certification PEM.

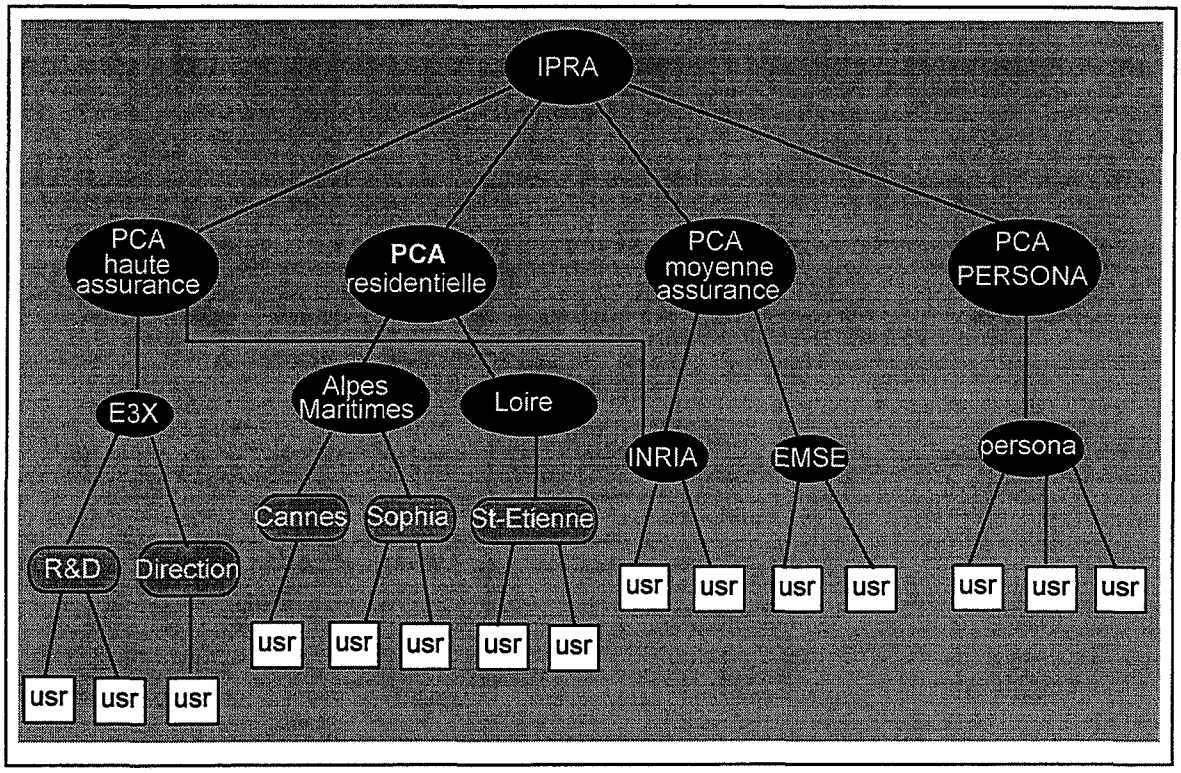

Figure 7. Exemple Fictif du Modèle de Certification PEM

Le modèle de certification PEM constitue une hiérarchie de certification isomorphe à la hiérarchie de nommage X.500, à deux exceptions près :

(1) l'IPRA forme la racine de l'arbre tandis que la racine du DIT n'est pas instantiée comme un noeud,

(2) les PCA forment les "racines" de sous-arbres dans la mesure où ces entités représentent des politiques différentes de certification.

D'autres aspects de la politique générale de l'IPRA incluent les points suivants :

- Typiquement, une CA sera certifiée par une seule PCA. Cependant, il est valide qu'une CA soit certifiée par plusieurs PCA. Sur le plan de la certification, cela implique que cette CA est capable d'émettre des certificats sous plusieurs politiques différentes.

  Dans un tel cas, pour chaque PCA sous laquelle la CA se fait certifier, cette dernière doit présenter une clé publique différente. Cette restriction a comme objectif de permettre l'identification des différentes politiques sous lesquelles cette CA émet des certificats, pourvu que pour chaque politique différente, la CA signe des certificats avec des clés privées différentes.

- Tout certificat PEM doit être tel que le nom de l'entité certifiée est subordonné au nom de l'entité certificatrice, sauf pour les certificats émis par l'IPRA, par les PCA ou par les CA Persona.

Au démarrage, l'approche PEM n'était pas envisagée comme une technologie à long terme pour la sécurisation de messagerie électronique, mais comme une solution intérimaire avant la

disponibilité de la messagerie sécurisée OSI et des services d'annuaire X.500 [Ken93b]. Néanmoins, étant donnée l'acceptation de l'approche PEM, et selon la viabilité et la disponibilité des systèmes X.400-88 et X.500 sur le marché, PEM peut devenir une solution à long terme pour la distribution de clés publiques non seulement pour la sécurisation de la messagerie mais aussi pour tous les services de haut niveau de l'Internet. Par ailleurs, ce système de certification n'est déjà plus considéré comme spécifique à PEM, mais comme "le système de certification Internet". Son utilisation dans d'autres contextes est également envisagée.

5.3. PGP

PGP est l'abréviation de ***Pretty Good Privacy*** [Zim93]. PGP constitue un modèle de sécurité basé sur des techniques cryptographiques asymétriques conçu pour fournir les services d'intégrité, d'authentification, de non-repudiation d'origine, et de confidentialité, applicables à l'échange de fichiers et de messages.

Contrairement à PEM, PGP n'adhère pas aux principes introduits par le cadre d'authentification X.509. PGP constitue une approche plutôt anarchiste qui s'oriente vers les utilisateurs qui ne souhaitent pas intégrer un modèle de confiance publique.

PGP utilise le système cryptographique asymétrique RSA [RSA78] pour la distribution de clés publiques et l'établissement de communications sûres entre utilisateurs, i.e., l'intégrité, l'authentification, la non-repudiation d'origine et la confidentialité applicable à l'échange de fichiers et de messages.

Des systèmes cryptographiques symétriques sont utilisés uniquement pour garantir la confidentialité des clés privées de l'utilisateur et des documents conservés dans l'environnement local de l'utilisateur. A l'heure actuelle, le système cryptographique adopté par PGP est l'algorithme IDEA [LM90].

Les clés publiques et privées sont encapsulées dans des certificats dont le format est propre à PGP. Ces certificats contiennent l'identification de l'utilisateur, un identificateur pour la clé, une estampille de temps indiquant la date de génération, et la clé, le tout accompagné d'une signature numérique.

Dans PGP, les clés publiques et privées sont conservées dans des fichiers dans l'environnement local de l'utilisateur. Ces fichiers sont dits "anneaux de clés" (*key rings*). Chaque utilisateur maintient sa paire d'anneaux de clés, l'un contenant des clés publiques et l'autre des clés privées.

Pour garantir l'intégrité et l'authenticité des données, PGP utilise la notion de "signature de certification" (*certifying signatures*). A travers ce mecanisme les données sont encapsulées dans une enveloppe contenant sa signature, l'identificateur de la clé nécessaire à sa vérification, l'identification de son détenteur et un estampillage de temps indiquant la date de création de la signature. Cela est appliqué tant à des documents qu'aux clés publiques.

En ce qui concerne la confidentialité, les documents chiffrés sont préfixés par l'identificateur de la clé publique utilisée pour le chiffrement.

5.3.1. Gestion de Clés

La gestion de clés telle qu'abordée par PGP reflète les relations sociales naturelles entre les individus. Cette approche permet à chacun de choisir à qui confier la tâche de distribution des clés. Dans cette section nous présentons les méthodes de distribution et de vérification de clés publiques dans PGP.

5.3.1.1. Distribution de Clés

Dans PGP, la notion de confiance est une question locale à chaque utilisateur. Un utilisateur X peut obtenir la clé publique d'un autre utilisateur Y au moyen de deux méthodes :

(1) directement de Y,

(2) travers un autre utilisateur W auquel X et Y font confiance.

Dans l'alternative (2), W certifierait la clé publique de Y, attestant l'intégrité et l'origine de cette clé. Cela demande que X soit en possession de la clé publique de W pour pouvoir valider la signature du certificat. Ainsi, dans la terminologie PGP, W assure le rôle d'*introducteur* entre X et Y.

Contrairement au cadre d'authentification X.509, il n'existe pas le concept d'Autorité de Certification dans l'approche PGP. Il constitue un système décentralisé et interpersonnel. En tant que décentralisé, ce système permet à tout utilisateur d'agir comme introducteur d'autres utilisateurs dont il a obtenu la clé publique d'une manière sûre.

Un utilisateur W peut certifier des clés d'autres utilisateurs, obtenues par des moyens sûrs et les stocker dans des BBS (**B**ulletin **B**oard **S**ystem). Postérieurement, d'autres utilisateurs qui font confiance à W et le considèrent comme introducteur peuvent récupérer ces clés publiques à partir d'un BBS et les conserver dans leur environnement local.

De même, chaque utilisateur X peut publier, dans plusieurs BBS, sa clé publique accompagnée de signatures de certification de plusieurs introducteurs. Ainsi, quelqu'un d'autre qui considère au moins un de ces utilisateurs comme introducteur peut récupérer la clé de X lorsqu'il en aura besoin.

Figure 8. Certification de Clés Publiques dans PGP

Les clés publiques de l'utilisateur local sont conservées dans son anneau de clés publiques avec des clés publiques appartenant à d'autres utilisateurs. Pour identifier chacune de ces clés, un identificateur est formé par les 64 bits de poids faible de la clé. Cette methode permet l'identification globale des clés publiques dans le système. Le destinataire d'un document signé peut utiliser cet identificateur pour récupérer, dans son anneau, la clé publique nécessaire à la validation de la signature. Ce même identificateur est attaché au composant privé correspondant dans l'anneau de clés privées. Ce mecanisme permet d'établir la correspondance entre les deux clés d'une paire dans les anneaux de clés d'un utilisateur.

L'utilisateur peut également associer des mots clés à chaque clé dans l'anneau. Un mot clé peut être une chaîne de caractères quelconque capable d'identifier et de distinguer un utilisateur, par exemple, un numéro de téléphone ou une adresse électronique. Cela permet une identification conviviale des clés publiques.

L'approche PGP exige une certaine capacité de jugement de la part de tous ses utilisateurs. Ceci dit, tout utilisateur, lorsqu'il récupère la clé publique d'un autre utilisateur, doit être sûr que cette clé n'a pas été modifiée et qu'elle appartient vraiment à l'utilisateur qui prétend être son détenteur.

Les mêmes précautions que ci-dessus doivent être prises lorsque l'on demande à un utilisateur de signer la clé publique de quelqu'un d'autre. Lorsque X signe la clé publique de Y, X se porte garant que cette clé appartient vraiment à Y. Ainsi, d'autres utilisateurs qui font confiance à X accepteront la clé publique de Y parce qu'elle porte la signature de X.

5.3.1.2. Vérification de Clés Publiques

La façon d'aborder la validation de clés publiques dans PGP est basée sur les principes suivants :

(1) la clé de l'utilisateur local demeure "axiomatiquement" valide dans le modèle de certification PGP,

(2) les clés certifiées par un introducteur sont considérées valides,

(3) les clés publiques des introducteurs doivent être certifiées par l'utilisateur local lui-même ou par d'autres introducteurs auxquels cet utilisateur fait confiance.

Ainsi, PGP considère chaque utilisateur comme le sommet de sa propre pyramide privée de certification. La figure 9 illustre un tel scénario.

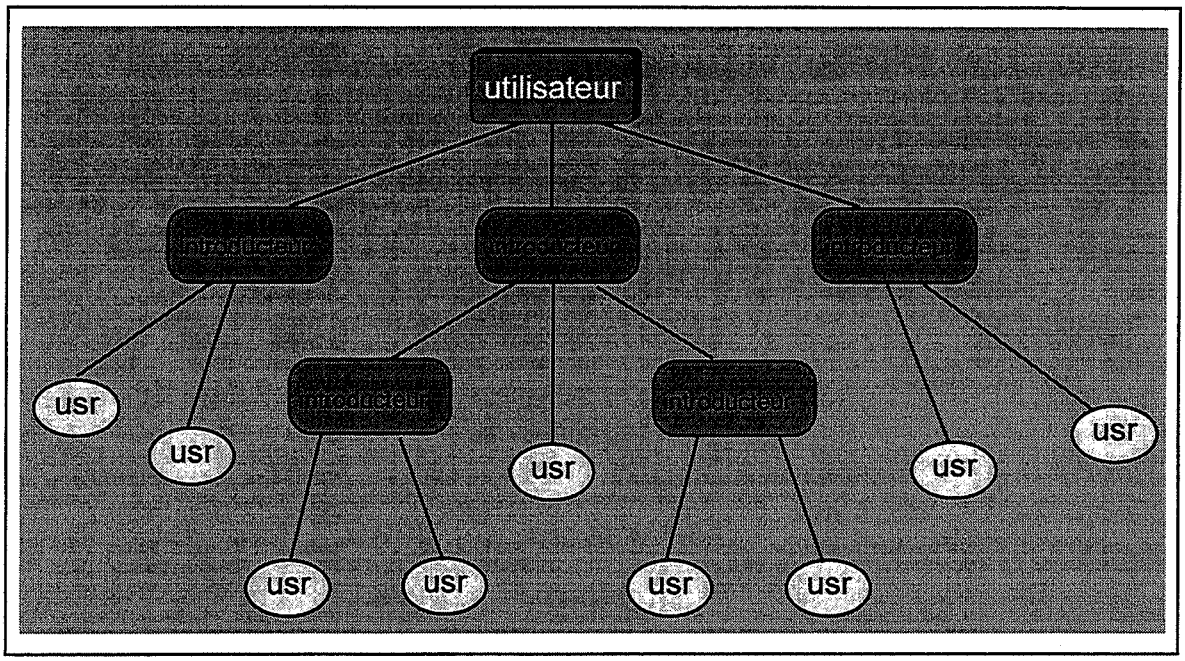

Figure 9. Modèle de Certification PGP

L'approche PGP offre aussi à l'utilisateur la possibilité de spécifier des variations de degré de confiance attribuée à chaque introducteur. Il est possible de désigner une personne comme inconnue, non fiable, légèrement fiable ou totalement fiable. Ces connotations se réfèrent à la capacité de cette personne de certifier la clé publique de quelqu'un d'autre, i.e., sa compétence comme introducteur.

Pour la validation d'une clé publique, le niveau de confiance attribué à chaque signature de certification est examiné et un score pondéré de fiabilité est calculé. Par exemple, deux signatures légèrement fiables peuvent être jugées aussi crédibles qu'une signature totalement fiable. Cette méthode d'évaluation est configurable par l'utilisateur ; il peut exiger deux signatures totalement fiables ou trois légèrement fiables pour considérer une clé comme valide.

Au cours du temps, chaque utilisateur accumulera des clés des personnes qu'il souhaite désigner comme introducteurs. De même, tous les utilisateurs accumuleront des signatures de certification attribuées par d'autres personnes, en espérant que toute autre entité fera confiance à au moins une ou deux de ces signatures. Cela peut résulter en l'émergence d'un graphe de confiance résistant et décentralisé comprenant toutes les clés publiques du système.

5.3.2. Vulnerabilités du Modèle PGP

L'approche décentralisée et non institutionnelle qui PGP utilise pour distribuer les clés publiques a ses bénéfices, mais peut poser quelques problèmes de gestion qui peuvent ajouter des vulnérabilités au système résultant, comme nous le remarquerons dans les paragraphes suivants.

5.3.2.1. Estampilles de Temps

L'une des vulnérabilités du modèle PGP est celle de la possibilité de création d'estampilles de temps trompeuses. Puisque chaque utilisateur signe localement son propre certificat, le modèle ne peut pas empêcher un utilisateur peu honnête de modifier l'horloge de son système (system's clock) pour générer ses certificats et signatures ayant des estampilles de temps volontairement erronées.

Un utilisateur peut faire comme s'il avait signé un document avant ou après la date réelle de signature ; ou comme sa paire de clés avait été générée avant ou après la date réelle de sa génération. Cela peut amener un avantage ou profit (bénéfice) légal ou financier, par exemple en créant un certain type d'échappatoire qui permettrait de renier une signature.

Ainsi, pour que des transactions électroniques effectuées sous le modèle PGP puissent avoir de valeur légale, il est nécessaire la disponibilité d'un notaire ayant la tache d'appliquer des estampilles de temps dignes de foi aux certificats et aux signatures électroniques.

5.3.2.2. Révocation de Clés Publiques

L'approche décentralisée et non institutionnelle adoptée par PGP pour la distribution les clés publiques implique également l'impossibilité de s'appuyer sur une liste unique et centralisée de clés publiques dont le composant privé a fait l'objet de compromission. Ce mécanisme rend le système quelque peu incapable de limiter les conséquences de la compromission de clés privées.

Le seul mécanisme offert par PGP pour restreindre l'utilisation de clés publiques dont le composant privé a été découvert est l'émission de "certificats de révocation de clé". Ces certificats sont signés au moyen de la clé privée qui a été découverte et servent à annoncer qu'une clé publique n'est plus associée à l'identité de son détenteur. Ce certificat doit alors être divulgué le plus largement possible pour prévenir les utilisateurs que dorénavant ils ne doivent plus utiliser cette clé pour établir des communications sûres.

La faiblesse d'un tel mécanisme de révocation de clés publiques entraîne le fait que lorsque l'on récupère et/ou utilise un certificat de clé publique PGP, on n'est pas tout à fait assuré de sa validité, sauf par contact direct avec le propriétaire de la clé.

5.4. Le Modèle Chimæra

Chimæra est un modèle pour la sécurité des systèmes ouverts proposé par Anas Altarah [Alt92]. Ce modèle introduit une nouvelle vue de l'organisation des Autorités de Certification (CA) dans une structure de distribution de clés publiques sous la forme de certificats utilisés par les mécanismes d'authentification en vigueur aujourd'hui.

5.4.1. Distribution des Autorités de Certification

En ce qui concerne la distribution de CA dans le système, le modèle Chimæra critique l'approche hiérarchique "suggérée" par la Recommandation X.509 et adoptée par des implémentations existantes. Dû aux inconvénients et à la dépendance de l'approche hiérarchique (car c'est la seule implémentée aujourd'hui), Chimæra cherche à définir une organisation des CA "en réseau", s'appuyant sur l'analogie entre la topologie des réseaux et la distribution des autorités de certification[13].

De la même façon que les routeurs cherchent à établir un circuit dans le réseau pour envoyer des paquets d'un noeud à l'autre, les systèmes de certification cherchent à établir des rapports de certification entre les CA de façon à permettre l'établissement de chemins de certification entre deux utilisateurs. Chimæra fait la correspondance entre une connexion entre deux routeurs et la certification croisée entre deux CA. Pour réaliser l'approche, Chimæra compare, d'une manière générale :

- les CA aux Routeurs,
- les Certificats Croisés aux Liens,
- les Chemins de Certification aux Circuits Virtuels.

De la même façon, Chimæra compare les domaines de routage aux domaines[14] de certification. Ceci est justifié par le fait qu'une interconnexion entre deux réseaux reflète un échange important de données entre les systèmes et un minimum de confiance et de connaissance entre les deux réseaux.

En partant du principe que, du point de vue du routage, le système OSI est composé de plusieurs *domaines de routage* interconnectés, l'auteur considère le système global comme l'interconnexion de plusieurs *domaines de certification*. Pour concrétiser cela, le modèle définit une classification en trois niveaux pour les éléments d'un domaine de certification, toujours en analogie avec les notions élémentaires de routage :

---

[13]La description du modèle Chimaera présente un mélange de concepts assez important, à savoir, les concepts de "certification" et "d'établissement de chemin de certification" sont confondus. En fait, la certification dénote l'acte de certifier la clé publique d'un principal en produisant un certificat. Par contre, la procédure proposée dans la Recommandation X.509 pour la validation de la clé publique d'un principal est définie comme "obtention de la clé publique de l'utilisateur" ou "établissement d'un chemin de certification".

[14]Un domaine est défini par un ensemble des règles ou des critères qui peuvent être de nature admistrative, politique ou géographique.

- les *utilisateurs* du système d'authentification analogues aux *systèmes d'extrémité* (ES, End Systems),
- des *CA de Bas Niveau* analogues aux *systèmes intermédiaires* (IS, Intermediate Systems),
- des *CA de Haut Niveau* analogues aux *systèmes frontaliers* (BIS, Boundaring Intermediate Systems).

Dans le modèle proposé, les CA de bas niveau sont chargées de certifier les entités locales au domaine et de la certification mutuelle avec toutes les autres CA d'un même domaine. Les CA de haut niveau ont les mêmes tâches que les CA de bas niveau, avec la charge supplémentaire de certification entre les domaines. La figure 10 illustre l'approche de distribution de CA dans Chimæra.

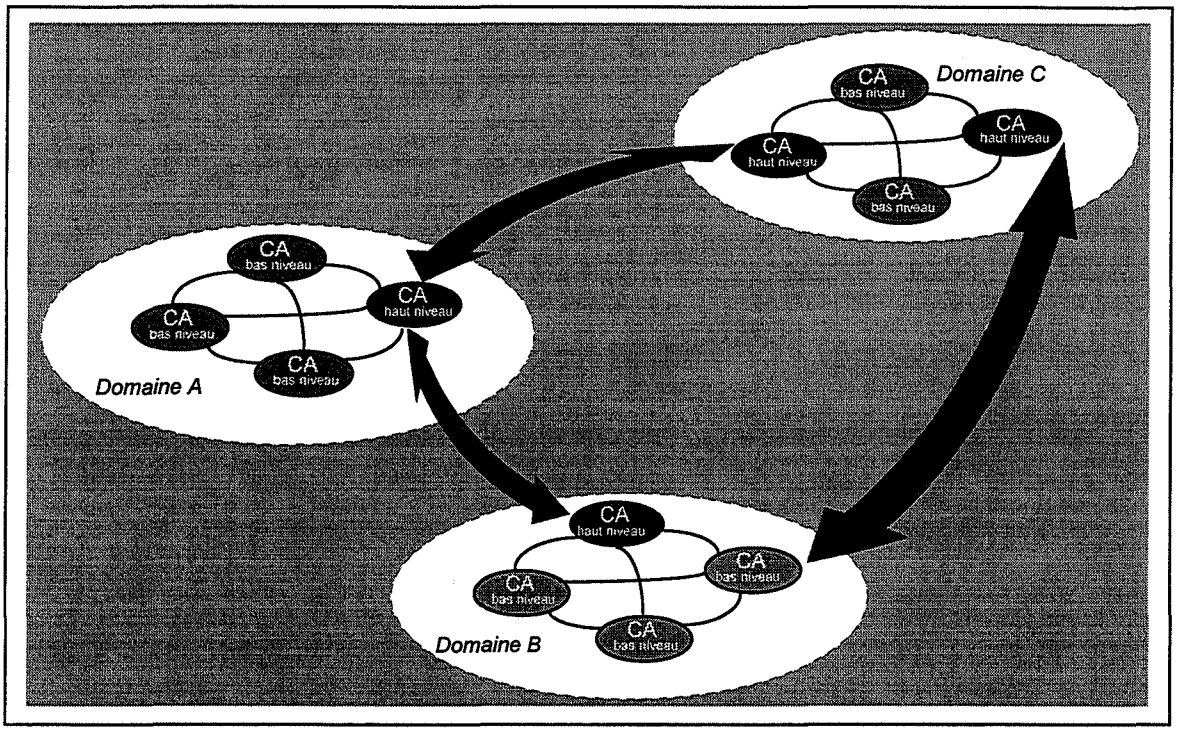

Figure 10. Modèle de Certification Chimaera

5.4.2. Etablissement de Chemins de Certification

L'objectif de l'analogie entre la topologie des réseaux et la disposition des CA est de représenter la distribution de CA par un graphe pour essayer d'appliquer les algorithmes de routage pour l'établissement de chemins de certification. Pour ce faire, nous pouvons représenter le système par un graphe dont chaque noeud représente une CA et dont chaque arc représente un certificat croisé entre deux CA. Par exemple, dans le cas de l'algorithme de plus court chemin, Chimæra propose de reprendre la même idée pour construire le "meilleur chemin de certification" parmi un ensemble de CA. Comme dans le cas du routage, nous devons définir une métrique permettant d'évaluer les chemins. En effet, plusieurs métriques peuvent être retenues :

- En donnant la même valeur à tous les certificats croisés du système, le calcul du meilleur chemin de certification revient à calculer le chemin de certification le plus court, i.e., contenant le moins de certificats croisés.
- En attribuant à chaque certificat croisé une valeur de confiance, le calcul du meilleur chemin de certification revient à trouver le chemin le plus fiable.

Dans Chimæra, la construction et la vérification des chemins de certification est à la charge des CA. Les CA de bas niveau communiquent entre elles afin d'établir des chemins de certification dans le cas où tous les certificats nécessaires sont internes au domaine, i.e., l'authentification entre deux principaux situés dans un même domaine. Les CA de haut niveau communiquent entre elles afin d'établir des chemins de certification dans les cas où des certificats inter-domaine sont nécessaires, i.e., authentification entre deux principaux situés dans des domaines différents.

Si une CA de bas niveau cherche à établir un chemin de certification vers une CA appartenant à un autre domaine, vers laquelle un chemin n'a pas encore été etabli, elle délègue cette tâche à la CA de haut niveau la plus proche appartenant toujours à son propre domaine. Ceci est facilité par le fait que les CA de haut niveau sont certifiées mutuellement avec toutes les autres CA de son domaine. Par contre, cela nécessite l'existance d'au moins une CA de haut niveau par domaine.

Pour faciliter l'échange d'informations entre les CA, Chimæra propose un protocole pour la communication entre ces entités. Le Protocole de Communication des CA, appelé *CAC-Protocol*, a été défini avec l'objectif de faciliter la coopération et l'échange d'informations entre les CA pour permettre l'établissement des chemins de certification. Bien que ce protocole soit utilisé entre les CA des deux niveaux, il existe une différence dans l'utilisation de ce protocole suivant le type de CA : une CA de bas niveau ne peut pas communiquer directement avec des CA appartenant à d'autres domaines, elle doit envoyer sa requête à une CA de haut niveau de son domaine qui se chargera de l'interaction avec d'autres domaines pour établir le chemin de certification en question.

Le modèle inclut la possibilité d'exprimer la confiance dans les certificats et dans les chemins de certification pour donner lieu à la notion d'*Evaluation de Certificats*. L'idée consiste à attribuer une métrique et une sémantique des valeurs aux certificats et aux chemins de certification. Les métriques adoptées doivent avoir un rapport avec la politique de sécurité du domaine. Cette approche a pour but d'attribuer des degrés de confiance à chaque certificat ou chemin de certification dans le système et de permettre le choix du meilleur chemin de certification, selon une métrique choisie, dans les cas où plusieurs chemins sont disponibles.

5.4.3. Lacunes du Modèle Chimæra

En ce qui concerne la distribution des autorités de certification, bien que Chimæra critique l'apprcohe de distribution hierarchique des CA, il faut noter le fait que ce sont les CA de haut niveau qui certifient les clés publiques des CA de bas niveau. Ainsi, l'organisation des CA proposée par Chimæra est aussi hiérarchique. Par consequent, cette approche subit également l'inconvénient que la sécurité des noeuds inférieurs dépend de la sécurité des noeuds supérieurs. La compromission d'une CA de haut niveau met en péril tout un domaine.

De plus, on revient au problème de l'approche hiérarchique quant à la vérifiabilité[15] des certificats des CA de niveau inférieur. En cas de compromis détecté de la CA de haut niveau, le domaine qu'elle gère devient isolé de l'extérieur. Cela représente une dégradation du service car la verifiabilité des certificats d'un domaine à partir de l'extérieur dépend de la vérifiabilité des certificats des CA de haut niveau.

Le fait que toutes les CA d'un domaine soient cross-certifiées entre elles, implique que le compromis de n'importe quelle CA met en péril toutes les CA d'un domaine. Par conséquent, un tel événement mettra aussi en péril tout domaine connecté au domaine mis en danger. De plus cela exige un effort considérable de gestion, d'abord pour mettre en place le système, ensuite pour le remplacement de certificats périmés, mais aussi pour la révocation de certificats.

Lorsque le compromis d'une CA de bas niveau est détecté, il faut le communiquer à toutes les CA du domaine pour que toutes les listes de révocation soient mises à jour. Cela devient plus complexe dans le cas de compromis d'une CA de haut niveau car l'information doit être propagée vers plusieurs domaines. De plus, dans le cas spécifique du modèle Chimæra, en cas de compromis, non seulement les listes doivent être mises à jour, mais aussi la valeur de confiance attribuée aux certificats et aux chemins de certification qui sont distribués dans le système, ce qui est fort laborieux.

En ce qui concerne le mode d'opération des CA, Chimæra suppose que toutes les CA du système opèrent comme des entités *on-line*. Cela contredit un des principes de protection des CA et la tendance actuelle d'établissement des CA dans des boîtes "*tamper-proof*" qui s'auto-détruisent même si elles sont ouvertes.

En outre, le protocole de communication entre les CA défini pour l'obtention de clés publiques, il ne prend pas en compte l'obtention de clés dans des contextes multi-destinations ni la spécification de plusieurs métriques et/ou la combinaison de métriques.

Finalement, Chimæra présente le mécanisme d'évaluation de certificats comme "une alternative élégante aux listes de révocation de certificats, car une CA locale, sachant qu'une CA participant à la construction d'un chemin de certification a été compromise, peut considérer ce chemin de certification comme invalide". Nous avons des critiques à cet argument.

Il est clair que toute entité doit refuser un chemin de certification contenant des certificats appartenant à des CA compromises. Le problème est qu'une CA n'a pas le pouvoir de "deviner" qu'une autre CA est compromise. Les listes de révocation sont (au moins pour l'instant) le moyen de publier ce genre d'événement. En fait, les mécanismes d'évaluation de certificats ne doivent par être considérés comme des remplaçants des listes de révocation, mais comme des compléments aux car il permet un "filtrage" des certificats et des chemins de certification.

5.5. Les Specifications PKCS

[15]vérifiabilité: capacité ou potentiel de vérification.

PKCS (ou **P**ublic-**K**ey **C**ryptography **S**tandards) est un ensemble de documents développés par un consortium formé par des organismes académiques et industriels et coordonné par RSA Data Security. Ce consortium comprend RSA Inc., MIT, Lotus, MicroSoft.

Le groupe de travail PKCS considère que certains aspects de la cryptographie à clé publique doivent être normalisés, de la même façon que dans le monde OSI on se concentre plutôt sur les définitions des éléments de service et sur le format des données, sans prendre en compte les mécanismes et les algorithmes eux-mêmes. Ainsi, tout au long de ses travaux, PKCS vise à proposer une définition cohérente et standard de certains mécanismes (ou applications) de la cryptographie à clé publique qui sont :

* **Les signatures numériques**

  PKCS considère que trois paramètres doivent être normalisés, à savoir :

  - une syntaxe pour les données signées indépendante des algorithmes,
  - des algorithmes de scellement spécifiques,
  - des techniques de chiffrement spécifiques.

* **L'enveloppe sécurisée**

  L'enveloppe sécurisée (digital enveloping) est une application permettant à un expéditeur "d'enfermer" un message la sorte que personne d'autre que le destinataire prétendu ne soit capable de "l'ouvrir". Typiquement, la réalisation de ce service consiste à chiffrer un message au moyen d'un algorithme symétrique, dont la clé de chiffrement est ensuite chiffrée au moyen d'un algorithme asymétrique.

  Selon PKCS, trois paramètres liés à l'enveloppe sécurisée doivent être normalisés, à savoir :

  - une syntaxe indépendante d'algorithme pour l'enveloppe sécurisée,
  - des techniques de chiffrement symétrique spécifiques,
  - des techniques de chiffrement asymétrique spécifiques.

* **La certification de clés publiques**

  PKCS distingue cinq paramètres qui doivent être normalisés, à savoir :

  - un format indépendant d'algorithme pour les certificats,
  - un format indépendant d'algorithme pour des certificats étendus et extensibles,
  - un format indépendant d'algorithme pour les listes de révocation,
  - des syntaxes de représentation pour les clés publiques,
  - des algorithmes de signature numérique.

* **L'accord sur clé secrète** (key agreement)

  Il s'agit de définir une procédure permettant à deux parties communicantes, sans arrangement préalable, d'échanger ou d'établir une clé secrète connue seulement par ces deux parties. Cette clé secrète peut alors servir au chiffrement d'une communication future entre ces deux parties.

Dans ce contexte, PKCS considère deux aspects qui doivent être normalisés, à savoir :

- un format indépendant d'algorithme pour les messages destinés à la réalisation de ce service,
- des algorithmes spécifiques à ce propos.

PKCS constitue une collection de composants dits PKCS #1, #3, #5, #6, #7, #8, #9, #10[16] ; le but étant d'offrir une base pour le développement des applications sécurisées interopérables. Au niveau de l'interopérabilité avec d'autres approches existantes, les travaux PKCS visent à :

∇ maintenir la compatibilité avec PEM tant en ce qui concerne le format de certificat, que la syntaxe de messages sécurisés de façon à permettre la conversion de messages PEM en messages PKCS et vice-versa,

∇ proposer des evolutions à l'approche PEM de manière à être capable de manipuler des types de données quelconques, d'intégrer un ensemble plus riche de données dans des certificats, de supporter la méthode d'échange de clés Diffie-Hellman [DH76], et de pouvoir offrir un ensemble plus riche de services de sécurité à travers l'utilisation de signatures numériques et de techniques d'encapsulation,

∇ proposer des standards et des méthodes convenables pour l'incorporation dans des travaux futurs OSI. Les documents PKCS utilisent l'ASN.1 pour la représentation abstraite de données et reposent sur le BER (**B**asic **E**ncoding **R**ules) comme syntaxe de transfert.

Dans PKCS, la question de la définition de techniques communes pour la sécurisation de systèmes ouverts prend en considération deux niveaux indépendants d'abstraction. Le premier niveau traite la syntaxe de données véhiculées, tandis que le deuxième niveau traite les algorithmes et mécanismes spécifiques. L'idée est de faire en sorte que ces deux aspects soient orthogonaux. Par exemple, une définition pour la syntaxe de messages signes doit convenir à tout algorithme de chiffrement à clé publique (et non seulement l'algorithme RSA) ; en contrepartie, une définition d'un mécanisme de chiffrement (qui peut être le RSA) doit être applicable aux différents formats de messages signés. Dans la suite, nous décrivons brièvement les traits de chacun des documents PKCS.

---

[16]Les documents PKCS #2 et PKCS #4 ne sont plus actifs.

5.5.1. PKCS #1

PKCS #1 couvre les aspects liés à l'algorithme RSA. Il spécifie des variantes de cette méthode de chiffrement qui incluent des techniques de remplissage capables d'augmenter la sécurité des données chiffrées. Trois variantes de remplissage sont définies, à savoir, "block type 00", "block type 01" et "block type 02".

Dans le mode "block type 00", la chaîne de remplissage est constituée d'octets de valeur 00. Cela équivaut, en fait, au chiffrement sans remplissage et est défini pour maintenir la conformité avec la méthode classique de chiffrement RSA, qui est, par ailleurs, celle utilisée dans le cadre X.509.

Dans le mode "block type 01", la chaîne de remplissage est composée d'octets de valeur FFx. Ce mode de remplissage est recommandé pour les opérations de chiffrement avec clé privée. Cela vise à protéger les données chiffrées contre des attaques du type proposé par Desmedt et Odlyzko [DO85].

Dans le mode "block type 02", chaque octet de chaîne de remplissage doit être choisi de façon aléatoire. Ce mode de remplissage est recommandé pour les opérations de chiffrement avec clé publique. Inspiré des résultats de Hastad [Has88], ce mode de remplissage vise à protéger les données chiffrées contre des attaques par déduction.

En ce qui concerne la représentation (ou stockage) de clés, d'une part PKCS #1 reprend la syntaxe de description de clés publiques définie par X.509 ; d'autre part, PKCS #1 définit une syntaxe de représentation de clés privées, où d'autres paramètres liés à la génération de clés sont stockés pour augmenter l'efficacité des opérations à clé privée. Cette définition respecte les propriétés décrites par Quisquater et Couvreur [QC82].

Finalement, PKCS #1 spécifie trois algorithmes de signature numérique qui correspondent aux méthodes de chiffrement avec remplissage mentionnés ci-dessus, couplés aux algorithmes de scellement MD2 [Kal92], MD4 [Riv90] et MD5 [Riv92]. Pour permettre l'identification non ambiguë de ces méthodes, le groupe de travail PKCS définit des identificateurs d'objets pour chacune de ces techniques.

5.5.2. PKCS #2

Le document PKCS #2, qui traitait les aspects liés à l'algorithme RSA et aussi aux résumés de messages (à travers les fonctions de scellement), n'existe plus. Ce document a été incorporé dans la version actuelle de PKCS #1.

5.5.3. PKCS #3

PKCS #3 définit une approche pour l'implémentation de la méthode Diffie-Hellman d'établissement de clé secrète. A travers cette méthode, deux parties peuvent convenir d'une clé secrète, sans aucun accord préalable, de sorte que la clé résultante ne peut être découverte par une tierce personne observant le dialogue.

Cette clé secrète peut alors être utilisée pour chiffrer la communication postérieure entre les deux parties. Cette approche a été originairement conçue pour la sécurisation des associations, telle que proposé pour les couches transport et réseau du modèle OSI [ISN6285] [ISN2559].

L'approche consiste à déléguer à une autorité centrale la tâche de génération des paramètres publics de la méthode Diffie-Hellman, dits g et p. Ensuite, chacune des deux parties impliquées (disons A et B) choisit, en privé, sa valeur secrète SV_A et SV_B et, en utilisant les paramètres communs, calcule sa valeur publique PV_A et PV_B.

PV_A et PV_B sont calculés tels que $PV_A = g^{SV_A} \bmod p$ et $PV_B = g^{SV_B} \bmod p$. Apres avoir échangé les valeurs publiques, les deux parties sont capables de calculer leur clé secrète commune SK, qui est telle que:

$$SK = PV_B^{SV_A} \bmod p = PV_A^{SV_B} \bmod p$$

Pour permettre l'implémentation de cette approche, PKCS #3 définit un identificateur d'objet et une syntaxe pour l'échange des paramètres. Ceux-ci sont désignés de manière à pouvoir être employés dans des protocoles OSI ou dans n'importe quelle implémentation basée sur le cadre X.509, à travers la macro "ALGORITHM".

5.5.4. PKCS #4

Le document PKCS #4, qui définissait une syntaxe de représentation de clés RSA, n'existe plus. Ce document a été incorporé dans la version actuelle de PKCS #1.

5.5.5. PKCS #5

PKCS #5 définit une méthode pour le chiffrement de données en utilisant une clé secrète dérivée d'un mot de passe. La méthode utilise l'algorithme DES en mode CBC pour le chiffrement, où la clé secrète est dérivée à travers le scellement d'un mot de passe en utilisant l'algorithme MD2 ou MD5.

Bien que cette méthode puisse être utilisée pour chiffrer une chaîne d'octets quelconque, elle a été désignée originairement pour chiffrer des clés privées lorsqu'elles sont transférées d'un système à l'autre.

Dans cette méthode, chaque entité choisit, en privé, une chaîne d'octets P, comme son mot de passe. L'entité doit également sélectionner une chaîne de huit octets comme sa valeur de "salt" et un entier comme compteur d'interactions. La valeur de "salt" est ajoutée au mot de passe pour limiter l'efficacité des attaques par dictionnaire [MT79], et le compteur d'interactions détermine le nombre d'applications de la fonction de scellement sur le mot de passe. Ceci rend les attaques par dictionnaire encore plus coûteuses.

Pour permettre l'implémentation de cette approche, PKCS #5 inclut la définition d'un identificateur d'algorithme dans les termes X.509, i.e., un identificateur d'objet et une syntaxe de représentation des paramètres, qui dans ce cas sont la valeur de salt et le compteur d'interactions.

5.5.6. PKCS #6

PKCS #6 définit un format de certificat étendu. Ceci est constitué par un certificat du type X.509, déjà signé par une CA, et en un ensemble d'attributs qui fournissent d'autres informations sur le détenteur du certificat. Le tout est collectivement signé par la même CA émettrice du certificat X.509.

Ainsi, tant les attributs que le certificat X.509 sont vérifiés au moyen d'un seule opération cryptographique. De plus, le format de certificat PKCS #6 permet d'extraire le certificat X.509 sans intervention cryptographique. Le certificat PKCS #6 est défini comme suit:

```
ExtendedCertificate ::= SIGNED  SEQUENCE {
    version      Version,
    certificate  Certificate, -  as defined by the X.509 Authentication Framework
    attributes   Attributes }

Version ::= INTEGER

Attributes ::= SET OF Attribute
```

Où,

- L'élément "version" indique le numéro de version du format de certificat. Cela permet de maintenir la compatibilité avec des versions futures.
- L'élément "certificate" constitue un certificat conforme à la Recommandation X.509.
- L'élément "attributes" contient un ensemble d'attributs qui peuvent être nécessaires à l'identification de l'entité certifiée. Cela permet d'associer d'autres informations d'identification à la clé publique d'une entité, telles que l'adresse électronique, l'adresse de présentation ou le rôle que l'entité remplit dans une organisation.

Il faut rappeler que le format certificat definit dans X.509 ne permet d'associer qu'un nom d'annuaire à la clé publique d'une entité. D'autres applications peuvent requérir d'autres types d'information pour l'identification des entités. Le format de certificat étendu PKCS #6 permet alors de remédier le format restreint des certificats X.509. PKCS #9 présente une liste non exhaustive d'attributs mais il est prévu que cette liste soit étendue selon les besoins.

5.5.7. PKCS #7

Ce document propose une syntaxe générale pour des données sécurisées au moyen des techniques cryptographiques, telles que des signatures numériques ou des enveloppes sécurisées. Cette syntaxe admet la recursivité de telle sorte qu'une enveloppe puisse être imbriquée dans un autre, ou qu'une partie puisse signer des données déjà soumises à l'envelopage sécurisé.

PKCS #7 permet d'authentifier des attributs quelconques avec le contenu d'un message, et d'associer des contre-signatures à une signature préalablement appliquée aux données.

La syntaxe proposée est suffisamment générale pour supporter de nombreux types de contenus. Actuellement, PKCS #10 en définit six, à savoir, "données", "données signées", "données

enveloppées", "données signées et enveloppées", "données scellées" et "données chiffrées". La syntaxe générale pour les contenus échangés est la suivante:

```
ContentInfo ::= SEQUENCE {
    contentType     OBJECT IDENTIFIER,
    content         [0] EXPLICIT ANY DEFINED BY contentType OPTIONAL
}
```

Le contenu du type "données" est le plus simple et est défini comme un chaîne d'octets dont la sémantique est laissée aux applications concernées.

Le contenu du type "données signées" est un contenu de type quelconque qui peut être accompagné par des signatures produites par plusieurs signataires différents.

Le contenu du type "données enveloppées" est un contenu chiffré d'un type quelconque accompagné des exemplaires de la clé de chiffrement chiffrées pour chaque destinataire. La combinaison "contenu chiffré plus clé de chiffrement" constitue une enveloppe sécurisée pour chaque destinataire. Cette enveloppe sécurisée est telle que:

- la clé de chiffrement est la même pour tous les destinataires et doit être généré aléatoirement,
- pour chaque destinataire, la clé de chiffrement doit être chiffrée au moyen de la clé publique de ce destinataire.

Le contenu du type "données signées et enveloppées" consiste en un contenu chiffré d'un type quelconque accompagné des exemplaires chiffrées de la clé de chiffrement pour chaque destinataire et des signatures "doublement" chiffrées par un ou plusieurs signataires. Le double chiffrement comprend le chiffrement du résumé du contenu avec la clé privée du signataire suivi du chiffrement avec la même clé utilisée pour chiffrer le contenu.

Le but de ce deuxième chiffrement est d'éviter qu'un adversaire puisse récupérer le résumé (résultat de scellement) du contenu. Sans cela, un adversaire serait capable de déterminer, parmi un ensemble de "candidats à contenu", lequel correspond au contenu en question, en comparant son résumé avec les résumés des "candidats".

Le contenu du type "données scellées" est un contenu d'un type quelconque accompagné d'un résumé produit au moyen d'une fonction de scellement. L'application prévue pour ce type de contenu est qu'il soit, en fait, utilisé comme contenu d'entrée pour les contenus du type "données enveloppées" de manière à incorporer le service d'intégrité.

Le contenu du type "données chiffrées" est uniquement un contenu chiffré d'un type quelconque. Ce type de contenu n'a pas de destinataires ni de clés de chiffrement. Ce type de contenu est destiné au stockage local des donnés confidentielles. Dans un tel cas, la méthode de chiffrement peut être celle basée sur des mots de passe, spécifiée dans PKCS #5.

5.5.8. PKCS #8

PKCS #8 spécifie une syntaxe pour le stockage des informations privées chiffrées. Ces informations privées incluent la clé privée et un ensemble d'attributs dont les éléments sont

choisis par chaque utilisateur. Le but est de fournir aux utilisateurs un moyen de protection de données simple et fiable.

A travers cette méthode un utilisateur pourrait protéger, par exemple, le certificat des CA qui ne peuvent pas être vérifiés dans le modèle de confiance et qui sont distribués par des moyens sûrs, tel que le certificat des CA de plus haut niveau dans un modèle de certification hiérarchique. Pour le chiffrement de ces informations, PKCS #8 recommande la méthode de chiffrement basée sur mot de passe spécifiée dans PKCS #5.

Cette méthode semble redondante par rapport à celle fournie à travers le type de contenu "données chiffrées" de la syntaxe PKCS #7.

5.5.9. PKCS #9

PKCS #9 présente la liste d'attributs utilisables :

- dans des certificats du type PKCS #6,
- dans d'enveloppes sécurisées du type PKCS #7,
- comme des informations privées du type PKCS #8,
- dans les requêtes de certification du type PKCS #10.

Cela inclut les types d'attributs normalisés de l'annuaire X.500 [X.520] et de la messagerie X.400 [X.402] et bien d'autres types d'attributs spécifiques à PKCS tels que les types spécifiés dans les documents PKCS et des formats non structurés pour la désignation des noms et des adresses électroniques ou postales.

5.5.10. PKCS #10

A travers ce document, le consortium PKCS propose une syntaxe de requête pour l'obtention d'un certificat. Dans PKCS #10, une demande de certificat comprend un nom d'annuaire, une clé publique et, de manière optionnelle, un ensemble d'attributs. L'entité qui requiert la certification, signe cette requête avec le composant privé correspondant à la clé publique contenue dans la requête. La requête est alors envoyée à une CA, qui la transformera en un certificat du type X.509 ou PKCS #6, selon la présence ou non des attributs supplémentaires.

Un problème se pose en ce qui concerne ces attributs supplémentaires: Etant donné que ces attributs ne sont pas signés par une autorité de désignation, mais sont fournis directement par l'utilisateur à la CA, en principe, cette dernière n'a pas les moyens de s'assurer que ces attributs sont véritablement associés au prétendu utilisateur. Cela peut constituer une source d'attaque au système de certification.

Par exemple, si un utilisateur A réussi à faire certifier sa clé publique avec l'adresse électronique d'un autre utilisateur B, et d'autres usagers utilisent la clé publique contenue dans ce certificat pour envoyer des données secrètes à B, alors A sera capable d'accéder, même temporairement, à ces informations confidentielles destinées à B.

CHAPITRE 6
NORMALISATION DE LA SECURITE DANS OSI

Le modèle de référence pour l'interconnexion des systèmes ouverts (OSI) [X.200] est devenu globalement accepté comme la base de développement des systèmes ouverts. Des grands fabricants de systèmes informatiques ont adhéré aux normes OSI et même d'autres réseaux, comme l'Internet, commencent à développer leur stratégie d'évolution en essayant une convergence avec l'approche OSI.

Au démarrage, les travaux d'OSI se sont concentrés plutôt sur les aspects d'interopérabilité, laissant de côté les aspects d'administration et de sécurité. Plus tard, le modèle OSI a été étendu avec le mode de communication non connecté (*connectionless*), et les aspects de nommage, d'adressage, d'administration et de sécurité. Malheureusement, ces extensions ne sont pas encore au point comme l'est le modèle de base OSI.

Bien que notre travail ne soit pas dépendant ou exclusif au modèle OSI, ce modèle est utilisé comme cadre général de référence. Ainsi, nous devons maitenir une vision critique de ce modèle. Dans ce chapitre, nous allons donc discuter l'architecture de sécurité du modèle OSI et le placement des services de sécurité parmi les sept couches de ce modèles, pour ensuite aborder les modèles de sécurité spécifiques à deux applications clé de notre travail, à savoir l'annuaire X.500 et la messagerie X.400.

6.1. L'architecture de Sécurité OSI

La norme "*OSI Security Architecture*" [X.800] aborde la sécurité en listant un nombre de menaces et ensuite définit un nombre de services et de mécanismes de sécurité avec pour but la protection contre ces menaces. Cette norme définit les emplacements possibles pour ces services et mécanismes parmi les sept couches du modèle OSI, mais laisse encore de nombreux points ouverts dont nous pouvons citer les suivants :

- La plupart des services peuvent être placés dans plusieurs couches (spécialement les couches 3, 4, 6 et 7). Cette liberté doit être restreinte pour

permettre l'interopérabilité entre les systèmes sans affecter la sécurité du système résultant.

- La norme ne donne aucune indication sur l'intégration de ces services et mécanismes dans les systèmes (services et protocoles existants), sur comment leur utilisation doit être négociée et activée.

Les divers services et applications des systèmes ouverts ont différents besoins en sécurité. Pour répondre à ces besoins, le cadre générale de sécurité OSI [X.800] mentionne cinq services de sécurité, à savoir : l'authentification, intégrité, la confidentialité, la non-répudiation et le contrôle d'accès. De plus, ce cadre définit des sous classes pour certains de ces services, afin de distinguer :

- l'application d'un service durant *toute* une association, de l'application sélective du service à *chaque* PDU[17] échangée durant une association ; ou bien pour distinguer entre l'application d'un service à des protocoles en mode connecté et non connecté. Par exemple, le service d'intégrité en mode connecté et en mode non connecté.
- si un certain service doit tenter de récupérer l'information affectée lorsqu'une attaque est détectée. Par exemple, intégrité avec (ou sans) recouvrement d'erreurs.

En fait, le cadre donne très peu de directives sur la signification de ces sous-classes et les rapports entre elles. Ici, nous allons analyser les interdépendances entre ces services afin d'établir quelques observations générales.

Fondamentalement, les services de sécurité nécessaires aux deux principales modalités de services OSI (modes connecté et non connecté) semblent être approximativement les mêmes. Selon [X.800], la différence principale est que dans des services de sécurité en mode connecté, le service s'applique à toute la durée de la connexion ; tandis que pour les services en mode non connecté, le service peut s'appliquer ou non à chaque message.

En fait, l'utilisateur d'un service en mode connecté peut très bien souhaiter ou avoir besoin de sélectionner les services de sécurité différents pour chaque message et peut vouloir appliquer ou non un même service de sécurité sélectivement à chaque message échangé. De plus, comme nous l'avons conclu dans la Section 2.4 de la Partie I, les services de sécurité sont très liés entre eux.

Cette observation permet de conclure qu'il est important de disposer d'une vision claire et intégrée de la sécurité des systèmes ouverts. L'application effective de cette norme exige encore l'élaboration d'autres recommandations et des profils internationaux de sécurité qui définiraient en détail l'emplacement, l'implémentation et l'utilisation de ces services et mécanismes de sécurité vis-à-vis des protocoles existants.

Cette discussion essaie de donner une vision concrète de la façon dont les services et mécanismes de sécurité décrits dans l'architecture de sécurité OSI peuvent être placés parmi les couches OSI. Une définition consciencieuse et bien fondée du placement des mesures de sécurité sont d'une grande importance pour la définition et la construction de systèmes ouverts fiables.

---

[17]PDU : Unité de Données de Protocole.

6.1.1. A propos du Placement des Services de Sécurité

A l'origine le modèle de référence OSI supposait qu'à partir de la couche transport toute communication était en mode connecté et de bout-en-bout. Avec l'acceptation croissante du modèle client-serveur, le mode de communication non-connecté a été inclus pour répondre aux besoins de ce modèle. En outre, les services relayés comme celui de la messagerie électronique deviennent de plus en plus importants. Cependant le modèle OSI n'a pas été à l'origine conçu pour ces types de service, laissant ainsi beaucoup de ses fonctionnalités en dehors du monde OSI et à la charge des applications elles-mêmes.

Certains auteurs [Kar91] considèrent que tous les services de sécurité peuvent être placés au niveau de la couche transport pour garantir des fonctionnalités de sécurité de bout-en-bout et un niveau de granularité suffisante. Nous ne sommes pas tout à fait d'accord avec cet argument. Nous en donnerons les raisons par la suite.

La couche transport n'est pas capable de décoder une PDU de la couche application pour authentifier un utilisateur final. La couche transport peut être considérée comme fournisseur de services de bout-en-bout au niveau des machines mais non au niveau de l'utilisateur final. Ainsi, cette couche peut être utilisée pour la mise en place d'un contrôle d'accès à ce niveau là.

Il est vrai que lorsque des services de sécurité sont offerts par des couches plus basses, ces services deviennent immédiatement disponibles à une gamme plus vaste de protocoles et services. Toutefois, il faut reconnaître qu'une vraie vision/fonctionnalité "de bout-en-bout" de la sécurité ainsi qu'une granularité suffisante ne sont accomplies qu'en plaçant les fonctions de sécurité au niveau de la partie des applications qui s'occupe de l'utilisateur final.

Certaines architectures ne se satisfont pas uniquement d'un contrôle d'accès généralisé mais nécessitent la prise en compte de services de sécurité depuis la station de l'utilisateur jusqu'au destinataire (station de travail ou serveur). Il s'agit des environnements tels que la messagerie, l'EDI et les échanges de fichiers. Dans ce cas, l'implantation des techniques de sécurité ne peut pas être centralisée en un point défini du réseau, mais doit être répartie sur l'ensemble des stations utilisateurs et doivent être interopérables.

Par exemple, dans le cas des services relayés tels que la messagerie X.400, du point de vue OSI, chaque agent intermédiaire (un agent de transfert de message dans X.400) constitue une instance de communication de bout-en-bout. En fait, chaque pas intermédiaire fonctionne comme un sous-réseau à traverser. Ces agents exécutent des fonctions de routage et d'administration (dans le sens application) basés sur des informations de contrôle qui accompagnent les données de l'utilisateur final. Ainsi, tout agent intermédiaire est capable de toutes sortes d'attaque active ou passive.

Le placement des services de sécurité au niveau de la partie des applications liée à l'usager final permet :

- de répondre aux particularités de chaque service,
- le contrôle direct des utilisateurs finals,
- de restreindre l'accès au matériel cryptographique privé de chaque utilisateur.

De plus, le service de non-répudiation est strictement dépendant de l'application et ne peut donc être placé ailleurs. Par conséquent, nous adhérons personnellement à "l'argument de bout-en-bout" déjà défendu auparavant par Clark [SRC84] et Voydock [VK83].

En contrepartie, lors de l'implémentation, une telle approche doit être soigneusement architecturée sous peine d'avoir à définir et à implémenter les mêmes fonctions et mécanismes de sécurité pour chaque application, ce qui peut amener à une duplication inutile d'effort et de fonctionnalité.

6.1.2. Mesures Complémentaires dans les Couches Basses

Des mesures de sécurité complémentaires peuvent être introduites au niveau des couches 1 à 4. Il doit être noté qu'une telle approche ne sert pas à garantir la sécurité des services mais à renforcer la sécurité globale des systèmes. Des mesures telle que la confidentialité du flux de trafic ne peuvent être efficacement réalisées qu'à travers ces couches-là.

Normalement, lors de l'application de mesures de chiffrement aux données d'une couche, tous les en-têtes des PDU des couches au-dessous sont transmis en clair et sont donc vulnérables à l'analyse et à la manipulation non-autorisées.

En particulier, les adresses réseaux doivent rester en clair au moins dans les noeuds de routage, ce qui permet à ceux-ci d'analyser tout le trafic les traversant. Cela implique qu'un minimum de sécurité (mais non de confiance) doit être placé au niveau des routeurs.

Des mesures telles que le chiffrement et le remplissage de trafic peuvent être employées au niveau des couches physique et de liaison de données pour empêcher l'analyse de trafic entre les routeurs par une tierce partie malveillante [Kar91].

En qui concerne le placement de mécanismes de sécurité au niveau des couches réseau et transport, la différence réside dans le fait qu'une connexion de transport sert toujours à une seule instance de communication entre deux systèmes de la couche application, tandis qu'une connexion réseau peut être multiplexée entre plusieurs connexions de transport.

Pour une communication en mode non connecté, au niveau réseau les datagrammes sont envoyés entre les machines communicantes, tandis qu'au niveau transport ces datagrammes sont échangés entre les processus d'application servant ainsi à une seule instance de communication entre deux applications.

Bien que le placement des mesures de sécurité au niveau de la partie plus haute de la couche réseau puisse offrir des services de sécurité de "machine-à-machine", ces mesures ne pourraient pas offrir un service sécurisé au niveau de chaque instance de communication entre deux applications.

Quelques possibilités d'utilisation des mesures de sécurité dans les couches 1 à 3 sont :

- l'application de mécanismes de chiffrement au niveau de la couche physique afin d'offrir la confidentialité de flux de trafic, notamment en ce qui concerne le trafic transmis par des lignes de communication considérées suspectes,

- l'application de mécanismes de chiffrement au niveau de la couche de liaison de données sur des lignes de communication fiables, telles que des liaisons radiophoniques (*radio link*), et/ou des fonctions d'authentification, de confidentialité et d'intégrité entre paires de stations d'un média de diffusion (*broadcast medium*), tels que dans les LAN et les réseaux basés sur la radio communication cellulaire,
- l'application de fonctions de sécurité au niveau de la couche réseau pour sécuriser les communications entre deux machines d'un même sous-réseau ou de réseaux interconnectés.

Bien que les mécanismes de contrôle d'accès doivent en principe être laissés aux applications, il est souvent utile de disposer d'un contrôle d'accès multi-niveau dans un environnement distribué afin de réduire le nombre d'attaques capables d'atteindre les couches supérieures, par exemple les attaques du type *worm*.

- un mécanisme de contrôle d'accès au niveau de la couche liaison de données peut être employé pour limiter l'accès provenant de machines ou de réseaux critiques ou non fiables. Par exemple, on peut ignorer tous les messages provenant d'une certaine passerelle vers une certaine machine d'un site ou réseau,
- de même, le contrôle d'accès au niveau de la couche réseau peut utiliser les adresses réseau pour limiter et filtrer le flux de données entrant et sortant de machines critiques,
- le même contrôle d'accès ci-dessus peut être effectué au niveau de la couche transport, basé sur les points d'accès au service transport (TSAP - Transport Service Access Point).

L'élimination d'attaques plus génériques faite au niveau des couches basses des machines du type serveur ou même des passerelles, permet l'affectation de plus de ressources pour le traitement d'attaques plus ingénieuses visant des services ou des fonctionnalités spécifiques. Cependant, il doit être noté que les mesures ci-dessus ne doivent pas être considérées comme suffisantes mais comme complémentaires. Celles-ci sont employées pour limiter le nombre d'attaques capables d'atteindre des niveaux plus hauts des systèmes. Ces techniques n'offrent pas la granularité et le niveau de service nécessaires à l'utilisation sécurisée de plusieurs services par divers utilisateurs à travers un réseau considéré comme non fiable.

Encore une fois, comme l'on ne peut pas faire confiance à l'application de mesures de sécurité et au respect des politiques de sécurité de façon globale dans le réseau, il nous faut rejeter l'idée d'un réseau sûr et appliquer des mesures de sécurité indépendantes de la sécurité globale du réseau.

6.2. Des Applications Normalisées

Quelques applications OSI ont défini leurs propres modèles de sécurité. Nous allons présenter les approches retenues pour les services d'annuaire et pour la messagerie X.400. Ces applications ont fait, entre autres, l'objet de notre expérimentation, ce qui nous a permis d'en détecter des problèmes ouverts. Nous décrirons ceux-ci dans la partie suivante de ce travail.

6.2.1. L'Annuaire X.500

Avant de dépeindre le modèle et les services de sécurité de cette application normalisée, une brève description de l'annuaire X.500 et de ses fonctionnalités semble utile.

6.2.1.1. Aperçu General de l'Annuaire X.500

L'annuaire est un ensemble de systèmes ouverts qui coopèrent pour établir une base de données logique contenant des informations sur un ensemble d'objets dans le monde réel. L'information contenue dans l'annuaire est appelée Base de Données d'Annuaire (DIB - **D**irectory **I**nformation **B**ase).

La DIB est un ensemble d'entrées correspondant à des objets. Chaque entrée contient un ensemble d'informations sur un objet : les attributs de l'objet. Chaque attribut est typé et possède une ou plusieurs valeurs.

La DIB a une structure d'arbre : l'arbre d'information de l'annuaire (DIT - **D**irectory **I**nformation **T**ree). Le DIT est conçu suivant une hiérarchie de nommage regroupant toutes les entités du système. Chaque entrée de l'arbre correspond à un nom de la hiérarchie de nommage et donc contient un ensemble d'attributs propres à cette entrée. Afin d'éviter la centralisation de la base de données de l'annuaire, les sous-arbres du DIT sont distribués sur le système. Chaque partie de la DIB (qui peut être constituée de plusieurs sous-arbres) est liée à une entité gérante appelée DSA (**D**irectory **S**ervice **A**gent) qui est responsable de :

- l'administration de la DIB locale et des copies de DIB gérées par d'autres DSA,
- l'interaction avec d'autres DSA afin d'assurer l'ubiquité et la distribution du service de l'annuaire,
- offrir aux utilisateurs l'accès à la DIB.

Du coté de l'utilisateur, l'interrogation de l'annuaire est assurée par un DUA (**D**irectory **U**ser **A**gent). Le protocole d'accès à l'annuaire est défini par les recommandations X.500. Ce protocole est dit DAP (**D**irectory **A**ccess **P**rotocol). L'interaction entre DSA est assurée par le protocole DSP (**D**irectory **S**ervice **P**rotocol).

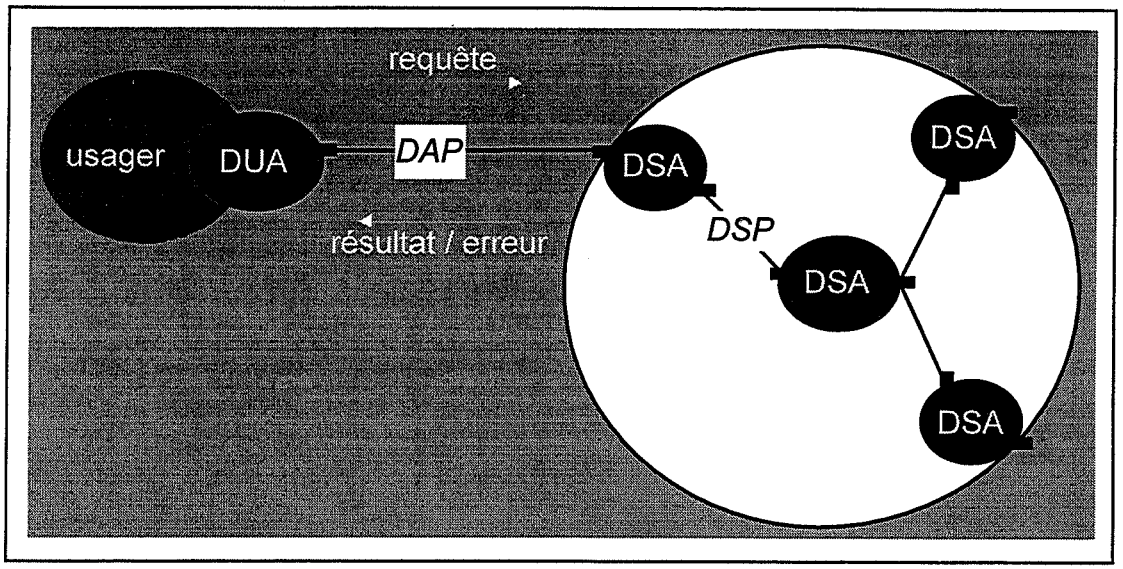

Figure 11. Modèle Fonctionnel de l'Annuaire X.500

L'ensemble de services offerts par l'annuaire constitue trois groupes d'Eléments de Services d'Application (ASE - **A**pplication **S**ervice **E**lement) qui sont :

* READ : les services de lecture, qui permettent, à partir du nom d'une entrée, de restituer des valeurs des attributs de cette entrée; ou de vérifier si une valeur donnée d'un certain attribut correspond à une valeur de cet attribut,
* SEARCH : les services de recherche, qui permettent de restituer la liste des subordonnés immédiats d'une entrée désignée; ou de restituer l'information provenant de toutes les entrées dans une certaine partie du DIT satisfaisant à un filtre,
* MODIFY : les services de modification, qui permettent de créer, de supprimer ou de modifier des entrées du DIT.

Ces services sont assurés par la coopération de l'ensemble des DSA de la sorte qu'une requête de l'utilisateur peut être relayée à un ou plusieurs DSA.

6.2.1.2. Le Modèle de Sécurité de l'Annuaire

L'objectif du modèle de sécurité de l'annuaire est garantir la protection de ses serveurs (DSA), de sa base d'information, de l'accès et de la communication entre ses composants contre des nombreuses menaces. Certaines d'entre elles sont traitées par le modèle de sécurité de l'annuaire, d'autres doivent être prises en compte par des mécanismes locaux et des mesures de maintenance. Les possibles menaces[18] à la sécurité de l'annuaire comprennent[19] :

- *L'interception d'identité*

---

[18]Nous avons décrit chacune de ces menaces dans la Partie I, Section 1.2.

[19]La description qui suit fait référence à des mécanismes qui seront décrits par la suite.

Le modèle de sécurité de l'annuaire n'offre pas de protection contre cette menace. En particulier, il doit être noté que si l'opération nécessite le chaînage, il devient difficile de déterminer les points finals de la communication.

- *Fausse identité*

  Les mécanismes d'authentification simple protégée et d'authentification forte visent à fournir une protection contre cette menace.

- *Re-jeu*

  Les mécanismes d'authentification simple protégée et d'authentification forte visent à fournir une protection contre cette menace.

- *Interception de données*

  Le modèle de sécurité de l'annuaire n'offre pas de protection contre cette menace. Dans la Section 10.2.2, nous proposons une solution à ce problème.

- *Manipulation de données*

  Le mécanisme d'authentification forte fournit une protection contre cette menace.

- *Rejet* ou *Répudiation*

  Le mécanisme d'authentification forte peut être utilisé pour fournir une protection contre cette menace.

- *Refus de service*

  Ce danger est très général et dépend de l'intention de l'interruption non autorisée. Le modèle de sécurité de l'annuaire n'offre pas de protection contre cette menace. Toutefois, une implémentation plus évoluée peut prendre en compte des variantes de ce danger, par exemple, un DSA peut refuser de réaliser des opérations qu'il considère excessivement coûteuses.

- *Acheminement erroné*

  Le modèle de sécurité de l'annuaire n'offre pas une protection directe contre cette menace. Cependant, l'utilisation du mécanisme d'authentification forte pendant le chaînage des opérations peut permettre la détection de ce type d'attaque.

- *Analyse de trafic*

  Le modèle de sécurité de l'annuaire n'offre pas de protection contre cette menace. Les dangers provenant de l'analyse de trafic ne concernent pas exclusivement une couche OSI déterminée. Toutefois, on peut assurer partiellement une protection contre cette menace par la production d'un trafic additionnel inintelligible (remplissage de trafic) en utilisant des données chiffrées ou aléatoires.

6.2.1.2.1. L'authentification et l'autorisation dans l'annuaire

La base d'information de l'annuaire est répartie entre de nombreux DSA, dont chacun en maintient une partie. L'accès à une entrée particulière se fait en contactant directement le DSA qui gère l'entrée, via DAP, ou en chaînant la requête entre DSA, via DSP, au nom du DUA demandeur.

Des contrôles doivent être effectués sur l'opération de l'annuaire, ceux-ci en termes d'authentification et d'autorisation. Une politique d'authentification est utilisée pour décrire comment les fournisseurs et les utilisateurs du service d'annuaire s'identifient les uns les autres afin d'acquérir le niveau de confiance nécessaire à l'accès au service. Les recommandations X.500 définissent différentes méthodes d'authentification.

Une politique d'autorisation doit alors être utilisée pour dériver les droits d'accès des entités authentifiées, afin de libérer ou refuser la réalisation d'un service ou l'accès à une entrée demandée. La version 1988 des standards X.500, qui correspond aux implémentations disponibles actuellement, ne définit aucun mécanisme permettant la mise en vigueur des politiques d'autorisation. Cependant, les spécifications de l'annuaire ont évolué dans ce sens. La version 1994 des normes X.500 contiendra un schéma de contrôle d'accès à travers lequel les serveurs d'annuaire pourront définir et exprimer leurs politiques d'autorisation et d'authentification [X.50093].

6.2.1.2.2. L'authentification durant l'association

Pour qu'une entité puisse établir une association avec un DSA, elle doit s'identifier auprès de ce serveur. Cette entité peut être un DUA ou un autre DSA. Les normes X.500 spécifient quatre niveaux d'authentification [X.511] :

* **pas d'authentification**

  L'initiateur fournit seulement son nom d'annuaire. Dans ce cas, il n'est pas possible d'authentifier l'initiateur. En pratique, ce type d'accès peut être vu comme équivalent à un accès anonyme.

* **l'authentification simple non-protégée**

  L'initiateur fournit son nom d'annuaire avec un mot de passe. Toutefois, ce mot de passe est fourni en clair, ce qui expose l'initiateur au danger d'interception d'identité, i.e., l'information d'identification de l'initiateur peut être facilement notée pour une postérieure utilisation non autorisée.

* **l'authentification simple protégée**

  L'initiateur fournit son nom d'annuaire, une estampille de temps, un nombre aléatoire. Le tout, avec le mot de passe de l'initiateur, est scellé au moyen d'une fonction de scellement. Le résultat de ce scellement est également fourni au DSA. Ce dernier pourra alors reproduire le scellement en utilisant une copie locale du mot de passe de l'utilisateur.

* **l'authentification forte**

  L'initiateur fournit un jeton signé au moyen de sa clé privée et, optionnellement, un chemin de certification qui permet au DSA de vérifier la clé publique de l'initiateur. Après avoir validé la clé publique de l'initiateur, le DSA peut l'utiliser pour vérifier la signature du jeton de manière à s'assurer de son origine et de son integrité.

Contrairement aux méthodes d'authentification simple, l'authentification forte ne requiert pas qu'un secret soit partagé entre l'initiateur et le DSA. En contrepartie, cette dernière exige

l'existence d'un point commun de confiance entre les deux parties pour la vérification des clés publiques.

Dans tous les cas, il est possible que les accréditations soient transférées non seulement de l'initiateur vers le DSA, mais aussi du DSA vers l'utilisateur, afin de permettre l'authentification mutuelle entre ces deux parties. De plus, la norme X.500 permet l'utilisation des accréditations définies de façon externe. Dans ce cas, tant la syntaxe que la sémantique du schéma d'authentification sont définies par accord bilatéral.

6.2.1.2.3. L'authentification des opérations

Les utilisateurs de l'annuaire ainsi que les autorités de gestion qui fournissent les services d'annuaire peuvent, à leur convenance, exiger l'authentification des opérations d'annuaire. Pour chacune de ces opérations, la nature de la procédure d'authentification dépend de la politique de sécurité en vigueur.

L'annuaire permet l'authentification du demandeur d'une opération ; il s'agit de l'identification de l'auteur de la demande initiale de service par un DSA. D'après l'identité de l'initiateur, le DSA peut concéder ou refuser l'accès à une entrée en particulier. L'annuaire définit deux procédures pour l'authentification du demandeur.

La première est fondée sur le simple transfert d'un nom d'annuaire en clair. En fait, cette procédure ne doit pas être considérée comme une procédure d'authentification car elle n'assure aucun niveau de sécurité.

La deuxième procédure est, en fait une application des protocoles d'authentification définis dans la cadre X.509 (voir 5.1.2). Cette procédure spécifie que toute opération, qu'elle soit directe ou chainée, peut être optionnellement accompagnée des paramètres de sécurité et d'une signature numérique. A titre d'exemple, l'opération de lecture est spécifiée en ASN.1 comme suit :

```
Read ::= ABSTRACT-OPERATION
    ARGUMENT    OPTIONALLY-SIGNED ReadArgument
    RESULT      OPTIONALLY-SIGNED ReadResult
    ERRORS      {AttributError, NameError, ServiceError,
                Referral, Abandoned, SecurityError}
```

Dans l'argument et le résultat de chaque opération (*ReadArgument* et *ReadResult*, par exemple) se trouve le champ "*SecurityParameters*" défini comme suit :

```
SecurityParameters ::= SET {
    certificationPath  [0]  CertificationPath OPTIONAL,
    name               [1]  DistinguishedName OPTIONAL,
    time               [2]  UTCTime OPTIONAL,
    random             [3]  BIT STRING OPTIONAL,
    target             [4]  INTEGER {none (0), signed (1)}
}
```

Où,

- Le composant *certificationPath* se compose du certificat de l'expéditeur et, en option, d'une séquence de certificats de CA permettant de valider le certificat de ce premier.
- Le composant *name* est le nom d'annuaire du premier destinataire prévu de l'argument ou du résultat.
- Le composant *time* est la date d'expiration prévue pour la validité de la signature.
- Le composant *random* est un nombre aléatoire qui doit différer pour chaque jeton encore valide. Il est utilisé conjointement avec le composant *time* pour permettre la détection des attaques de répétition.
- Le composant *target* n'apparaît que dans les demandes d'exécution d'opération et sert indique si le résultat correspondant doit être signé.

Conjointement, ces paramètres et la signature permettent d'authentifier l'origine des données, de garantir l'intégrité des données transférées et de détecter des attaques par répétition. Pour chaque requête, il est possible de demander que le résultat soit protégé. Ainsi, théoriquement, l'annuaire facilite l'authentification des résultats. Il s'agit de l'authentification, par le demandeur, de tous les résultats qui lui sont retournés. Le degré de protection effectivement appliqué au résultat est indiqué par la forme du résultat et peut être égal ou inférieur à celui demandé, en fonction des restrictions de l'annuaire. Pour l'authentification des résultats, l'annuaire définit seulement l'authentification forte, i.e., basée sur les paramètres de sécurité et la signature du DSA exécutant de l'opération.

Ces mécanismes d'authentification forte son applicables tant au niveau de l'accès à l'annuaire (le protocole DAP) qu'au niveau de son opération répartie (le protocole DSP). Nous pouvons identifier trois contextes différents :

(1) La signature du demandeur initial (l'utilisateur) est transférée à travers tous les DSA qui participent au chaînage, jusqu'au DSA capable de satisfaire la requête. Cela permettrait que ce DSA puisse authentifier l'auteur de la requête et en suite consulter la politique d'autorisation pour décider s'il doit ou non réaliser l'opération.

(2) D'autre part, la signature du DSA qui a pu satisfaire entièrement ou partiellement une requête peut être transférée à travers tous les DSA participant à opération distribuée, jusqu'à l'usager demandeur du service. Cela permettrait que l'utilisateur puisse authentifier les sources des résultats.

(3) Lorsque plusieurs DSA fournissent des résultats partiels à une requête, chacun de ces résultats partiels est accompagné de la signature du DSA concerné.

(4) Chaque opération chaînée entre deux DSAs peut être signée par le DSA initiateur. Cela constitue une mesure additionnelle pour assurer que seuls des DSA de confiance (ou autorisés) participent à l'opération distribuée et pour garantir l'intégrité des données propres à opération distribuée de l'annuaire. Ce mécanisme s'applique tant aux requêtes qu'aux résultats de l'opération repartie.

L'authentification des messages d'erreur n'est pas prise en charge par ces procédures. Cela peut être considéré comme une lacune du modèle de sécurité de l'annuaire, car un intrus peut remplacer un résultat, même signé, par une réponse d'erreur sans que cela ne soit détecté. Cela constitue un risque de refus de service qui pourrait être contourné si les réponses d'erreur pouvaient être signés.

6.2.2. La Messagerie Electronique X.400

X.400 est un standard conçu dans le but de fournir un service de messagerie électronique, à grande échelle, sans restrictions sur le type des informations codées véhiculées. Cette norme a été proposée en 1984 par le CCITT sous forme d'une série de recommandations définissant un système de messagerie en mode "enregistrement et retransmission" (store and forward). L'architecture du système de messagerie X.400 [X.400] est décrite sous forme d'un modèle MHS (**M**essage **H**andling **S**ystem) dont la structure est illustrée par la Figure 12.

La version 1984 des Recommandations X.400 définissent deux types entités de base, à savoir, les Agents Utilisateurs (UA) et les Agents de Transfert de Messages (MTA). Dans ce modèle, un utilisateur peut être un individu ou une application. L'interface entre et le système de messagerie est assuré par un UA, qui l'aide à composer et à émettre des messages, en le soumettant à un MTA. Les messages sont transmis via un ou plusieurs MTA, qui coopèrent en mode store-and-forward, pour acheminer les messages depuis un UA expéditeur jusqu'à l'UA destinataire. L'ensemble des MTA forme collectivement ce qui est connu comme le Système de Transfert de Messages (MTS).

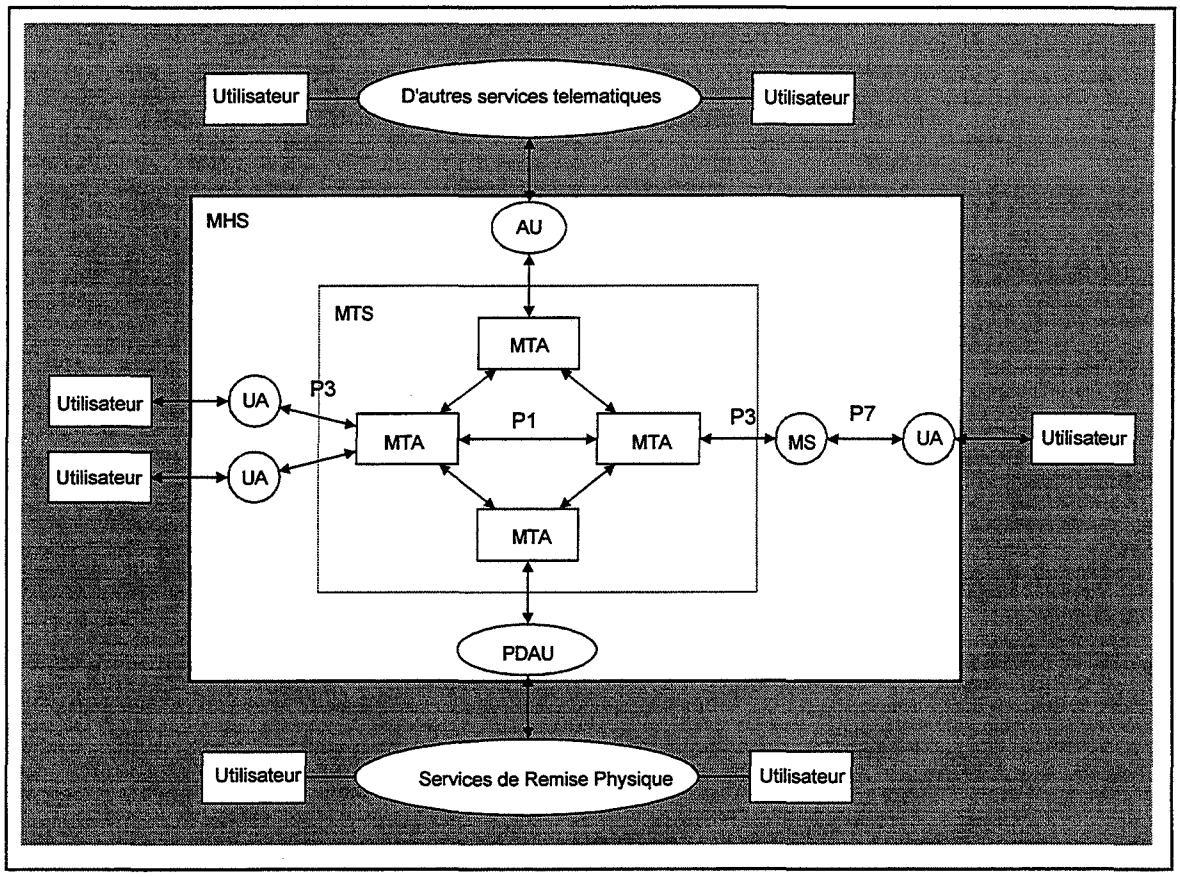

Figure 12. Modèle Fonctionnel du MHS

Les Recommandations X.400 définissent également les protocoles qui gouvernent la communication entre les composants d'un MHS. Le protocole P1, le Protocole de Transfert de Messages, définit le transfert de messages entre des MTA. Le protocole P3, constitue le Protocole d'Accès au MTS et est donc utilisé pour la soumission et la remise des messages entre UA et MTA. Finalement, le protocole P2 définit un service de courrier interpersonnel au-dessus du

MTS, en spécifiant le format de messages échangés entre les utilisateurs finaux. D'autres formats de messages sont définis pour supporter différentes applications comme l'EDI (**E**lectronic **D**ata **I**nterchange) [X.435].

Un message est composé d'une enveloppe dans laquelle se trouvent des informations de contrôle (par exemple, pour l'acheminement du message) et d'un contenu contenant le message proprement dit.

Le standard a été étendu en 1988 de manière à apporter des corrections à des déficiences de la version originelle et pour ajouter de nouvelles fonctionnalités. Un exemple de ces nouvelles caractéristiques est la définition d'une nouvelle entité du MHS, à savoir, le **M**essage **S**tore (MS). Le MS est un composant placé entre l'UA et le MTA de manière à jouer le rôle de boite aux lettres privée. L'existence d'un tel composant est essentielle dans le but d'étendre l'utilité de la messagerie X.400 au contexte des communications mobiles.

Un autre trait de l'évolution des normes X.400 est la définition des éléments de service de sécurité. Ces nouveaux éléments de service ont été conçus pour permettre la fourniture d'une grande variété de services de sécurité, tant au niveau des utilisateurs finaux que des services orientés vers l'association entre les entités du MHS.

6.2.2.1. Les Services de Sécurité

Avant de dépeindre les services de sécurité disponibles pour la messagerie X.400, il semble utile de considérer les possibles menaces aux systèmes de messagerie électronique. Celles-ci incluent : l'usurpation d'identité, le rejeu et le mise en désordre (re-sequencing) de messages, la modification et la lecture non-autorisée de messages, la répudiation d'origine et de remise des messages, le refus de service et la perte d'information.

Bien entendu, il n'est pas possible de faire face à tous ces dangers au niveau seulement de la couche application. Par exemple, l'analyse de trafic et la perte d'information auraient lieu dans le cadre de la surveillance des volumes de trafic entre deux noeuds du réseau. Cela pourrait avoir lieu même dans le cas où le contenu des messages est chiffré. La protection contre ces menace nécessite des services de sécurité au niveau des couches basses du modèle OSI, ce qui est hors de portée des services d'application.

De nombreux services de sécurité peuvent être fournis à un MHS. De façon générale, la sécurité de la messagerie comporte deux aspects qui sont la sécurité de messages et la sécurité de l'administration (et de la gestion d'accès).

En ce qui concerne la sécurité de messages, ces services peuvent être applicables de bout-en-bout, i.e., entre des utilisateurs du MTS. Dans ce cas, la réalisation des services ne requiert pas la participation active du MTS. D'autre part, les services de sécurité peuvent être appliqués par le MTS pour sécuriser la structure de la messagerie proprement dite.

Quant à la sécurité de l'administration et de la gestion d'accès, les services concernés peuvent requérir la participation active et la coordination des utilisateurs du MTS et des composants du MTS. Ainsi, ces fonctions peuvent être appliquées à n'importe quelle paire de composants du MHS, comme, entre UA ↔MTA, UA ↔ MS, MS ↔ MTA, MTA ↔ MTA.

Sept groupes de services sont considérés par le modèle de sécurité du MHS à travers les recommandations X.400 :

- Les *services d'authentification d'origine* permettent l'authentification de l'identité des parties communicantes et des sources de données. Parmi ces services nous identifions :
  - ◊ les services d'authentification de l'origine de messages, de demandes d'essai et d'avis. Ceux-ci permettent d'authentifier, respectivement l'expéditeur d'un message, d'une demande d'essai et d'un avis. Dans le cas de l'authentification de l'origine de messages, les normes X.400 spécifient deux façons différentes d'implémenter ce service de manière à ce qu'il puisse être fourni seulement de bout-en-bout, i.e., entre l'expéditeur et les destinataires du message; ou de façon extensible à tous les composants du MHS.
  - ◊ les services de preuve de soumission et de remise de messages. A travers ces services l'expéditeur d'un message peut s'assurer qu'il a été soumis au MTS et qu'il a été remis aux destinataires prévus.
- Les *services d'intégrité* de données visent à protéger les données transitant dans le MHS contre la modification non-autorisée. Dans ce contexte, deux services peuvent être fournis au moyen des mécanismes disponibles dans le modèle de sécurité de X.400 :
  - ◊ le service d'intégrité des contenus de messages permet au(x) destinataire(s) d'un message de détecter une possible modification d'un message pendant le transfert,
  - ◊ le service intégrité de séquence de messages permet de détecter une éventuelle mise en désordre d'une séquence de messages.
- Les *services de non-répudiation* visent à fournir une preuve irréfutable de l'origine, de la soumission ou de la remise d'un message, de telle sorte que cela puisse être vérifié par une tierce partie, par exemple un notaire, à n'importe quel moment.

  Cependant, il doit être noté que modèle de sécurité de X.400 ne fournit pas de mécanismes spécifiques à la non-répudiation. En fait, ce service est fourni à travers la combinaison d'autres services, à savoir, les services d'authentification d'origine et d'intégrité de données.

  De plus, pour pouvoir fournir ce service, la politique de gestion de clé doit prendre ce propos en compte explicitement.
- Les *services de confidentialité* visent à protéger des données du MHS contre la divulgation non autorisée. Dans ce contexte, le modèle de sécurité de la messagerie X.400 définit trois services : la confidentialité de contenus de messages, des associations et des flux de messages.

  Il faut rappeler que le modèle de sécurité de X.400 ne permet pas de fournir le service de confidentialité d'associations par ses propres moyens. Toutefois, il est possible d'obtenir les données cryptographiques nécessaires à la fourniture de ce service, comme résultat de l'authentification au moment de l'établissement de l'association. Par exemple, par l'établissement d'une clé de session permettant le chiffrement de toute une association entre deux composants du MHS.

  De même, le service de confidentialité des flux de messages ne peut pas être entièrement fourni en utilisant seulement les mécanismes disponibles dans le modèle de sécurité de X.400. Cela nécessite l'intégration des mécanismes de sécurité, tel que le remplissage de trafic, dans des couches basses de la pile OSI.

- Les *services de sécurité de gestion d'accès* visent à fournir un niveau plus fort de protection des ressources du MHS contre l'utilisation non autorisée. Pour cela, deux services sont définis :
  - ◊ le service d'authentification des entités homologues permet la confirmation de l'identité des parties au moment de l'établissement d'association, de manière à vérifier qu'aucune des parties n'utilise une fausse identité ni ne rejoue des données d'une association précédante,
  - ◊ le service de contextes de sécurité vise à contrôler le passage de messages entre les entités du MHS à travers la consultation des "étiquettes de sécurité" associées aux messages. Cela peut être utilisé pour construire des politiques d'autorisation pour l'utilisation de ressources du MHS.
- Le *service d'étiquetage* permet d'associer des étiquettes de sécurité à toutes les entités dans le MHS, i.e., les MTA, les utilisateurs du MTS et les messages. En conjonction avec le service de contextes de sécurité, cela permet l'implantation de politiques de sécurité qui définissent quelles parties du MHS sont autorisées à manipuler des messages contenant une certaine étiquette. Les mécanismes disponibles dans le modèle de sécurité X.400 permettent de garantir l'intégrité, et optionnellement la confidentialité de ces étiquettes.
- Les *services d'administration de la sécurité* permettent l'établissement et la maintenance de paramètres de sécurité auprès des entités du MHS. Les services disponibles concernent la modification des accréditations retenues par des entités du MHS, précisément les MTA et les MS, l'établissement et la modification des étiquettes de sécurité auprès des MTA et des MS.

6.2.2.2. Les Mécanismes de Sécurité

Les mécanismes spécifiés dans les protocoles X.400 pour implémenter les services de sécurité sont basés sur l'approche X.509 tant au niveau de la gestion des clés qu'au niveau des protocoles d'authentification. Ainsi, le modèle de sécurité de X.400-88 suppose l'utilisation de certificats pour la distribution de clés publiques et définit un format de chemin de certification propre à ses besoins appelé "*Certificates*". La description ASN.1 de cette structure sera presentée dans la Section 8.4.1.

Les services de sécurité X.400 sont réalisés à travers la définition de signatures et de jetons d'authentification. Les signatures correspondent à des "*authentication checks*" qui peuvent être vérifiées par toutes les entités du MHS à travers lesquelles le message transite.

En ce qui concerne jetons d'authentification, X.400 en définit deux types, le jeton d'association et le jeton de messages. Le premier sert à sécuriser l'accès aux entités du MHS. Quant aux jetons de message, chacune de ses instances est destinée à un seul destinataire d'un message. Ces structures de données sont utilisées pour fournir les services de sécurité de bout-en-bout.

Le format de jetons est inspiré de l'approche X.509. Les jetons d'authentification X.400 contiennent le nom d'O/R du destinataire, la date et l'heure de génération du jeton et les champs "*signed-data*" et "*encrypted-data*" qui varient selon le service fourni. Le champ "signed-data" comporte des données spécifiques au type de jeton. Le champ "encrypted-data" est chiffré au

moyen de la clé publique du destinataire pour permettre de lui envoyer des données confidentielles. Ensuite, l'emsemble du jeton est signé au moyen de la clé privée de l'expéditeur.

```
AsymetricToken ::= SIGNED SEQUENCE {
        signature-id        AlgorithmIdentifier,
        recipient           ORName,
        time                UTCTime,
        signed-data         [0] TokenData OPTIONAL,
        encryption-id       [1] AlgorithmIdentifier OPTIONAL,
        encrypted-data      [2] TokenData OPTIONAL
}
```

X.400 utilise la notion de MACRO ASN.1 pour permettre la définition de types de données spécifiques au type de jeton. Dans le cas du jeton d'association, ces données sont constituées d'un nombre aléatoire pour protéger contre les attaques de répétition. Pour les jetons de message ces données contiennent, entre autres, des étiquettes de sécurité, un numéro de séquence et une signature permettant d'authentifier l'origine du message et de vérifier son intégrité.

Dans la Partie IV de ce travail, nous présentons des détails supplémentaires sur les mécanismes de sécurité de X.400, ainsi qu'une implémentation de ces mécanismes.

PARTIE III

EXPERIMENTATION

---

7. **LE PROJET PASSWORD**
8. **SECUCOMX : UNE ARCHITECTURE DE SECURITE POUR LES SYSTEMES OUVERTS**
9. **LA SECURISATION DES APPLICATIONS**
10. **DES PROBLEMES OUVERTS**

Cette partie du présent travail porte sur notre expérimentation dans le domaine de la sécurité pour les systèmes ouverts. Notre participation dans **Password**, un projet européen de recherche, nous a permis de prendre connaissance des besoins en sécurité de ces systèmes et des mécanismes pour y répondre. Nous allons donc présenter, dans le premier chapitre de cette partie, les points abordés dans ce projet.

Notre contribution personnelle a été la conception et le développement d'une architecture de sécurité, baptisée **SecUcomx**, dont l'objectif est de proposer des solutions aux problèmes actuels concernant la sécurité pour les applications ouvertes, et de fournir une infrastructure de sécurité permettant l'implémentation de toute une gamme de services de sécurité. Dans le deuxième chapitre de cette partie, nous décrirons cette architecture. Le troisième chapitre porte sur l'utilisation des éléments de cette architecture pour la sécurisation des applications.

Ce travail nous a permis d'expérimenter l'approche X.509 et d'en détecter les obstacles à la mise en place d'une infrastructure commune pour les services d'authentification. Dans le quatrième chapitre nous soulèverons ces problèmes.

CHAPITRE 7
LE PROJET PASSWORD

Password est un acronyme pour "**P**iloting **A**uthentication and **S**ecurity **S**ervices **w**ithin **O**SI Applications for the **R**esearch and **D**evelopment Community in Europe". Ce projet regroupe des équipes de recherche du monde académique et industriel de trois pays Européens, collaborant pour définir et déployer une infrastructure de sécurité pour la communauté R&D.

Le projet a démarré en Janvier 1992, dans le cadre du Programme CEC VALUE, pour se terminer en Juin 1994. L'équipe du projet est composée de trois consortiums distribués en France, Allemagne et Angleterre, avec les partenaires suivants: INRIA et TS-E3X en France, UCL, Cambridge University et NeXor en Angleterre, GMD et DANET en Alemagne.

L'objectif du projet est d'élaborer une infrastructure commune d'authentification basée sur l'approche X.509 afin de valider cette approche et d'en tirer une expérience. Pour atteindre cet objectif, Password vise à introduire et piloter le déploiement de services sécurisés pour les principales applications OSI, à savoir :

- l'annuaire X.500,
- la messagerie X.400,
- l'ODA (**O**pen **D**ocument **A**rchitecture),
- ainsi que d'autres applications orientées connexion, à travers l'une des unités fonctionnelles de l'ACSE (**A**ssociation **C**ontrol **S**ervice **E**lement).

En outre, le projet est fortement lié aux activités menées au sein de l'Internet Engineering Task Force (IETF) en ce qui concerne la compatibilité avec l'approche PEM.

Ce projet envisage d'introduire ces services pilotes sécurisés dans des sites connectés au réseau européen de la recherche. Afin d'assurer un ensemble raisonnablement ouvert de services, le projet offre diverses implémentations de l'infrastructure d'authentification et de la majorité des services sécurisés. En développant toute une gamme de services sécurisés à partir de différentes implémentations, les acteurs du projet cherchent à acquérir une compréhension réelle des

exigences liées aux services de contrôle d'accès, d'authentification, d'intégrité et de confidentialité de données pour la communauté R&D.

Password envisage de démontrer que ces services pilotes peuvent être déployés sans nuire à l'ouverture et à la fonctionnalité des systèmes. Enfin, ce projet a pour objectif de démontrer que cette architecture permet l'usage d'un ensemble commun de services sécurisés pour les applications mentionnées plus haut

Durant le projet Password, les besoins en sécurité ont été identifiés et évalués tant au niveau des services qu'au niveau des utilisateurs. L'un des résultats de ce travail d'évaluation a été la publication d'une série de documents décrivant non seulement ces besoins mais aussi les directives pour la mise en place d'une structure commune pour la fourniture des services de sécurité dans le cadre des systèmes ouverts [Roe92, Roe93, RSK92, Lub93, Lue93, LW93, KSW92]. Ces directives comprennent le choix des algorithmes et des protocoles appropriés, ainsi que les critères d'implémentation nécessaires lors de la mise en œuvre des standards de façon à ce que cela n'empêche ni altère l'interfonctionnement entre les divers systèmes.

Ce projet représente l'une des premières tentatives de pilotage à grande échelle d'un schéma de distribution de clés basé sur l'avis X.509. Les résultats d'un tel effort offriront une base pour des projets futurs et permettront la révision des recommandations existantes. A l'heure actuelle, le projet se trouve dans sa phase pilote dont les principaux objectifs sont :

- déployer les applications sécurisées,
- vérifier "l'utilisabilité" des services de sécurité,
- observer l'impact de l'intégration de la sécurité sur les services existants.

Dans la suite de ce chapitre nous décrirons les aspects abordés dans le projet Password. Dans le chapitre suivant nous présenterons notre contribution personnelle dans le projet. Nous soulèverons également des problèmes non résolus dans le cadre du projet.

7.1. L'Infrastructure de Sécurité

L'un des résultats du projet Password est le développement de différentes implémentations de la technologie de base avec le but de faciliter la mise en place des solutions différentes, tout en gardant l'interopérabilité. Chaque consortium du projet a développé une infrastructure de sécurité permettant l'implémentation des applications sécurisées, dont nous pouvons citer *SecuDE* [Gri91] dans le consortium Allemand et *OSISEC* [Wil92] dans le consortium Anglais.

Notre contribution personnelle, dans le consortium français, a été la conception et le développement de toute une architecture de sécurité, à savoir **SecUcomx**, ainsi que l'intégration des éléments de cette architecture pour la sécurisation des applications. Dans le Chapitre 8, nous décrirons l'architecture de sécurité SecUcomx et, dans le Chapitre 9, l'utilisation de cette architecture pour le développement des applications sécurisées.

7.1.1. Les Techniques Cryptographiques

La façon d'aborder les services de sécurité par l'OSI comprend la combinaison de techniques de chiffrement symétriques et asymétriques. Puisque les méthodes symétriques sont moins coûteuses, elles sont utilisées pour le chiffrement/déchiffrement de quantités importantes de données. En contrepartie, des quantités réduites de données, telles que des clés symétriques ou des résultats de scellement (*check-sums*), peuvent être efficacement chiffrées au moyen d'un crypto-système asymétrique.

Dans des environnements ouverts, les procédures d'authentification sont plus fiables et plus faciles à gérer en utilisant des crypto-systèmes asymétriques car ceux-ci n'exigent pas le partage de secrets. Les services dérivés de telles procédures sont alors implémentés en ajoutant des certificats aux données signées. Ceux-ci facilitent l'authentification de la clé publique du signataire qui permettra, ensuite, la validation de la signature.

Afin d'offrir un ensemble de routines fournissant ces fonctionnalités de base, chaque consortium du projet Password a développé une boîte à outils de sécurité (*security toolkit*), i.e., une bibliothèque de fonctions utilisée par les diverses applications et contenant les fonctionnalités de base suivantes :

- des crypto-systèmes symétriques, tels que l'algorithme DES [ANSI81, ANSI83],
- des crypto-systèmes asymétriques, tels que l'algorithme RSA [RSA78],
- des fonctions de scellement à sens unique, ex., les algorithmes MD2 [Riv90] et MD5 [Riv92],
- des fonctions de génération/vérification de signatures numériques,
- des fonctions de manipulation des objets de sécurité nécessaires à l'implémentation du cadre d'authentification X.509 [X.509], i.e., des structures de données telles que des certificats, des chemins de certification et des listes de révocation de certificats, ainsi que les macros ASN.1 SIGNED, SIGNATURE, ENCRYPTED et ALGORITHM.

7.1.2. Les Mécanismes de Base

Comme nous l'avons déjà mentionné dans la Section 5.1, la Recommandation X.509 définit un cadre d'authentification afin de fournir une base pour le développement des applications sécurisées. Ce cadre présente deux niveaux d'authentification : un très simple et faible, utilisant des mots de passe, et un autre plus fort utilisant des techniques cryptographiques.

L'authentification forte repose sur le concept de signatures numériques. Les protocoles d'authentification forte du cadre X.509 sont basés sur l'échange de jetons signés par son expéditeur. La signature prouve l'intégrité et l'origine du jeton. Ces jetons sont accompagnés par un chemin de certification qui permet de valider la clé publique de l'expéditeur. Celle-ci, à sont tour, permettra la verification de la signature du jeton. Les procédures de challenge/réponse constituent des techniques conventionnelles permettant de déterminer l'identité des parties

communicantes à l'initialisation de la connexion [DS81]. Cette technique est utilisée, par exemple, pour sécuriser l'accès à l'annuaire [X.511].

Une variante du jeton d'authentification X.509 est utilisée dans les mécanismes de sécurité définis pour la messagerie X.400 [X.411]. Dans ce cas, les jetons sont utilisés non seulement pour authentifier l'origine d'un message mais aussi pour transférer la clé symétrique nécessaire au déchiffrement d'un message confidentiel.

Actuellement, les contenus des messages X.400 sont protégés au moyen de l'algorithme DES en mode ECB (**E**lectronic **C**ode **B**ook) [ANSI83]. D'autres modes de cet algorithme seront adoptés à court terme dans le projet, à savoir, le CBC (**C**ipher **B**lock **C**haining) et le CFB (**C**ipher **F**eed**B**ack) [ANSI83].

Les mécanismes de sécurité demandent la distribution/récupération prompte des données d'authentification associées à l'expéditeur et au destinataire des informations. Dans ce contexte, un avantage du cadre X.509 est la possibilité de conserver des certificats et des listes de révocation dans l'annuaire sous la forme d'attributs et de pouvoir les récupérer comme n'importe quelle autre information d'annuaire. Dans la section suivante nous décrivons la façon d'aborder la gestion de clés dans Password.

7.2. Gestion de Clés et de Certificats

Le niveau d'assurance fourni par un mécanisme de sécurité basé sur le chiffrement dépend fortement de la sécurité des clés utilisées. Dans le cas de clés symétriques, afin d'assurer un haut niveau de sécurité, elles doivent être changées fréquemment. Par exemple, ces clés peuvent être générées de façon aléatoire à chaque session entre des entités. Les clés asymétriques, au contraire, sont moins vulnérables et donc possèdent, en principe, une durée de vie plus longue.

Password n'adresse pas le problème général de la sécurité des stations de travail, mais considère que les clés privées doivent être maintenues confidentielles à toute entité autre que leur légitime propriétaire. Un accès non-autorisé à ces clés pourrait détruire tout le modèle de sécurité. Par exemple, une partie considérable de SecuDE s'attaque au problème de la protection de clés privées et envisage différentes approches, y compris l'utilisation de "cartes à puces". SecUcomx utilise un émulateur de carte à puces résidant dans l'environnement de l'utilisateur, inspiré de Chimæra [Alt92]. OSISEC conserve les clés privées d'utilisateur dans des fichiers chiffrés symétriquement.

Comme nous l'avons déjà mentionné, Password a adopté l'approche X.509 pour la distribution des clés publiques. Cela implique l'élaboration de mécanismes pour la gestion de certificats, y compris la mise en œuvre des Autorités de Certification pour l'émission et l'administration de ces certificats. Le cadre X.509 comporte plusieurs alternatives pour la production des paires de clés en rapport avec le transport subséquent du composant public pour la certification. L'équipe du projet Password a étudié ces alternatives [Roe92] mais n'impose pas une technique unique.

Tout un travail a été mené pour identifier les exigences devant être satisfaites pour offrir un service de certification et de distribution de clés fiable [Roe92]. A partir de ce travail, chaque consortium a élaboré des implémentations différentes pour les services communs suivants :

* des systèmes de génération de paires de clés asymétriques,

* des mécanismes de distribution/conservation sécurisée de clés privées,
* des Autorités de Certification implémentant des mécanismes de certification et de révocation de certificats.

Le déploiement de services de sécurité à grande échelle exige un mécanisme décentralisé pour le stockage et la récuperation de certificats. Dans le cadre du groupe de travail PEM, cette distribution est faite à travers l'échange de messages. Ce mécanisme est très orienté vers la communication interpersonnelle. Cela n'est donc pas convenable à l'authentification en temps réel, qui ne doit pas être basée sur de tels échanges préalables. La méthode la plus appropriée dans le contexte Password est l'utilisation d'un service de noms, par exemple l'annuaire X.500. De plus, la communauté R&D européenne dispose d'un Service d'Annuaire reparti, coordonné par le projet CEC Value PARADISE [Goo91], qui rejoint des projets pilotes nationaux du même service.

Tous les consortiums Password disposent de systèmes d'annuaire opérationnels dans le cadre de Paradise. En particulier, les consortiums supportent les attributs X.500 qui permettent le stockage de certificats. Les certificats de tous les participants du projet peuvent être récupérés à partir de l'annuaire X.500. Les tests d'interopérabilité ont permis la correction d'erreurs et d'incompatibilités dues à la nature hétérogène de ces deux projets. Ainsi, des certificats tout à fait compatibles sont conservés sous des plates-formes hétérogènes.

Comme nous le présenterons dans la Section 10.1.3, le format actuel de listes de révocation proposé par le cadre X.509 est redondant et manque de précision. Le projet Password contourne les déficiences du format des listes de révocation défini par la norme X.509 en adoptant le format proposé par l'approche PEM [Ken93a]. Bien entendu, chaque consortium supporte le format de listes X.509 de façon à maintenir la conformité vis-à-vis de la structure Paradise.

7.2.1. Le Modèle de Certification

Du fait de l'utilisation de crypto-systèmes asymétriques, le déploiement de services d'authentification à grande echelle exige une disponibilité répandue de clés publiques et un mécanisme décentralisé pour la distribution des certificats.

Afin de fournir un niveau de sécurité adéquat en préservant la conformité avec les standards, il est nécessaire d'établir une politique de certification[20]. Celle-ci correspond à la définition de l'emplacement, du rôle et de la juridiction des autorités de certification dans le système.

L'objectif premier d'une politique de certification est de garantir qu'un certificat ne sera émis que si le composant privé correspondant n'est connu que par son propriétaire. Les autres objectifs sont de limiter les conséquences d'éventuelles actions malveillantes de la part des opérateurs de CA, et de garantir que ceux-ci rendront compte de leurs actions. Puisque les CA sont des entités de confiance, la politique de certification doit limiter la possibilité d'un abus de pouvoir. Ceci se produit lorsqu'une entité se sert de son autorité pour réaliser une fonction non autorisée. Un

[20]Le concept de politique de certification sera détaillé dans la Partie IV de ce travail.

danger similaire est celui de l'abus de confiance. Ceci aurait lieu lorsqu'une entité se sert de la confiance qui lui a été accordée pour réaliser une action malveillante.

Il est donc nécessaire de contrôler et d'imposer des restrictions aux autorités de certification et aux utilisateurs finaux. Cependant le cadre X.509 n'offre pas des moyens pour la définition explicite de telles règles. Une solution consisterait à prescrire des conventions de nommage à ces entités. Toutefois, ces restrictions peuvent rendre plus difficile l'utilisation et l'administration du système. La politique de sécurité doit alors pouvoir gérer ces deux extrêmes pour trouver un compromis acceptable entre le besoin de sécurité et le besoin de gestion, étant donnés les standards existants et l'état de l'art actuel.

7.2.1.1. Le Schéma Password

Le schéma proposé impose des règles au système de certification qui ne sont pas exigées par le cadre X.509 ni par la technologie elle-même. Ces règles sont motivées par plusieurs raisons, notamment le besoin de définir explicitement des sémantiques d'authentification qui permettent l'automatisation de la validation et de la détermination des politiques sous lesquelles les certificats sont émis. Les règles générales de nommage du système de certification Password sont les suivantes :

- tant les CA que les utilisateurs doivent être identifiés par des noms d'annuaire non ambigus,
- seules des organisations peuvent jouer le rôle de CA,
- une CA n'étant pas un individu, son nom d'annuaire ne doit pas inclure des attributs du type "*common name*" ou "*title*" [X.520],
- l'attribut "*country*" [X.520] doit être présent dans le nom d'annuaire des CA pour indiquer le pays où celle-ci est localisée,
- des informations supplémentaires (telles que localité, unité organisationnelle, etc.) doivent être présentes, si elles sont nécessaires à l'identification des CA,
- les noms des utilisateurs doivent être subordonnés aux noms de leurs CA. Cela sert à définir la juridiction des CA dans le système de certification.

Le groupe Password a défini un modèle hierarchique comprenant trois catégories de CA :

* les *CA Top Level* (TLCA),
* les *Policy CA* (PCA),
* les *CA Organisationelles* (CA).

Le modèle permet à une organisation d'offrir le service d'une ou plus de ces catégories de CA en utilisant un même support matériel. Toutefois, l'autorité doit distinguer ces catégories en utilisant un nom et une clé publique différents pour chacune.

La mise en vigueur des règles de certification n'est pas réalisée par le modèle lui-même, mais par les destinataires des certificats, i.e., un certificat invalide doit être refusé par les utilisateurs qui le reçoivent. Cependant, les CA doivent respecter leur obligations afin d'émettre des certificats dignes de confiance.

Les CA Organisationelles

Une CA Organisationnelle certifie qu'un utilisateur appartient à une certaine organisation. La juridiction d'une CA de cette catégorie est limitée à une seule organisation. Lorsqu'un même opérateur désire offrir ce service à plus qu'une organisation, il doit utiliser des noms et des clés publiques distincts pour différencier ces rôles. De grandes organisations peuvent requérir diverses CA Organisationnelles pour les différentes parties de l'organisation. Dans un tel cas, les CA de chaque unité organisationnelle peuvent se faire certifier directement par une PCA. Alternativement, la PCA peut certifier une CA Organisationnelle, qui, à son tour, certifiera les CA subordonnées afin de leur déléguer son autorité.

Une CA Organisationnelle doit avoir le nom d'annuaire de l'organisation qu'elle sert. La juridiction d'une CA de cette classe est définie par le sous-arbre dont le nom d'annuaire de la CA constitue la racine. Autrement dit, une CA Organisationnelle ne peut certifier que des entités dont le nom d'annuaire est subordonné au sien.

Afin de différencier les certificats d'utilisateurs des certificats de CA Organisationnelle, le dernier composant d'un nom de CA doit être un attribut du type *organisationalName* ou *organisationalUnitName* [X.520], tandis que le dernier composant du nom d'un utilisateur ne peut être aucun de ces attributs.

Les Policy CA

Une Policy CA certifie qu'une CA Organisationnelle opère en conformité avec une politique de sécurité spécifique et qu'elle est autorisée à certifier au nom d'une organisation particulière. Le modèle admet plusieurs PCA afin de disposer de CA ayant des différents niveaux d'assurance et pouvant émettre de certificats sous des politiques diverses. Par exemple, dans le contexte de la messagerie interpersonnelle, certains utilisateurs peuvent requérir un niveau d'assurance peu élevé en échange d'un service moins cher. A l'inverse, des systèmes bancaires et militaires peuvent demander un service de haute assurance, et probablement ne seront pas prêts à se fier à des PCA commerciales.

Chaque PCA doit rendre publique sa politique de sécurité. Cela comprend le niveau d'assurance de ses composants logiciels/matériels et la rigueur de ses procédures de certification. Lorsqu'une PCA certifie une CA Organisationnelle, elle doit s'assurer que les composants et les procédures de certification de celle-ci sont conformes à sa politique de sécurité.

Bien que les PCA puissent être nommées n'importe où dans le DIT, il est recommandé que le dernier composant d'un nom d'une PCA soit un attribut du type *organisationalName* ou *organisationalUnitName*. Les PCA ont le droit de certifier des CA Organisationnelles, indépendamment de leur position relative dans le DIT. Cependant, une PCA ne doit pas certifier d'autres PCA. Cette restriction est due au fait que l'attribution d'un certificat par une PCA implique une délégation d'autorité sous une certaine politique et que chaque PCA englobe une politique de sécurité et un niveau d'assurance particulier ; la certification entre PCA résulterait donc en des certificats ambigus. De plus, le format des certificats X.509 ne permet pas d'indiquer le type d'entité certifiée.

Bien qu'il soit possible de distinguer entre un utilisateur et une CA Organisationnelle à travers la convention définie pour leurs noms d'annuaire, le modèle ne fournit pas un mécanisme d'identification des PCA.

Les CA Top Level

Les CA Top Level certifient les PCA et ne se certifient pas entre elles. Elles servent à simplifier la tâche de la distribution de clés lorsque le nombre de PCA devient important. Il peut avoir plusieurs TLCA offrant différents niveaux d'assurance ou servant à des différentes communautés.

Par exemple, chaque pays pourrait opérer sa propre TLCA, et chacune certifierait les PCA de son pays et celles des autres, selon le niveau d'assurance et les politiques de sécurité exigés. Cela permettrait la fourniture de services sécurisés entre différents pays sans porter atteinte à la sécurité d'un pays en particulier. En contrepartie, un accord pour qu'un seul pays ou organisation opère une TLCA unique résulterait en un transfert d'autorité difficile à révoquer.

Les TLCA constituent les racines de la hiérarchie de certification. Ainsi, un utilisateur qui exige un niveau d'assurance particulier, doit alors obtenir la clé publique de la TLCA concernée.

7.2.1.2. La Structure du Projet

Dans le scénario défini pour Password, chaque consortium opère sa propre TLCA. Chacune certifie toutes les PCA du projet. Cela implique la connectivité globale entre tous les consortiums, bien que cela ne soit pas une exigence du schéma de certification. Par exemple, d'autres TLCA qui ne désirent pas avoir la connectivité avec tout le groupe, peuvent joindre la hierachie en certifiant seulement les PCA de leur choix.

De plus, puisque chaque pays opère sa propre TLCA, chacun des pays a la possibilité de résilier l'accord avec un autre en révoquant les certificats émis pour les PCA étrangères.

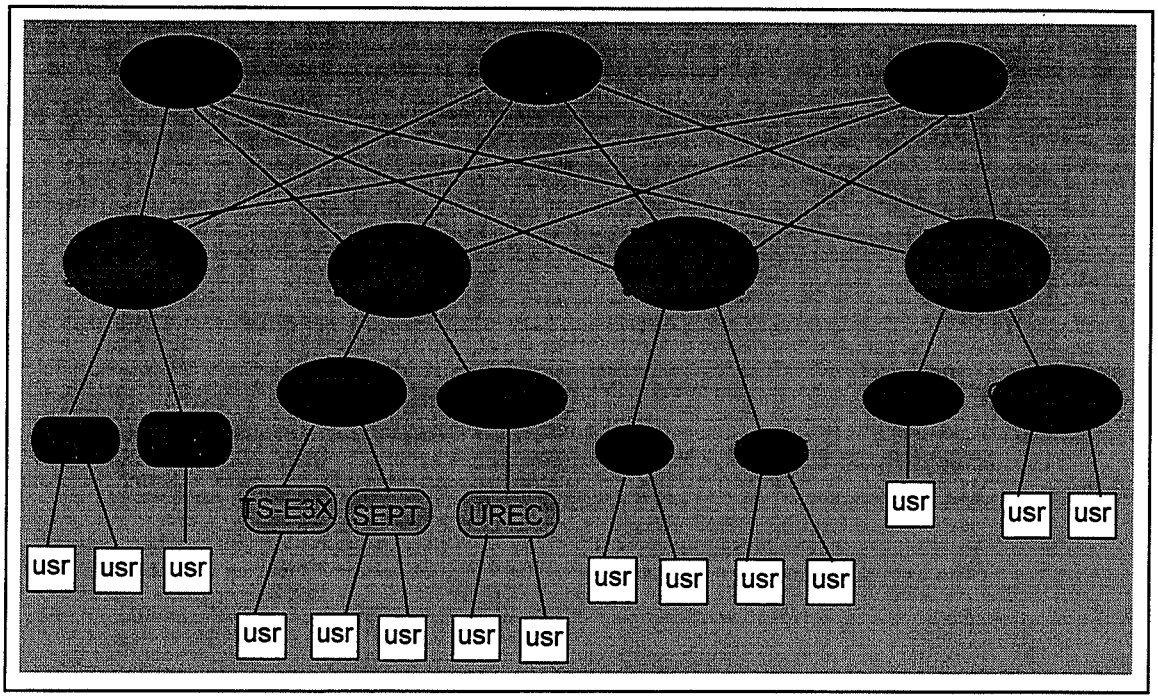

Figure 13. Modèle de Certification Password

Pendant la phase pilote du projet, d'autres institutions peuvent joindre la hierarchie. Pour le faire, celles-ci soumettront leur politique de sécurité et demanderont un certificat auprès de la PCA la plus conforme à leurs besoins.

Rapport avec le Modèle PEM

Le schéma de certification Password a été conçu en sorte d'être compatible avec celui défini pour PEM (voir 5.2.4). Ceci dit, il doit être possible à une CA de jouer le rôle de PCA (ou de CA Organisationnelle) dans les deux schémas de certification. Dans ce cas, cette CA doit se soumettre aux obligations des deux schémas.

Le modèle de certification PEM peut être vu comme un modèle dérivé du modèle Password. La différence principale entre eux est que dans le schéma PEM il existe une seule CA de haut niveau, à savoir, l'IPRA. Ainsi, ce modèle centralise l'autorité sur cette entité et oblige une connectivité globale entre tous ses intégrants.

Rapport avec le Modèle PGP

Contrairement à Password, le modèle PGP (décrit dans la Section 5.3) assume que l'espace de noms est non structuré et laisse à chaque utilisateur le soin de déléguer juridiction pour la certification. En termes de l'architecture Password, PGP correspond à un modèle complètement plat, où en principe personne n'est autorisé à signer des certificats.

Du fait que PGP constitue un modèle décentralisé et non institutionnel, un utilisateur PGP peut, à son choix, absorber, partiellement ou globalement, le schéma Password comme son modèle

privé[21]. Pour cela, il suffit que l'utilisateur déclare une des CA Password comme l'un de ses *introducteurs* (dans la terminologie PGP).

7.3. Les Applications

L'établissement d'une infrastructure d'authentification et le développement des boîtes à outils de sécurité, décrits dans les sections précédentes, ont été le point de départ du groupe de travail Password pour le développement des applications sécurisées. En fait, des mécanismes de sécurité ont été développés et intégrés à des implémentations existantes de la majorité des applications.

Pour chacune des applications, le travail de mise en place de la sécurité s'est déroulé en quatre étapes, à savoir, la spécification des exigences de sécurité, l'identification des utilisateurs et des procédures, le développement des mécanismes et les tests d'interopérabilité entre les partenaires du projet. Dans les sections suivantes nous décrivons les approches choisies dans Password pour la sécurisation de l'annuaire X.500 et de la messagerie X.400.

---

[21]La notion de modèle privé est détaillée dans la Partie IV de ce travail.

7.3.1. L'Annuaire Réparti X.500

Password considère l'annuaire non seulement comme une partie de l'infrastructure de sécurité, mais aussi comme un utilisateur des services de sécurité dans la mesure où il est nécessaire de protéger sa base d'information et ses communications contre l'accès non autorisé. L'annuaire X.500 et son modèle de sécurité ont été présentés dans la partie précédente (Section 6.2.1).

Comme point de départ pour la mise en oeuvre des mécanismes et des services de sécurité, le groupe Password disposait des implémentations de systèmes d'annuaire opérationnelles dans le cadre du projet Paradise, à savoir, QUIPU [KRRT91] dans les consortiums Anglais et Allemand, et UCOMX.500 [E3X92a] dans le consortium Français. Nous étions responsable, entre autres, du développement des extensions d'authentification forte pour l'implémentation disponible dans le consortium Français.

Le groupe de travail Password a étudié les exigences en sécurité de l'annuaire X.500 et a établi la politique de sécurité définissant les sémantiques associées aux mécanismes adoptés dans le projet [LW93].

Le modèle de sécurité X.500 s'adresse à la protection de la base d'information de l'annuaire lorsqu'elle est accédée à travers les protocoles DAP et DSP. Les aspects concernant l'accès local à la DIB, ainsi que son intégrité ou l'éventuel besoin de confidentialité ne sont pas pris en compte par ce modèle.

L'approche Password pour la sécurité de l'annuaire repose sur les services et mécanismes de sécurité standard. Ainsi, les politiques de sécurité Password à l'égard de l'annuaire, exigent strictement les services et mécanismes définis dans les Recommandations X.500[22]. Les fonctionnalités fournies par Password au niveau des DUA et DSA sont les suivantes :

- Des DUA capables d'appliquer la procédure d'authentification forte pour s'identifier auprès des serveurs d'annuaire (DSA) au moment de l'établissement de l'association.
- Des DSA capables d'appliquer la procédure d'authentification forte pour valider des requêtes d'établissement de l'association en DAP. Cela fournit au DSA une méthode sûre d'identification à partir de laquelle il peut non seulement consulter sa politique d'autorisation pour concéder ou refuser à l'utilisateur l'accès aux services d'annuaire, mais aussi obtenir une preuve de l'origine de la requête.
- Des DUA capables d'appliquer la procédure d'authentification forte pour authentifier le résultat d'établissement de l'association retourné par un DSA. Cela permet de garantir à l'utilisateur qu'il se connecte au serveur prétendu.
- Des DUA capables d'appliquer la procédure d'authentification forte pour envoyer des requêtes signées à des DSA. Ceci étant applicable à toutes les opérations pouvant avoir lieu pendant une session d'annuaire, i.e., les opérations de lecture, de recherche et de modification.

---

[22]Les mécanismes de contrôle d'accès définis dans la version 1994 des Recommandations X.500 n'ont pas été pris en compte dans Password.

- Des DUA capables d'appliquer la procédure d'authentification forte pour authentifier le(s) résultat(s) d'une opération d'annuaire. Cela permet à l'utilisateur d'identifier, de manière sûre l'origine des résultats, de vérifier qu'ils n'ont pas été modifiés pendant le transfert et qu'ils ne font pas l'objet d'un re-jeu.
- Des DSA capables d'appliquer la procédure d'authentification forte pour envoyer des résultats signés et pour valider des requêtes d'opérations signées. Cela fournit au DSA une méthode sûre d'identification à partir de laquelle il peut consulter sa politique d'autorisation pour concéder ou refuser à l'utilisateur l'accès au service et au(x) entrée(s) demandée(s). De plus, le DSA peut vérifier que la requête n'a pas été modifiée pendant le transfert et qu'elle ne fait pas l'objet d'un re-jeu.
- Des DSA capables de chaîner des opérations signées de façon transparente, par exemple transférer vers le demandeur originel d'une requête, des résultats signés par d'autres DSA.
- Des DUA et DSA capables de fournir un chemin de certification lors de l'utilisation des procédures d'authentification forte.

Pour la mise en oeuvre tant des jetons d'association que des opérations signées, le groupe Password a adopté les algorithmes de scellement MD2, MD4 et MD5 couplés à l'algorithme de chiffrement RSA.

Les services suivants sont considérés comme optionnels :

- la procédure d'authentification forte durant l'établissement de l'association en DSP,
- l'application de la procédure d'authentification forte pour envoi des requêtes en DSP.

L'implémentation des deux procédures ci-dessus aurait un impact important sur la performance du service d'annuaire. De plus, cela exige la conformité de tous les fournisseurs de services d'annuaires et l'adoption d'une politique globale de sécurité. Ainsi, une telle approche rendrait difficile l'interfonctionnement avec des implémentations ne respectant pas ce même profil.

A présent, des systèmes d'annuaire avec des extensions de sécurité sont opérationnels non seulement dans le cadre des tests pilotes Password mais aussi dans le cadre du projet Paradise. Cela montre que les DSA sécurisés développés dans le cadre de Password sont capables d'interfonctionner avec de systèmes X.500 non sécurisés.

Les Recommandations X.500 ne spécifient pas de services de confidentialité, car l'information de l'annuaire est généralement de nature publique et donc censée être globalement accessible. Cependant, l'annuaire peut être aussi utilisé pour conserver des informations qui ne doivent pas être globalement accessibles, notamment dans des contextes commerciaux. Cela constitue l'un des problèmes ouverts de la sécurité de l'annuaire pour lequels notre étude dans le domaine proposent des solutions. Ceux-ci sont présentés dans la Section 10.2.2.

7.3.2. La Messagerie Electronique

Tout utilisateur des services existants de courrier électronique est conscient de la vulnérabilité de ces systèmes. Il n'est pas nécessaire d'être un spécialiste pour envoyer un message en utilisant une fausse adresse ou l'adresse de quelqu'un d'autre. Une telle faiblesse peut être tolérable dans des contextes interpersonnels et informels, où les anomalies peuvent être vérifiées par des moyens

externes. En contrepartie, cela peut entraîner des conséquences graves dans le cas d'applications commerciales, par exemple l'EDI, qui traite la manipulation électronique de factures et d'ordres de paiement.

7.3.2.1. Les approches

Normalement, pour pouvoir sécuriser un message, il est nécessaire de lui ajouter certains paramètres de sécurité, par exemple, une clé chiffrée et/ou des jetons d'authentification. Ces paramètres de sécurité peuvent être ajoutés dans l'enveloppe du message, ou bien comme une partie du contenu du message, ou même les deux. Le choix de l'emplacement des paramètres de sécurité influe fortement non seulement sur le système résultant mais aussi sur les types de services de sécurité qui peuvent être fournis.

La série X.400-84 ne contient pas d'aménagements pour faciliter l'implémentation de procédures d'échange sécurisé de messages. Si l'on souhaite sécuriser la messagerie X.400-84, ou tout autre système de messagerie électronique ne disposant pas de mécanismes de sécurité incorporés, l'alternative est de dissocier les aspects de sécurité de la transmission électronique des messages, i.e., placer les paramètres de sécurité dans le contenu du message. Cela demeure valable si l'on souhaite interconnecter des systèmes de messagerie hétérogènes, même s'ils incorporent individuellement des mécanismes de sécurité. L'inconvénient d'une telle approche est que le contenu des messages passe de façon transparente par le MTS. Ainsi, ces paramètres de sécurité ne peuvent pas être utilisés pour fournir des services de sécurité au niveau de l'accès au MTS ni pour sécuriser le MTS lui-même.

Des exemples de systèmes de messagerie dans lesquels les paramètres de sécurité sont placés dans le contenu de messages sont les systèmes PEM [Lin93] et SDNS [SDN701]. Une solution de ce type, disponible au sein du groupe Password à travers le consortium Anglais, consiste en une implémentation sécurisée de l'ODA (**O**pen **D**ocument **A**rchitecture) [IS8613], un système normalisé de manipulation de documents éléctroniques. De tels documents peuvent être transmis via la messagerie X.400 et permettent l'authentification de l'auteur, la vérification de l'intégrité et la confidentialité des documents [IS8613AD].

Une des caractéristiques de l'approche X.400-88 est l'emplacement des paramètres de sécurité ; ceux-ci sont toujours placés dans les enveloppes de message et non dans le contenu. Si d'une part l'inclusion des paramètres de sécurité dans l'enveloppe permet la fourniture de services de sécurité au niveau du MTS, d'autre part cela rend problématique la fourniture de certains services de bout-en-bout, notamment si un Message Store est utilisé. Ces problèmes sont considérés en detail dans le Chapitre 10 (voir 10.3.3).

7.3.2.2. La messagerie X.400

Une des tâches du projet de Password est de rendre sûre la messagerie X.400. La version 1988 des Recommandations X.400 permettent la mise en œuvre de toute une gamme de services de sécurité. Ces services sont réalisés au moyen des éléments des protocoles X.400.

Les services de sécurité de la messagerie X.400-88 sont regroupés dans trois classes en fonction du niveau de service rendu [OIW91].

- ◊ *Classe S0/S0a* : spécifie le support de services de sécurité de bout-en-bout. Cette classe de service protège l'utilisateur final même dans les cas où le MTS ne rend pas un service sécurisé. La classe S0a rajoute le service de confidentialité de contenus à la classe S0.
- ◊ *Classe S1/Sa* : incorpore les services de la classe S0 et, en supplément, spécifie l'utilisation de services d'authentification pour l'accès au MTS et entre ses entités ainsi que la mise en vigueur de mécanismes de sécurité basés sur des étiquettes et l'interfonctionnement sécurisé entre des domaines de sécurité. La classe *S1a* rajoute le service de confidentialité de contenus à la classe S1.
- ◊ *Classe S2/S2a* : incorpore les services de la classe S1 et, en supplément, spécifie l'utilisation de services d'authentification de messages, d'avis de remise/non-remise et de demandes d'essai (probes) dans le MTS, de manière à pouvoir rendre les services de non-répudiation d'origine, de soumission et de remise de messages dans le MTS. La classe *S2a* rajoute le service de confidentialité de contenus à la classe S2.

La fonctionnalité minimale décidée par le groupe Password pour la messagerie X.400 est celle correspondante au profil S0a. Cela correspond à la fourniture de mécanismes de sécurité de bout-en-bout, i.e., UA ↔ UA, où les services rendus sont les suivants :

- * intégrité des contenus,
- * authentification de l'expéditeur,
- * confidentialité des contenu,
- * preuve de remise.

Les services de non-répudiation d'origine, de soumission et de remise sont inclus dans la classe S0a comme des services optionnels du fait de la complexité de leur réalisation. Si des techniques asymétriques sont utilisées, les mécanismes d'authentification peuvent aussi fournir des services de non-répudiation, selon le niveau de confiance accordé aux certificats. Si des techniques symétriques sont utilisées, les mécanismes d'authentification ne peuvent fournir la non-répudiation qu'à travers l'utilisation d'une tierce partie fiable (notaire) qui validera les données d'authentification.

Password disposait des implémentations non-sécurisées de la messagerie X.400-88 comme point de départ pour la mise en oeuvre de ces mécanismes. Le consortium Anglais utilise PP [KO91], le consortium Allemand utilise OSITEL/400 [DAN91] et le consortium Français utilise UCOMX.400 [E3X92b].

7.3.2.3. Les Choix d'Implémentation

Les mécanismes spécifiés dans les protocoles X.400 pour implémenter les services de sécurité sont basés sur l'approche X.509 tant au niveau de la gestion des clés qu'au niveau des protocoles d'authentification. Ceux-ci ont été brièvement introduits dans la partie précédante de ce travail (voir 6.2.2). Dans cette section nous détaillons ces mécanismes et décrivons les alternatives de

réalisation de ces services, les problèmes liés à leur implémentation, ainsi que les choix et les solutions retenus dans Password.

7.3.2.3.1. L'intégrité de messages

X.400 définit un type de données appelé *ContentIntegrityCheck* (CIC), destiné à fournir une preuve de l'intégrité du contenu d'un message. Il s'agit d'une signature calculée sur le contenu du message et l'identificateur de l'algorithme utilisé pour calculer la signature.

```
ContentIntegrityCheck ::= SIGNATURE SEQUENCE {
        algo-id     AlgorithmIdentifier,
        content     Content }
```

Le CIC constitue l'un des champs définis pour chaque destinataire du message (*PerRecipientMessageSubmissionFields*), et est toujours calculé à partir du contenu en clair du message, même si la confidentialité du contenu est requise.

Le CIC peut être inclus dans le jeton du message, ou hors ce jeton comme l'un des champs "par destinataire". Le service peut donc être rendu de deux manières différentes selon le placement du CIC. Encore une fois, le choix de placement va définir l'étendue du service résultant.

Si le CIC est inclus dans le jeton, il peut être accompagné de l'identificateur de contenu et d'une étiquette de sécurité [X.402], ce qui peut être significatif vis-à-vis de la politique de sécurité adoptée et du niveau de sécurité exigé par l'utilisateur. En outre, le CIC peut être inclus dans les champs *encrypted-data* ou *signed-data* du jeton

L'inclusion du CIC dans le champ *encrypted-data* sert à garantir sa confidentialité. En particulier, si le service de confidentialité est utilisé, l'inclusion du CIC dans ce champ empêche qu'un attaquant puisse déchiffrer la signature du CIC pour essayer de découvrir le contenu du message. Cela peut être fait en estimant des valeurs pour le contenu, en appliquant la fonction de scellement sur ces valeurs et puis en comparant ce résultat avec le résultat originel de scellement.

Si le CIC est transmis hors du jeton, les services additionnels mentionnés ci-dessus ne sont pas fournis. Mais dans les deux cas, le CIC fournit implicitement une preuve d'authenticité du message et permet de fournir le service de non-répudiation d'origine car il est formé d'une signature chiffrée au moyen de la clé privée de l'utilisateur. En contrepartie, puisque le CIC est calculé à partir du contenu en clair, si le service de confidentialité de contenu doit être assuré, alors l'authenticité du message ne peut être vérifiée que par les destinataires du message.

Il doit être noté que si le CIC est inclus dans le jeton, le service de non-répudiation résultant est d'une qualité supérieure car le contenu, en clair, du message est signé accompagné d'autres informations significatives telles que le nom du destinataire et la date-heure de génération du jeton.

7.3.2.3.2. L'authentification de l'expéditeur

Afin de fournir un moyen d'authentifier l'origine d'un message, les recommandations X.400 définissent la structure *MessageOriginAuthenticationCheck* (MOA). Il s'agit d'une signature

calculée à partir du contenu du message pouvant être accompagnée d'un identificateur de contenu et d'une étiquette de sécurité.

```
MessageOriginAuthenticationCheck ::= SIGNATURE SEQUENCE {
  algorithm-identifier        AlgorithmIdentifier,
  content                     Content,
  content-identifier          ContentId OPTIONAL,
  message-security-label      MessageSecurityLabel OPTIONAL }
```

Contrairement au CIC, le MOA est calculé à partir de la version chiffrée du contenu du message. Ceci permet que l'origine du message puisse être vérifiée par toute entité du MHS par laquelle le message transite, sans compromettre la confidentialité du message. Ainsi, le MOA offre un mécanisme d'authentification permettant de détecter l'utilisation de fausse identité. Lorsque couplé à un mécanisme de contrôle d'accès, le MOA fournit aux ressources du MHS une protection contre l'utilisation non autorisée.

En contrepartie, si le service de confidentialité est utilisé, le MOA ne peut pas rendre le service de non-répudiation d'origine. Cela vient du fait que ce service est considéré comme une preuve irréfutable et donc ne peut être fournie que si l'utilisateur signe l'information sous une forme lisible. Ainsi, si le contenu est chiffré avant la génération de la signature on ne peut pas prouver la liaison entre le contenu et l'expéditeur.

Si le service de confidentialité n'est pas utilisé, le MOA et le CIC rendent, en principe, le même service, sauf que le MOA permet d'associer une étiquette de sécurité au contenu, ce qui peut conférer un niveau de sécurité plus élevé. En contrepartie, à cause des informations supplémentaires contenues dans le jeton, si le CIC y est inclus, le service résultant confère un niveau de sécurité plus élevé que celui fourni par le MOA.

De plus, il doit être noté que le MOA est identique au CIC aux champs optionnels près. Tous ces aspects reflètent un manque de clarté et des redondances du modèle de sécurité défini pour la messagerie X.400.

Dans Password l'objectif a été plutôt d'étudier et d'expérimenter le modèle de sécurité défini pour X.400 que de définir des politiques de sécurité restrictives. Ainsi, le choix du projet a été d'implémenter tant le MOA que le CIC, celui-ci dans ou hors jeton. Cela permet aux implémentations d'interfonctionner avec toute autre implémentation sécurisée de X.400 indépendamment des politiques de sécurité, aussi bien que de s'adapter à toute une gamme de politiques plus ou moins restrictives.

En ce qui concerne la mise en oeuvre de ces mécanismes, le groupe Password a adopté les algorithmes de scellement MD2, MD4 et MD5, couplés à l'algorithme de chiffrement RSA.

7.3.2.3.3. La confidentialité de contenus

Dans X.400, les services de confidentialité sont rendus à travers le champ *encrypted-data* des jetons. L'algorithme de chiffrement n'est pas spécifié dans les recommandations et peut être du type symétrique ou asymétrique.

Password utilise l'algorithme symétrique DES en mode CBC [ANSI83]. Puisque les clés sont de longueur relativement courte (56 bits), une clé est sélectionnée de façon aléatoire à chaque fois qu'un message confidentiel doit être envoyé. Ensuite, un jeton de message est produit pour chaque destinataire de façon à ce que la clé de chiffrement soit elle-même chiffrée au moyen de la clé publique de chacun de ces destinatires.

7.3.2.3.4. La preuve de remise

Le dernier service de sécurité de bout-en-bout d'intérêt dans Password est la preuve de remise. Ce service permet à l'expéditeur d'un message de vérifier de façon fiable que le message à été livré au destinataire voulu. Ce service est d'une nature différente des autres déjà décrits car il n'est pas rendu par l'expéditeur d'un message mais par le destinataire (voir Figure 21).

Ce service est indépendant des services de sécurité fournis par l'expéditeur. Pour chaque destinataire d'un message, l'expéditeur indique s'il souhaite une preuve de remise du message en question. Cette indication peut être incluse dans le jeton de message pour garantir l'intégrité et l'authenticité de la requête.

La preuve de remise (PoD - ***Proof of Delivery***) est en fait une signature calculée à partir du contenu en clair du message reçu, conjointement à l'identificateur de contenu, le nom du destinataire, la date-heure de remise et d'autres paramètres relatifs au message, tels que l'identificateur de contenu et l'étiquette de sécurité. Ces paramètres doivent être inclus s'ils sont présents dans le message originel.

A la réception d'un tel message, l'UA du destinataire doit produire la valeur scellée de la preuve de remise qui ensuite doit être chiffrée au moyen de la clé privée de cet utilisateur. Ensuite, la preuve de remise doit être renvoyée au MTA destinataire, comme résultat de l'opération de remise de message, pour qu'elle puisse alors être retournée à l'expéditeur du message comme partie d'un avis de remise. La preuve de remise peut servir à des fins de non-répudiation de remise.

La définition de ce service mélange les responsabilités de différentes instances. Le PoD doit être retourné par le destinataire d'un message au MTA comme résultat de l'opération de remise. Ainsi, cette preuve de "remise" peut être interprétée comme une preuve qu'un MTA a bien remis le message au destinataire. En contrepartie, ce service doit être interprété plutot comme une preuve de la "réception" du message par un destinataire. Cela vient du fait que le destinataire doit signer le PoD à partir du message en clair, et ceci exige que le destinataire ait pris connaissance du message dans une forme lisible avant de retourner la preuve de remise.

Par conséquent, la réalisation de ce service exige la disponibilité et l'action effective du destinataire du message. D'abord, pour déchiffrer le message (si le service de confidentialité est fourni) et puis pour signer le PoD. Or, ni le destinataire ni l'UA ne sont pas forcément disponibles lors de la réception d'un tel message. De plus, les problèmes de réalisation du service de preuve de remise sont amplifiés dans le cas où l'utilisateur reçoit ses messages via un Message Store (MS) pour récupérer ses messages.

La réalisation de ce service se heurte ainsi à de nombreux problèmes. Dans Password, ce service a été étudié mais aucune solution n'a été définie. Ceci constitue l'un des problèmes ouverts auquel

nous nous sommes intéressés. Dans la Section 10.3.3 nous discuterons ce problème en essayant d'apporter une solution.

7.3.2.4. La messagerie PEM

Une autre contribution de Password dans le cadre de la sécurité de la messagerie électronique est l'implémentation de PEM. Cela représente l'une des premières tentatives d'implémentation de cette approche, en dehors du groupe de travail PEM et en utilisant un schéma de certification autre que celui défini dans le cadre de ce groupe de travail.

Un partenaire de chaque consortium a développé sa version de PEM, à savoir, l'INRIA dans le consortium Français, GMD dans le consortium Allemand et UCL dans le consortium Anglais. Les tests pilotes ont montré la capacité d'interopérabilité entre ces implémentations et l'indépendance de l'approche par rapport au modèle de confiance adopté. De plus, l'utilisation de l'infrastructure PARADISE a permis l'utilisation des services sécurisés PEM sans le besoin d'échange préalable de certificats.

7.3.3. D'autres Applications

Certains services normalisés, comme l'annuaire X.500 et la messagerie X.400-88, spécifient des extensions offrant des services de sécurité dans leurs propres protocoles. D'autres applications ne disposent pas de tels mécanismes. L'intégration de la sécurité dans des telles applications demanderait alors une nouvelle spécification de ces protocoles. Dans les cas où une granularité plus fine n'est necessaire, il est préférable d'essayer d'utiliser un "élément de service commun" de la couche application comme un moyen d'implémenter des mécanismes de sécurité.

7.3.3.1. ACSE

L'ACSE (**A**ssociation **C**ontrol **S**ervice **E**lement) est la partie de la couche Application du modèle OSI qui contrôle l'établissement et la fermeture des associations entre des entités d'application. Il constitue un composant commun et réutilisable pouvant faire partie de différentes applications.

En 1990, l'ISO a publié un amendement à la norme ACSE traitant l'authentification pendant l'établissement des associations [X.227]. Cet amendement a pour objet de fournir des extensions aux spécifications ACSE de façon à supporter des services d'authentification au moyen d'une nouvelle unité fonctionnelle, appelée "*Authentication*". Cette unité fonctionnelle permet le transfert de données d'authentification lors de l'etablissement de l'association.

Un élément de service d'application (ASE) peut disposer d'un module particulier désigné pour offrir des moyens de sécurité à son utilisateur de service. Lorsque l'unité fonctionnelle Authentication est employée, des paramètres optionnels sont fournis par la primitive de service A-ASSOCIATE pour échanger des données d'authentification entre des entités d'application. Le format défini, dans l'ACSE, pour ces données est le suivant :

```
Authentication ::= SEQUENCE {
  mechanism-name          [0] IMPLICIT OBJECT IDENTIFIER OPTIONAL,
```

```
authentication-value        [1] CHOICE {
    charstring                      [0] IMPLICIT GraphicString,
    bitstring                       [1] IMPLICIT BITSTRING,
    external                        [2] IMPLICIT EXTERNAL,
    other                       [3] ANY DEFINED BY mechanism-name }
}
```

Le champ *mechanism-name* sert à spécifier le mécanisme d'authentification qui sera utilisé par l'application en question. Ce champ peut aussi servir à specifier un mécanisme de sécurité plus général englobant un mécanisme d'authentification.

Le champ *authentication-value* comprend l'information que les utilisateurs du service ACSE utilisent pour s'acquitter de l'authentification. Il peut s'agir d'une accréditation, d'un jeton de sécurité, d'une signature numérique, etc. Il peut aussi déterminer le type et/ou le nom de l'objet à authentifier ; par exemple, l'entité d'application, un usager, etc.

La structure sémantique de la valeur d'authentification est spécifiée par le mécanisme d'authentification employé et est traitée par l'ACSE comme objet atomique. Puisque l'ACSE représente la fonction d'authentification sous cette forme, il peut faire face aux besoins d'échange d'information d'authentification sans avoir à comprendre la sémantique de cette information ou son mode d'utilisation.

Password a choisi l'ACSE comme l'élément commun de service à travers lequel des procédures de sécurité seront offertes, en utilisant l'unité fonctionnelle d'authentification. Toutefois, l'ACSE ne constitue pas un système complet en soi, mais un élément de service au-dessus duquel il est possible de bâtir différentes applications. Par conséquent, il n'est pas possible d'établir de façon définitive quelles sont les exigences que l'utilisateur peut avoir par rapport à cet élément. Ces exigences peuvent varier selon l'application en question. Cependant, il est possible de décrire un ensemble de mécanismes utililes pouvant être fournis à travers l'ACSE.

Le groupe a donc étudié les besoins générales en authentification des applications [RSK92]. Comme résultat, des protocoles d'authentification à sens unique, à double sens et à trois échanges ont été définis [Roe93]. Outre l'authentification, ces protocoles permettent de réaliser d'autres services de sécurité, à savoir l'intégrité, la confidentialité et la preuve d'origine (et par conséquent la non-répudiation, selon le mécanisme d'authentification choisi). Ces services peuvent être utilisés localement pour batir un mécanisme de controle d'accès, par exemple.

Il doit être noté que le protocole d'authentification à trois échanges ne peut pas être implémenté car la couche session ne comporte pas ce nombre d'échanges.

7.3.3.1.1. Le Protocole à sens unique

Le protocole d'authentification à sens unique, permettant l'authentification unilatérale, est défini de sorte que l'entité initiatrice *A* envoie à l'entité *B* auprès de laquelle elle désire s'authentifier, les données suivantes :

$$A \rightarrow B : [\{A, B, t1, r1, K_{ab}\}K_a^{-1}]K_{ab}, [K_{ab}]K_b$$

Où,

- A → B denote "A envoie à B",
- *t1* est une indication du temps courant,
- *r1* est un nombre aléatoire non utilisé auparavant, au moins avec B,
- K_a est la clé publique de A,
- K_a^{-1} est la clé privée de A,
- K_{ab} est la clé de session entre A et B,
- *[data]K* indique les données *data* chifrées au moyen de la clé *K*,
- *{ data }K* indique les données *data* signées au moyen de la clé privée K^{-1}.

A travers l'échange ci-dessus, l'entité B obtient la preuve de l'identité de A. En outre, une clé symétrique de session K_{ab} est transférée à B de façon à permettre l'échange postérieure de données confidentielles :

$$A \rightarrow B \text{ et } B \rightarrow A : \{data\}\, K_{ab}$$

7.3.3.1.2. Le Protocole à double sens

Le protocole de sécurité à double sens permet l'authentification mutuelle entre les parties communicantes. Ceci adopte une variante de la méthode Diffie-Hellman pour l'établissement d'une clé commune. Ce protocole a la définition suivante :

$$A \rightarrow B : \{ A, B, t1, r1, H(G_a, G0) \}K_a^{-1}$$

$$B \rightarrow A : \{ B, A, t2, r2, H(G_b, G0) \}K_b^{-1}$$

Dans ce protocole, deux sortes de fonction de scellement sont utilisées, une pour le calcul de la signature du jeton et une autre pour le calcul des sous-clés, à savoir *H*, sous la forme $H(G_a, G0)$ et $H(G_b, G0)$. Celles-ci permettront la construction d'une clé de session *Kab*. En effet, n'importe quelle fonction à sens unique de haute assurance peut être utilisée pour la signature du jeton. En contrepartie, la fonction de scellement *H* utilisée pour l'établissement de la clé de session doit, en plus, obéir à la relation de commutativité suivante :

$$H(G_a, H(G_b, G0)) = H(G_b, H(G_a, G0))$$

D'après le mécanisme proposé par Diffie-Hellman, une fonction qui possède cette propriété est la fonction *H* tel que $H(x,g) = g^x \ mod\ p$. Les nombres *p* et *g* doivent être choisis au préalable et peuvent être publics, mais ils doivent être tels que *p* est un grand nombre premier et *g* un élément primitif du champ de Galois de *p*, GF(p) ; et *x* represente une variable privée, devant être differente pour chaque partie commnucante. Dans le cas de l'échange entre A et B :

- le paramètre public *g* correspond à *G0*,
- la variable *x* correspond à G_a e à G_b. Ceux-ci sont les paramètres privés de A et B.

Bien que ce mécanisme impose des contraintes, à savoir, l'accord sur *G0* et *p* et le coût de génération d'un grand nombre premier *p*, l'avantage de ce mécanisme est que la sécurité de la clé

de session ne dépend pas de la sécurité des clés privées des participants. Pour qu'un intrus puisse découvrir la valeur de K_{ab}, il doit être en possession des valeurs de G_a et G_b.

Pour mettre en oeuvre les protocoles d'authentification, un format général d'accréditations a été défini. Ces accréditations peuvent être transférées au moyen du champ "*external*" de l'unité fonctionnelle Authentication de l'ACSE. Le format de ces accréditations est le suivant :

```
StrongCredentials ::= SET {
    [0]   Certificates OPTIONAL,
    [1]   Token
}

Token ::= SIGNED SEQUENCE {
    algorithm       [0]   AlgorithmIdentifier,
    claimant        [1]   DistinguishedName,
    verifier        [2]   DistinguishedName OPTIONAL,
    time1           [3]   UTCTime,
    random1         [4]   BIT STRING,
    time2           [5]   UTCTime OPTIONAL,
    random2         [6]   BIT STRING OPTIONAL
}
```

Il doit être noté que le format de jeton ci-dessus ne permet pas l'échange de données spécifiques aux applications, ni l'échange de données confidentielles. Dans la Section 10.4 nous proposons un format général de jeton permettant de remédier ces limitations.

CHAPITRE 8

SECUCOMX : UNE ARCHITECTURE DE SECURITE POUR LES SYSTEMES OUVERTS

Un des résultat du projet Password est le développement de différentes implémentations de la technologie de base dans l'intention de faciliter la mise en place de solutions différentes, tout en gardant l'interopérabilité. Notre contribution personelle, dans le cadre du consortium français, a été la conception d'une architecture à partir de laquelle toute une gamme des services de sécurité peut être implémentée.

L'architecture de sécurité **SecUcomx** a été conçue avec l'objectif de proposer des solutions aux problèmes actuels concernant la sécurité des applications ouvertes, et de fournir une infrastructure de sécurité adéquate et assez flexible pour supporter l'évolution des applications et de la technologie de sécurité elle-même. Des hypothèses initiales ont été fixées lors de la conception de cette architecture. Celles-ci peuvent être résumées par les points suivants :

- la compatibilité avec l'approche X.509 de manière à pouvoir la valider, vu que cette approche est à la base de divers modèles de sécurité et applications existants,
- l'intégration dans l'environnement OSI, ce qui nécessite la fourniture de procédures de sécurité indépendantes des couches OSI,
- la compatibilité avec l'approche Password.

Après une étape d'analyse des modèles et schémas existants, la première phase du développement de cette architecture a été l'implémentation des techniques cryptographiques. La phase suivante a concerné la gestion de clés. Le choix de l'approche X.509 comme base conceptuelle de l'architecture nous a motivé à des recherches sur la gestion des autorités de certification, leur rapport avec l'annuaire X.500 et, notamment, l'obtention et la validation de certificats. Après avoir testé les fonctionnalités de l'architecture au sein du projet Password, nous avons procédé à l'intégration de cette architecture dans des applications.

Ce chapitre présente le modèle et les mécanismes offerts par chacun des composants de l'architecture SecUcomx. Le chapitre suivant décrit l'utilisation des composants de cette architecture pour la sécurisation de l'annuaire X.500 et de la messagerie X.400.

8.1. Les Services et Composants

L'architecture SecUcomx consiste en une collection de mécanismes de sécurité qui coopèrent pour fournir une infrastructure de sécurité pour des utilisateurs de systèmes ouverts et pour permettre l'implémentation d'applications sécurisées sur un réseau ouvert. Les services de sécurité abordés comprennent :

* l'authentification,
* l'intégrité de données,
* la confidentialité,
* la preuve d'origine (permettant le service de non-répudiation),
* le contrôle d'accès.

SecUcomx prétend couvrir les aspects de fourniture et de gestion de la sécurité pour les applications OSI. Cela exige la définition d'une architecture de gestion de clés compatible avec les standards de base. Dans SecUcomx, ces aspects sont traités par les modules suivants :

* La boîte à outils de sécurité,
* *KS*, le module de génération de clés,
* *GS*, le gestionnaire de secrets,
* *CS*, l'outil de gestion de certificats (CA),
* *DAS*, le serveur de données pour l'authentification.

Chacun de ces modules sera décrit dans la suite de ce chapitre.

8.2. La Boîte à Outils de Sécurité

L'élaboration de l'architecture SecUcomx a nécessité le développement d'une infrastructure cryptographique permettant la réalisation des opérations de chiffrement, de scellement et de génération/vérification de signatures numériques.

Le tool-kit de sécurité est une librairie qui offre une API (**A**pplication **P**rogramming **I**nterface) dont les fonctions peuvent être appelées par chacun des modules de l'architecture et par les applications bâties sur l'architecture.

Cette API de sécurité est bâtie sur le compilateur *Mavros* [Hui91a]. Ce compilateur fournit des fonctions permettant la manipulation les types ASN.1 définis par le cadre X.509, notamment les macros de sécurité SIGNED, SIGNATURE, ENCRYPTED et ALGORITHM. Ainsi, toute application basée sur l'approche X.509 peut être développée en utilisant cette API de façon indépendante du reste de l'architecture.

L'élaboration de cette infrastructure cryptographique a été facilitée par l'utilisation d'une version avancée du compilateur ASN.1 Mavros qui intègre dans ses fonctionnalités de base :

- une bibliothèque d'entiers longs portable,
- des opérations arithmétiques élémentaires (l'addition, la soustraction, la multiplication) ainsi que des opérations plus complexes, tels que le modulo et la fonction puissance, sur des entiers longs,
- la génération de grands nombres premiers ainsi que des fonctions de test sur ces nombres.

De plus, l'API de base de SecUcomx offre les algorithmes et fonctionalités suivants :

- L'algorithme de chiffrement symétrique DES en mode ECB [ANSI83],
- L'algorithme de chiffrement asymétrique RSA [RSA78] et la méthode de remplissage PKCS #1 block type 1 et 2 [PKCS #1],
- Les fonctions de scellement à sens unique MD2 [Kal92], MD4 [Riv92] et MD5 [Riv92],
- Des fonctions de génération et de vérification de signatures numériques,
- Des outils de gestion de jetons d'authentification X.509,
- Des outils pour l'enregistrement des traces sur la manipulation d'objets liés à la sécurité.

Cette infrastructure cryptographique est à la base de toute l'architecture SecUcomx, et fournit l'accès à la technologie de base. L'API de sécurité est intégrée dans tous les éléments de l'architecture. Ces éléments sont décrits par la suite.

8.3. La Gestion de Clés

La sécurité de toute architecture basée sur l'utilisation des systèmes cryptographiques dépend du niveau d'assurance des mécanismes de gestion et de manipulation de clés. La fiabilité de la génération des clés et la sécurité de la distribution de ces clés comptent autant que l'assurance de l'algorithme de chiffrement utilisé. En effet, une distribution peu sûre de clés peut compromettre tout le schéma de certification et rendre inutile la complexité des algorithmes.

Comme nous l'avons déjà vu, dans le cas de systèmes symétriques, une instance de clé symétrique ne doit être utilisée que pendant une seule session entre deux entités pour la confidentialité de leur communication. Cependant, l'architecture permet une plus grande granularité, i.e., que cette clé symétrique puisse être renouvelée au cours d'une même session, au choix de l'utilisateur. Les clés de session, de longueur moyenne et de courte durée de vie, sont générées localement, du coté de l'utilisateur, par des fonctions de l'infrastructure cryptographique et sont échangées après avoir être chiffrées par une clé asymétrique, à savoir, la clé publique du destinataire des données.

En ce qui concerne les clés asymétriques, il s'agit de systèmes plus complexes, où le composant privé n'est pas partagé mais utilisé pour prouver l'identité des entités. Ces clés sont de longueur relativement importante et peuvent donc avoir une durée de vie plus longue. Puisque les clés asymétriques sont utilisées pour l'authentification, chaque utilisateur doit obtenir ses paires de clés d'une source fiable. Dans la suite de cette section, nous abordons les aspects liés à la génération de clés asymétriques ainsi que la gestion de clés publiques et privées.

8.3.1. La Génération de Clés

8.3.1.1. L'emplacement de la Génération de Clés

Le choix de l'emplacement de la génération de clés asymétriques doit prendre en compte trois aspects importants :

- **Le transfert du composant public pour la certification**

  On doit éviter qu'un intrus puisse obtenir un certificat en utilisant l'identification de quelqu'un d'autre, ou remplacer la clé publique d'un utilisateur par la sienne. Dans les deux cas, l'intrus serait capable d'usurper l'identité de l'utilisateur et de déchiffrer des données confidentielles destinées à celui-ci.

- **Le transfert de la clé publique de la CA à l'utilisateur**

  L'utilisateur doit s'assurer que cette clé provient bien de la CA prétendue, pour qu'il ne soit pas trompé par des faux certificats lors des futures procédures d'authentification.

- **La mise à disposition du composant privé**

  Toute partie ayant connaissance de clé privée d'un usager est capable d'usurper son identité et de déchiffrer des données confidentielles destinées à cet usager.

Par conséquent, il est fortement recommandé que la méthode de distribution de clés soit de haute assurance, car un transfert peu sûr de clés peut compromettre tout le schéma de sécurité indépendamment de la complexité du système cryptographique. Pour cela, plusieurs approches peuvent être envisagées. Chaque méthode a ses avantages et ses inconvénients, comme nous le décrirons par la suite.

8.3.1.1.1. Tierce partie

La solution la plus commode est d'établir un module central ayant la charge de générer les paires de clés pour tous les utilisateurs. L'avantage d'une telle approche est que ce module central pourrait être doté des capacités de calcul garantissant la bonne qualité des clés. En contrepartie, cela ne constitue pas une bonne solution du point de vue de la sécurité des utilisateurs pour les raisons suivantes :

- il faut accorder un montant élevé de confiance à ce type de serveur car il a le potentiel d'usurper l'identité et de manipuler les communications de tous les utilisateurs,
- le serveur lui-même doit être fortement sécurisé car une attaque contre ce serveur peut, en cas extrême, mettre tous ses utilisateurs en danger,
- une telle approche peut impliquer la transmission des clés privées sur le réseau, ce qui expose ces clés à des attaques. Une telle procédure doit être toujours evitée.

8.3.1.1.2. CA

Une autre possibilité serait d'attribuer aux CA la tâche de générer les paires de clés des utilisateurs. Cette méthode résoud les problèmes liés au transfert de la clé publique pour la certification et simplifie la communication de la clé de la CA à ses clients car cette clé peut être chiffrée au moyen de la clé privée de l'utilisateur. De plus, ayant considéré les sites opérants des CA comme des sites bien protégés et dotés de bonnes capacités de calcul, l'autre avantage de cette méthode réside dans la bonne protection et la qualité des clés.

Toutefois, il doit être noté qu'il est techniquement possible à la CA d'usurper l'identité d'un utilisateur en utilisant une copie de sa clé privée. Des plus, un utilisateur peut faussement prétendre que des opérations authentifiées sous cette clé aient été, en fait, effectuées par une action malveillante de la CA.

En outre, la clé privée doit être livrée à l'utilisateur tout en préservant sa confidentialité. Cela constitue un autre inconvénient de cette approche car la livraison de la clé privée nécessiterait un contact direct entre l'utilisateur et, par exemple, un ingénieur système responsable de la sécurité (ou l'opérateur de CA). Cela sous-entend le potentiel d'attaques intérieures (*insider attacks*) car cela laisse à la CA, et au personnel responsable de cette autorité, le potentiel d'usurpation de l'identité des utilisateurs.

8.3.1.1.3. L'utilisateur

La solution qui nous avons choisi pour SecUcomx est de faire générer les clés par les utilisateurs. Chaque utilisateur dispose, dans son environnement, des capacités de génération de clés cryptographiques. Cette approche permet de réduire les risques d'exposition des clés privées et élimine le besoin de faire confiance à une source externe.

Dans SecUcomx chaque utilisateur est doté d'un module de génération de clés asymétriques qui opère dans l'environnement local de l'utilisateur. Ce module est appelé **KS** (***K**ey Generation **S**ervice*). Le KS protège l'environnement de l'utilisateur pendant la génération des clés et détruit activement toute information relative à la génération de la paire de clés, en effaçant les zones de mémoires utilisées pour conserver des informations pendant la génération des clés.

Une fois les clés générées, celles-ci doivent être validées pour garantir la non prédictabilité et leur bonne qualité cryptographique, ensuite :

- le *composant publique* doit être transporté et soumis à l'autorité (CA) chargée de certifier cette clé publique.
- l'utilisateur doit avoir accès au *composant privé* pour le chiffrement et le déchiffrement des données (génération de signatures et accès à des informations confidentielles, respectivement),

Pour la certification de la clé publique, le KS construit un "*message de demande de certification*" contenant la clé publique de l'utilisateur, qui sera envoyé à la CA. Ce message est protégé de façon à ce que la CA désignée par l'utilisateur puisse vérifier l'authenticité et l'intégrité du message et qu'elle soit la seule à pouvoir décoder le message. La procédure de certification est détaillée dans la Section 8.3.3.

L'accès et de la protection de clés privées constitue aussi un problème difficile et important puisque cette clé est utilisée pour prouver l'identité des entités. Dans la Section 8.3.2 nous abordons les solutions possibles et le choix retenu dans SecUcomx pour résoudre ce problème.

8.3.1.2. La Génération de Nombres Aléatoires

La sécurité des crypto-systèmes à clé publique repose sur la confidentialité des clés privées. Afin de maintenir cette confidentialité, des mécanismes de sécurité doivent prévenir l'accès non autorisé aux clés privées stockées. Indépendamment du niveau d'assurance de ces mécanismes, un attaquant pourrait encore déterminer la valeur d'une clé privée en prenant connaissance de la façon dont cette clé a été initialement générée. En particulier, toute personne ayant examiné un logiciel de génération de clés, peut connaître les détails de son fonctionnement. Ainsi, la procédure de génération de clés doit comprendre un élément "véritablement aléatoire" pour empêcher qu'un attaquant puisse prédire la valeur des clés privées.

8.3.1.2.1. Générateurs de Nombres Pseudo-Aléatoires

La plupart des ordinateurs et des langages machine sont complètement déterministes, i.e., pour une même valeur d'entrée, un même programme produira la même valeur de sortie. Il n'est donc pas possible de créer des valeurs aléatoires simplement par logiciel [Roe92]. Les sous-programmes disponibles dans de nombreux systèmes génèrent des séquences *pseudo-aléatoires*. Les nombres ainsi générés partagent des propriétés statistiques avec les séquences aléatoires, et ainsi peuvent être utilisés dans de nombreuses applications. La cryptographie est l'une des quelques applications où ces générateurs de nombres pseudo-aléatoires conventionnels ne sont pas acceptables.

Certains algorithmes de génération de nombres pseudo-aléatoires ont la propriété additionnelle d'être *cryptographiquement forts*. Ceux-ci prennent comme entrée, une valeur initiale de germe (*seed*) véritablement aléatoire, et produisent à partir de celle-ci, une séquence pseudo-aléatoire. Il n'est pas possible de dériver, par calcul, des éléments inconnus d'une séquence pseudo-aléatoire en connaissant d'autres éléments. Cependant, le résultat de ces générateurs de nombres pseudo-aléatoires n'est pas non plus véritablement aléatoire. Ce résultat peut être prédit par un attaquant qui connaît la valeur initiale de germe.

Toutefois, si la valeur initiale de germe est maintenue secrète et si le crypto-système est suffisamment fort, ces générateurs de nombres aléatoires (*RNG - Random Number Generators*) cryptographiques peuvent être utilisés par certaines applications cryptographiques. Malgré cela, l'utilisation de ces RNG cryptographiques n'élimine pas le besoin d'un RNG en support matériel (*Hardware RNG*) car de toutes façons, la valeur de germe initiale doit être générée.

8.3.1.2.2. Générateurs Matériels de Nombres Aléatoires

Pour être capable de produire des nombres aléatoires, un système doit être étendu avec un dispositif matériel pouvant se comporter de façon aléatoire. Le comportement de ce dispositif, tel que observé par la machine, doit être influencé par un processus physique de manière impossible à prédire.

Presque tous les périphériques (par exemple, disques, clavier, interfaces réseaux) pourraient être utilisés pour cela, mais pour la plupart d'entre eux il est difficile de *garantir* l'absence de prédictabilité. Par exemple, le comptage du nombre de paquets ethernet reçus à chaque seconde donne un résultat apparemment aléatoire, mais cela est facilement prédictible par quelqu'un monitorant le même ethernet.

Une meilleure approche est d'utiliser un périphérique spécialement conçu pour fournir un niveau maximum de non prédictabilité. Une approche commune à des RNG matériels consiste à prendre une bonne source électronique de bruit, comme la tension ou le courant électrique, l'amplifier et puis en prélever des valeurs à intervalles réguliers.

8.3.1.2.3. L'utilisation pour la Generation de Clés

Il existe deux alternatives pour l'introduction d'un élément aléatoire dans la procédure de génération de clés :

(1) une nouvelle valeur d'entrée aléatoire peut être obtenue pour chaque nouvelle clé générée,

(2) une valeur aléatoire unique peut être obtenue une seule fois durant l'initialisation du système, et utilisée comme valeur initiale de germe d'un RNG cryptographiquement fort.

L'alternative (2) est utile dans des situations où une même entité génère plusieurs clés différentes, par exemple dans le cas où la CA ou une tierce partie génère les clés des utilisateurs. Cette approche a aussi l'avantage de réduire le nombre nécessaire de bits véritablement aléatoires, cependant, elle est vulnérable sur deux autres points :

- Le crypto-système utilisé pour la génération de nombres pseudo-aléatoires peut être cassé, permettant à un attaquant connaissant une seule clé privée, de calculer toutes les clés privées générées par la même autorité. L'utilisation de cette méthode rend tout le système vulnérable à des attaques mathématiques sur le RNG cryptographique et sur l'algorithme de signature à clé publique. Pour cette raison, il est préférable d'utiliser un RNG dont la sécurité est équivalente à la sécurité de l'algorithme de signature utilisé.
- Si un attaquant réussit à découvrir la valeur initiale de germe, il sera donc capable de calculer toutes les clés privées déjà générées. Bien que cette valeur soit fortement protégée, celle-ci peut être révélée par un opérateur de CA corrompu ou menacé, ou par un vol du dispositif matériel utilisé pour la génération de clés [Roe92].

  Pour se protéger contre cette attaque, il est souhaitable d'utiliser un RNG cryptographique qui ne requiert pas la conservation de la valeur initiale de germe, mais qui au lieu de cela, met à jour une valeur stockée après la génération de chaque clé, utilisant une fonction à sens unique. Après la génération de chaque membre de la séquence, l'information nécessaire à son calcul peut être détruite afin de ne pas être retrouvée à posteriori.

8.3.2. La Gestion de Clés Privées

La plus part des mécanismes cryptographiques de sécurité font usage de clés dont une partie doit être maintenue secrète par l'utilisateur. Particulièrement, dans le cas de systèmes asymétriques, les clés privées ne doivent jamais être divulguées ni exposées. Ces clés sont trop longues (entre 500 et 1000 bits environ) pour être mémorisées et saisies par l'utilisateur à chaque session ou accès pour le chiffrement. Ainsi, la question du stockage et de l'accès aux clés privées constitue un problème très important. Comme solution, nous pouvons envisager diverses approches :

- Une solution extrême serait de fournir une copie de la clé privée à chaque application en ayant besoin.

  Cette solution impliquerait que chaque application puisse acquérir la clé privée depuis un support du type carte ou fichier. Comme cette clé doit être protégée, chaque application devrait demander à l'utilisateur la clé utilisée pour le chiffrement du composant privé, ce qui est peu pratique. De plus, si chaque application peut accéder à la clé, alors la clé privée serait exposée d'avantage à des attaques du type "Cheval de Troie".

- Une alternative serait de conserver les clés chiffrées dans un dispositif propre à l'utilisateur et de les déchiffrer lorsque l'utilisateur en a besoin.

  Dans un tel cas, il est nécessaire que des mésures adéquates soient utilisées de façon à ce que les administrateurs système ne soient pas capables de déchiffrer les clés d'utilisateurs et qu'il n'y ait pas de possibilité de perte des clés.

  Dans la majorité des solutions du type ci-dessus [WR90] [Alt92], la clé utilisée pour déchiffrer la clé privée est dérivée du mot de passe de l'utilisateur. Ainsi, dès que le mot de passe d'un usager a été découvert, la clé privée est aussi dévoilée, i.e., toute la sécurité de l'utilisateur réside sur un simple mot de passe. Or, il est nécessaire qu'un mécanisme plus fort soit utilisé pour la protection des clés privées, ou tout au moins que ce mécanisme comprenne un mot de passe indépendant.

- Une autre alternative est de conserver les clés privées sur un dispositif amovible, par exemple, une disquette.

  Dans ce cas aussi, les clés doivent être conservées chiffrées pour éviter qu'en cas de vol, l'imposteur ne puisse se faire passer par le propriétaire du dispositif. En outre, il est à la charge de l'utilisateur d'assurer que des mesures adéquates de sauvegarde soient prises.

- La solution préférable est d'enfermer la clé privée dans un dispositif physique, du type *carte à puce*.

  Cette carte doit conserver la clé privée de l'utilisateur et, possiblement une copie de son certificat et une copie du certificat de sa CA. Le détenteur de la carte reçoit un numéro, le *PIN* (***Personal Identification Number***), qui doit être maintenu secret et qui donne accès aux données conservées dans la carte. En fait, le PIN identifie l'utilisateur vis-à-vis de la carte.

  Chaque utilisateur disposerait d'une carte qu'il insérerait dans un lecteur attaché à la machine qu'il désire utiliser. Toute opération de chiffrement/déchiffrement au moyen de la clé privée doit être effectuée dans la carte et ainsi les possibilités de compromis seraient considérablement réduites.

  Bien que la technologie disponible à ce jour permete une puissance de calcul acceptable, les coûts de tels équipements sont encore élevés et leur capacité de stockage est encore reduite.

Dans la suite de cette section nous présenterons les approches retenues dans SécUcomx pour pallier aux problèmes liés au stockage et à l'accès aux clés privées.

8.3.2.1. Le Stockage des Clés Privées

Dans SecUcomx, les clés privées des utilisateurs ne sont jamais véhiculées ni fournies aux applications. Nous avons décidé d'implémenter une solution logicielle, orientée vers notre environnement de travail classique, à savoir des stations de travail basées sur le système d'exploitation UNIX.

Lorsque la génération de clés est terminée, le KS stocke la clé privée de l'utilisateur dans un fichier dans l'environnement local avec des droits de lecture exclusifs à l'utilisateur. Pour protéger la clé privée, le KS procède de la manière suivante :

(1) Ayant généré les clés RSA, nous disposons des nombres premiers p, q et n, et des exposants public e et privé d [RSA78], tels que :

$$n = pq$$

$$ed \bmod (p-1)(q-1) = 1$$

(2) Ensuite le KS obtient deux autres nombres premiers y et ey, tels que y est d'une taille de 32 bits choisi aléatoirement et :

$$ey\ y \bmod (p-1)(q-1) = 1$$

Ainsi, ey est calculé par rapport à y de la même façon que l'exposant privé d est calculé par rapport à l'exposant public e, selon l'algorithme RSA.

(3) Le nombre ey (d'une taille de 512 bits) est utilisé comme exposant de chiffrement du matériel cryptographique privé m de l'utilisateur, à travers l'operation :

$$m' = m^{ey} \bmod n$$

L'exposant ey est détruit immédiatement après le chiffrement des données privées m.

(4) Le nombre y sert donc comme exposant de déchiffrement et est livré à l'utilisateur comme étant son **PIN**.

Le déchiffrement du matériel cryptographique privé de l'utilisateur est alors effectué à travers l'operation :

$$m = m'^{y} \bmod n$$

Le PIN est, en fait, un exposant RSA de 8 digits, utilisée par l'utilisateur lorsqu'il a besoin d'accéder à sa clé privée.

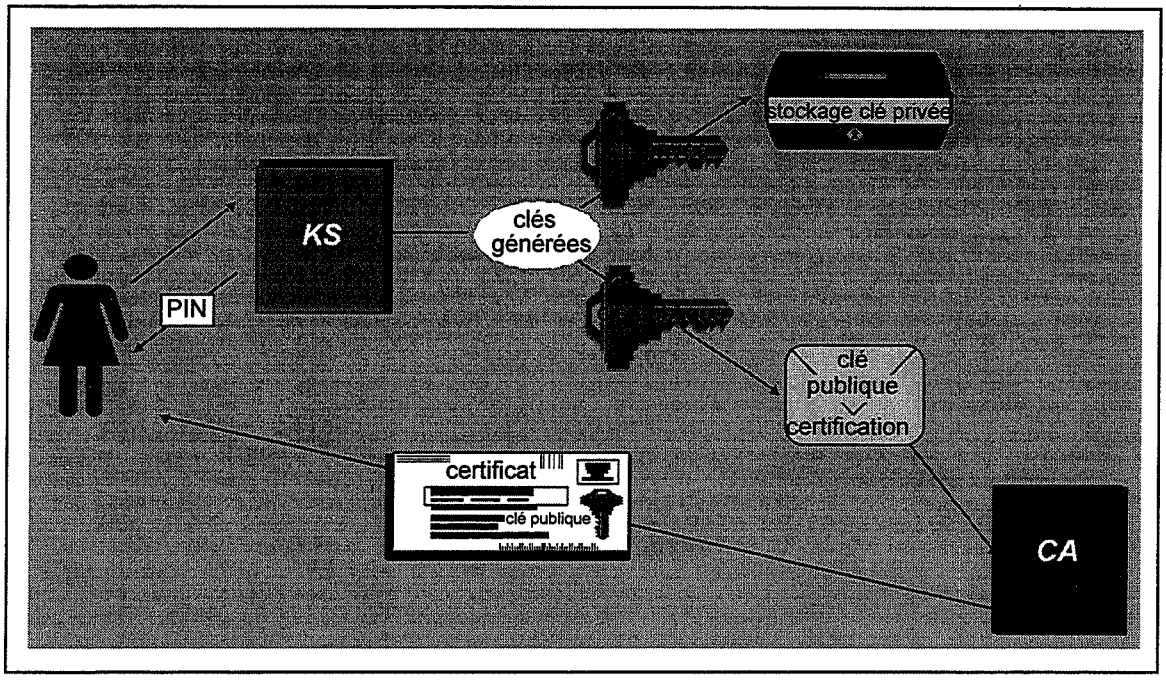

Figure 14. Génération , Stockage et Certification des Clés

L'approche adoptée pour l'accès à la clé privée sera décrite ci après.

8.3.2.2. L'Accès à la Clé Privée

L'approche de stockage de clé privée décrite ci-dessus nous a permis d'implémenter une solution logicielle pour l'accès à la clé privée des utilisateurs, émulant le fonctionnement d'une carte à puce. Le composant de l'architecture chargé de gérer l'accès à la clé privée est connu sous le nom de **GS** ou **G**estionnaire de **S**ecrets. Cette solution est inspirée du Gestionnaire de Secrets développé par Altarah comme partie intégrante du modèle Chimæra [Alt92]. Les différences entre ces deux approches sont les suivantes :

(1) Le GS-SecUcomx offre plus d'assurance tant au niveau de la protection de la clé privée qu'au niveau de l'interaction entre le GS et les processus d'application. Des détails seront donnés par la suite.

(2) Contrairement au GS-Chimæra, dans SecUcomx, le GS ne joue pas le rôle d'autorité de certification. Dans SecUcomx, les CA sont conçues pour opérer comme des entités *off-line* pour réduire les risques de compromission.

Dans notre architecture, le GS est lancé par l'utilisateur dans son environnement local. Dans sa phase d'initialisation, le GS authentifie l'utilisateur en lui demandant son PIN. Ensuite, le GS va lire en mémoire la clé protégée pour la déchiffrer à l'aide du PIN de l'utilisateur.

Pour compléter l'authentification de l'utilisateur, le GS vérifie l'exactitude de ses clés. Cela est fait en chiffrant un nombre aléatoire à l'aide de la clé publique de l'utilisateur, et ensuite en le déchiffrant au moyen de la clé privée. Si le résultat est égal à la valeur initiale, l'utilisateur est alors authentifié. Dans le cas contraire, le GS considère que le PIN fourni par l'utilisateur est erroné.

Il doit être rappelé que, contrairement au GS-Chimæra, dans SecUcomx, la sécurité de la clé privée de l'utilisateur ne dépend pas de son mot de passe conventionnel. Dans notre approche le PIN est, en fait, un nombre aléatoire qui fonctionne comme un exposant RSA, ce qui assure un bon niveau de sécurité.

Après cette étape initiale d'authentification de l'utilisateur, le GS construit le contexte à travers lequel les applications pourront lui envoyer des données à chiffrer, comme si elles avaient envoyé des données à une carte à puce. A ce stade, il faut éviter que le GS puisse être utilisé par des applications non autorisées.

8.3.2.2.1. L'interaction entre le GS et les Applications

Les demandes de chiffrement sont envoyées au GS par des applications qui, dans notre environnement, sont des tâches UNIX. Il est nécessaire d'assurer que seules les applications autorisées, c'est-à-dire celles lancées par l'utilisateur propriétaire de la clé privée, sont capables d'accéder au GS. Pour cela, nous nous appuyons, d'une part, sur les services de contrôle d'accès inhérents au système UNIX ; et d'autre part, contrairement au GS-Chimæra, sur des techniques cryptographiques.

Dans l'approche Chimæra, le GS fabrique la clé de session, choisit un numéro de port et écrit ces données en clair dans un fichier de l'environnement de l'utilisateur. Bien que ce fichier soit protégé contre la lecture, nous savons que, dans l'environement UNIX, tout autre utilisateur possédant le mot de passe de "*super user*" peut accéder à ce fichier, d'autant plus que les données qu'il contient ne sont pas elles-mêmes protégées.

Dans SecUcomx, pendant l'étape d'initialisation, le GS demande à l'utilisateur un mot de passe qui servira, à la fois comme clé de session et comme "challenge" aux applications. Ensuite, le GS choisit un port de communication sur lequel il attendra les messages à chiffrer. Le GS écrit alors ce numéro de port dans un fichier protégé de l'environnement de l'utilisateur. Ce fichier a des droits de lecture exclusifs à l'utilisateur et est chiffré symétriquement au moyen de la clé de session. Le GS choisit un port de communication diffèrent à chaque session.

En ce qui concerne les applications sécurisées, elles disposent des fonctions "*gs-client*" permettant d'initialiser l'interaction avec le GS et de réaliser le protocole de communication avec le GS pour les demandes subséquentes de chiffrement. Lorsque l'utilisateur lance son application, il doit lui fournir le même "mot de passe d'application" qu'il a fourni au GS. La fonction "*gs-client-init*" pourra alors lire le fichier contenant le numéro port choisi et chiffré par le GS, et ensuite effacer et détruire ce fichier.

Lorsqu'une application cherche à chiffrer des données avec la clé privée de l'utilisateur, elle compose un message contenant ces données et des informations supplémentaires nécessaires à l'identification de l'origine des données. L'application chiffre tout le message au moyen de la clé de session et l'envoie au GS sur le port de communication qu'il a spécifié. Le format des messages envoyés au GS est le suivant :

```
gsClientRequest ::= ENCRYPTED SEQUENCE {
    appli-name      OCTET STRING,
    object          OCTET STRING,
```

```
        time            UTCTime,
        random          BIT STRING
    }
```

Où,

- *appli-name* représente le nom de l'application de l'utilisateur,
- *object* constitue les données à chiffrer,
- *time* contient la date et heure de la requête de chiffrement,
- *random* est un nombre aléatoire destiné à empêcher les attaques par re-jeu.

A la réception de ce message chiffré, le GS le déchiffre et en vérifie le contenu. Après le déchiffrement, le GS est sûr de l'authenticité et de la validité du message reçu. Ensuite, il signale à l'utilisateur de la réception d'une telle requête, en affichant le champ "appli-name" ci-dessus, et attend la confirmation de l'utilisateur, selon le mode d'opération choisi. Les modes d'opération du GS sont décrits dans le paragraphe suivant.

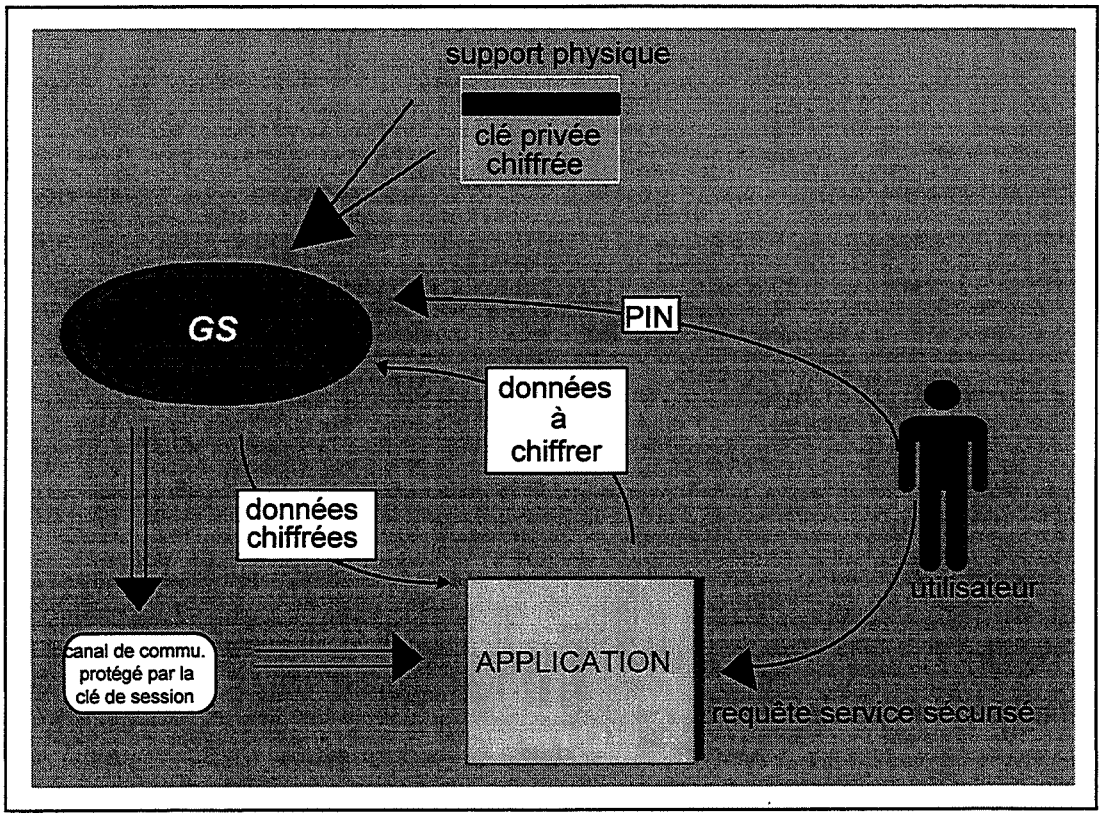

Figure 15. Le Fonctionnement du Gestionnaire de Secrets

Si l'utilisateur confirme la requête de chiffrement, le GS procède au chiffrement et renvoie à l'application client les données chiffrées au moyen de la clé privée de l'utilisateur accompagnées du nombre aléatoire reçu dans la requête. Le tout est chiffré au moyen de la clé de session. Le message de réponse a donc le format suivant :

```
gsClientResponse ::= ENCRYPTED SEQUENCE {
    encrypted       ENCRYPTED object,
```

```
        random          BIT STRING
}
```

8.3.2.2.2. Les Modes d'Opération du GS

Outre le besoin d'authentifier les applications pour empêcher que le GS puisse être utilisé par d'autres utilisateurs ou des applications non autorisées, un autre problème important reste à résoudre : il faut éviter qu'un intrus puisse profiter d'un poste de travail inutilisé pour acquérir l'identité de son propriétaire. Pour protéger l'utilisateur (et sa clé privée) de ce type d'attaque quelques mesures sont prises :

(1) L'utilisateur attribue une durée de vie au GS. Au bout de cette durée de vie paramétrée, l'utilisateur devra recommencer la phase d'initialisation et fournir son PIN à nouveau.

(2) Le GS comporte trois modes d'opération permettant à l'utilisateur de sélectionner le niveau de confirmation qu'il désire par rapport aux requêtes de chiffrement envoyées par ses applications. L'utilisateur choisit alors le mode d'opération en fonction du niveau de sécurité souhaité. Les modes d'opération sont les suivants :

- *mode check* : C'est le mode par défaut. Il détermine que le GS doit présenter chaque requête reçue à l'utilisateur, qui doit à son tour autoriser le chiffrement en rentrant son PIN. Le GS validera alors le PIN de l'utilisateur avant de donner suite au chiffrement.
- *mode confirmé* : Dans ce mode, le GS doit présenter chaque requête reçue à l'utilisateur, qui doit à son tour confirmer l'opération en répondant "oui" ou "non". Si l'utilisateur répond "oui", le GS donnera suite au chiffrement.
- *mode automatique* : Dans ce mode, le GS ne demande pas l'autorisation de l'utilisateur et donne suite à toute requête valide de chiffrement.

Le passage d'un mode plus restreint à un autre moins restreint requiert que l'utilisateur présente son PIN au GS.

(3) Le fonctionnement du GS peut être bloqué par l'utilisateur. Cela est utile lorsque l'utilisateur doit s'absenter pour un certain laps de temps et qu'il ne souhaite pas quitter le GS. Dans cet état, le GS ne répond à aucune requête de chiffrement. Pour débloquer le GS, l'utilisateur doit lui présenter son PIN pour être ainsi authentifié.

(4) Le GS garde des traces de tout son fonctionnement. Cela inclut toutes les requêtes reçues, acceptées ou non, les modes d'opération choisis, les demandes de changement de mode d'opération et de blocage/déblocage du GS. Cela permet à l'utilisateur de tenir compte du fonctionnement du GS quelque soit le mode d'opération choisi.

8.3.2.2.3. L'impact de l'Environnement sur le GS

Puisque le GS est conçu pour opérer sous l'environnement UNIX, nous devons analyser l'influence de ce système sur l'assurance du GS. En effet, les systèmes actuels intègrent des outils d'administration, tel que "*netstat*", pouvant être utilisés pour surveiller la communication entre les applications et le GS.

Le chiffrement des messages entre l'application et le GS a donc l'objectif de, à la fois, masquer les messages à un tiers écouteur et d'identifier l'expéditeur du message car la clé de session est connue seulement par l'utilisateur et par le GS. De plus, le GS vérifie l'adresse de la machine émettrice et refuse tout message émis depuis une adresse non locale.

En outre, la clé de session n'est stockée en clair nulle part dans le système et le port de communication est chiffré au moyen de la clé de session. Ainsi, même les personnes possédant le mot de passe de "*super user*" ne peuvent déterminer ni la clé de session ni le port où le GS écoute.

Bien entendu, ce schéma nécessite qu'un minimum de sécurité soit assuré sur le poste de travail. Nous supposons que les stations sont munies d'un disque local et qu'elles ne font pas de "*swap*" sur le réseau car l'analyse du fichier de *swap* peut compromettre la confidentialité des clés. De plus, l'image mémoire du GS est protégée contre les arrêts accidentels. De même, les traces du type "*core dump*" sont interdites.

8.3.3. La Gestion de Clés Publiques

Selon le cadre d'authentification X.509, les clé publiques sont gérées par les autorités de certification (CA). Nous avons donc conçu, pour SecUcomx, un élément appelé **CS** (**C**ertification **S**ystem) responsable de réaliser les services et les fonctionnalités d'une autorité de certification. Dans cette section nous abordons le concept, le rôle et les fonctionnalités des CA ainsi que l'approche mise en œuvre dans SecUcomx.

8.3.3.1. Le Concept de CA

Une Autorité de Certification, CA, est un agent auquel l'on confie la tâche de certifier les clés publiques d'un groupe d'utilisateurs. La CA fonctionne ainsi comme un service d'introduction pour les utilisateurs vis-à-vis d'un environnement sécurisé. Apres s'être assurée de l'identité d'un utilisateur X, la CA atteste que la clé publique de X appartient vraiment à X. Cette attestation prend la forme d'un certificat signé au moyen de la clé privée de la CA, conformément à la description de la Section 5.1 de la Partie II.

Une CA doit donc s'assurer de l'identité d'un utilisateur avant de créer un certificat à son intention. De plus, il importe que le transfert d'information à la CA ne soit pas une source de compromission. Il convient que des mesures de sécurité appropriées soient prises. Par exemple, ce serait un grave manquement à la sécurité, si la CA livrait un certificat à un utilisateur ayant une clé publique qui aurait été falsifiée.

Vu la complexité et l'évolution croissante des algorithmes de chiffrement, il est attendu que, en pratique, les sources de compromission et les cibles d'attaque des systèmes de sécurité se situeront, plutôt au niveau de la gestion de clés qu'au niveau des algorithmes de chiffrement eux-mêmes. Par conséquent, le niveau d'assurance des méthodes de gestion de clés constitue un facteur très important.

8.3.3.2. Les Services d'une CA

Dans les systèmes ouverts, les services d'authentification dépendent, entre autres, d'un ensemble d'informations qui doivent être publiquement disponibles. Ces informations correspondent aux clés publiques des entités impliquées dans le système, à savoir, les certificats des utilisateurs, des CA et les listes de révocation de certificats. Tous ces éléments sont signés et gérés par les autorités de certification.

La fonctionnalité d'une CA est conçue de façon à ce que son action la plus sensible - la signature des certificats - puisse être effectuée de façon isolée du réseau (*off-line*). Une fois signés, les certificats et les listes de révocation peuvent être conservés dans des serveurs de noms pour être largement disponibles. Les services d'une CA comprennent :

* **L'établissement de la CA locale**

  Cela comprend la génération de la paire de clés et du certificat de la CA elle-même, et des structures de données nécessaires à son fonctionnement.

* **La certification d'autres CA**

  Cela comprend l'acceptation d'une autre CA dans le système et implique une délégation d'autorité selon l'accord sur une politique de securité.

* **La certification des utilisateurs**

  Cela comprend la preuve de l'identité de l'utilisateur, que sa clé publique n'a pas été falsifiée et qu'il n'existe pas un autre utilisateur ayant la même identification.

* **La certification croisée (ou mutuelle)**

  Il s'agit de l'action à travers laquelle deux CA expriment leur confiance mutuelle et leur accord bilatéral en ce qui concerne le niveau d'assurance offert et la politique de certification adoptée. Cela permet l'interfonctionement entre des utilisateurs appartenant à des domaines de sécurité différents.

  Pour la certification croisée, chacune de deux CA certifie la clé publique (déjà certifiée) de l'autre formant une paire de certificats croisés. Ces certificats croisés sont conservés dans les entrées des deux CA.

* **La révocation de certificats**

  Cela consiste à annoncer l'annulation de certificats à travers la maintenance d'une liste signée par la CA.

Pour que ces mécanismes de certification puissent offrir une sécurité effective, l'information à être signée doit être délivrée à la CA d'une manière fiable, i.e., l'authenticité et l'intégrité de ces informations doivent être garanties. La conception de procédures de certification et la mise en œuvre d'un tel niveau d'assurance sont hors du contexte de normalisation. La suite de cette section décrit les mécanismes de gestion d'autorités de certification que nous avons conçu pour SecUcomx.

8.3.3.3. Procédures de Certification

Lorsqu'un utilisateur ou une organisation s'enregistre auprès d'une CA, celle-ci peut exiger certaines informations nécessaires à l'identification de l'utilisateur ou de l'organisation. L'échange de clés entre un utilisateur et une CA, ou entre deux CA, doit être effectué en utilisant des moyens sûrs. Cet échange doit garantir l'authenticité et l'intégrité de ces informations pour éviter :

(1) qu'un intrus puisse s'enregistrer en utilisant l'identité de quelqu'un d'autre et que la clé publique ne soit pas modifiée pendant son transfert,

(2) qu'une CA ne délivre de certificat pour deux utilisateurs différents ayant un même nom d'annuaire.

Dans les deux cas ci-dessus, cela constituerait une atteinte non seulement à la sécurité des procédures d'authentification mais aussi à la confidentialité des échanges, car l'intrus (dans le cas (1)) et chacun des deux utilisateurs (dans le cas (2)) serait capable de se servir de l'identité de l'autre et de déchiffrer des données confidentielles destinées à l'autre.

Comme nous l'avons déja mentionné, il est également important que la clé publique de la CA soit transférée à l'utilisateur de manière sûre. L'utilisateur doit s'assurer que cette clé provient bien de sa CA (et non d'une CA pirate), pour qu'il ne soit pas trompé par des certificats falsifiés lors des futures procédures d'authentification. Ainsi, cette clé doit être protégée quant à l'intégrité et l'authenticité.

La définition de procédures de ce type n'est pas prise en considération par les standards OSI. Nous avons conçu des procédures de certification prenant en compte les aspects mentionnés ci-dessus. Ces procédures sont décrites dans la suite de cette section.

L'unicité des noms d'annuaire

Comme nous l'avons déjà mentionné ci-dessus, une CA doit s'assurer que deux utilisateurs n'auront pas des certificats ayant le même nom d'annuaire. Dans le cas où les utilisateurs sont certifiés par une même CA, celle-ci peut contourner ce problème en vérifiant, dans sa base locale, les noms d'annuaire pour lesquels (et pour qui) des certificats ont été émis. Si deux personnes appelées "José da Silva" demandent un certificat à la même CA, celle-ci peut détecter la différence en comparant la preuve d'identité fournie par ces utilisateurs (numéro de passeport, des photographies, etc.). La CA peut alors exiger qu'ils utilisent des noms d'annuaires différents.

En contrepartie, il est plus difficile d'assurer que deux CA différentes n'émettront pas des certificats pour des utilisateurs ayant le même nom d'annuaire. Lorsque les CA ont juridiction sur des parties disjointes de l'espace de noms, celles-ci n'ont pas besoin de se consulter entre elles lors de l'émission de certificats. Dans le cas contraire, la possibilité de conflit reste. La politique de sécurité du modèle de distribution de CA doit prendre cet aspect en considération. Par exemple, dans les approches Password et PEM, les CA chargées de la certification des utilisateurs finals ont juridiction sur des parties disjointes de l'espace de noms. Accessoirement, dans PEM la CA de plus haut niveau (l'IPRA) maintient une base de données globale permettant de détecter une éventuelle duplication de noms de CA dans la hiérarchie.

Le même danger se présente lors de la réutilisation de noms d'annuaire, i.e., lorsqu'une autorité de désignation reattribue un nom d'annuaire à un utilisateur différent. Pour répondre à ce problème, la version 1994 de la recommandation X.509 fournit une nouvelle version de format de certificat contenant des identificateurs uniques pour les utilisateurs et pour les CA. Les CA peuvent alors faire usage de ces identificateurs pour distinguer entre des instances réutilisées des noms d'annuaire.

8.3.3.3.1. La Certification des Utilisateurs

Dans l'approche SecUcomx, tout utilisateur dispose d'un module, le KS, chargé de générer ses paires de clés asymétriques. Suite à la génération des clés, la clé publique de l'utilisateur doit être soumise à la CA. L'utilisateur doit alors présenter son composant public et son identité à la CA pour l'inclusion dans le certificat.

La CA doit s'assurer de l'identité prétendue par l'utilisateur et que la clé publique fournie est véritablement associée à l'utilisateur identifié par les informations présentées. A ce stade, le problème qui se pose est qu'une signature numérique ne peut pas prouver l'authenticité, car les parties n'ont pas encore échangé des clés. Par contre, la signature peut prouver que la clé publique est valide et que la clé privée correspondante est connue par quelqu'un (bien que cela ne preuve pas "qui" la connaît).

8.3.3.3.1.1. Le Message de Demande de Certification

Dans notre architecture nous avons conçu une solution permettant aux utilisateurs d'envoyer leur demande de certification à la CA en utilisant une méthode "*store and forward*". Dans notre approche, la CA exige que cette demande lui soit présentée sous la forme d'un *message de demande de certification*. Ce message inclut l'identification et la clé publique de l'utilisateur.

Le message demande de certification doit être protégé contre la modification pendant son transfert jusqu'à la CA. De plus, ce message doit fournir une "preuve d'identité". Ainsi, l'intégrité et l'authenticité doivent être garanties. Pour atteindre cet objectif ce message contient trois composants :

* la clé publique de l'utilisateur,
* un segment d'authentification,
* la signature de l'utilisateur.

La structure de ce message peut être représentée en ASN.1 sous la forme suivante :

```
CertificationRequestMessage ::= SIGNED SEQUENCE {
    publicKey       [0]  SubjectPublicKeyInfo,
    authSegm        [1]  ENCRYPTED AuthenticationSegment,
    entitySignature [2]  SIGNATURE SignatureData
}
```

L'élément *publicKey* contient la clé publique de l'utilisateur dans le format spécifié par la norme X.509, i.e., à travers la structure ASN.1 *SubjectPublicKeyInfo*, qui est spécifiée par la structure suivante :

```
SubjectPublicKeyInfo  ::= SEQUENCE {
      algorithm                AlgorithmIdentifier,
      subjectPublicKey         BIT STRING
}
```

L'élément *authSegm* constitue un segment d'authentification contenant le texte en clair de la signature de l'utilisateur, ses données d'identification et un numéro aléatoire. Ce segment d'authentification est chiffré avec la clé publique de la CA de sorte qu'elle est la seule à pouvoir déchiffrer cet élément.

```
AuthenticationSegment ::= SEQUENCE {
      signData   SignatureData,
      userName   EntityName,
      userPass   EntityPassword,
      signKey    BIT STRING
}

EntityName ::= OCTET STRING(SIZE(0..ub-entity-name))

EntityPassword ::= OCTET STRING(SIZE(0..ub-entity-pass))

ub-entity-name INTEGER ::= 64

ub-entity-pass INTEGER ::= 64
```

- L'élément *signData* contient les données, en clair, qui ont été signés par l'utilisateur au moyen du composant privé de sa paire de clés.

```
SignatureData ::= SEQUENCE {
      date       INTEGER,
      random     BIT STRING
}
```

- Les données d'identification à savoir, *userName* et *userPass*, peuvent être établis par accord préalable avec la CA ou peuvent être le nom de login et mot de passe UNIX de l'utilisateur.
- L'élément *signKey* sert à fournir une clé pour la vérification de la signature globale du message. De plus, puisque cette clé est constituée dans nombre aléatoire, elle sert également à "brouiller les pistes", i.e., protéger le segment contre une éventuelle attaque par déduction.

L'élément *entitySignature* constitue la signature de l'utilisateur à partir des données *SignatureData* ci-dessus. Cette signature est produite au moyen de la clé privée de l'utilisateur et sert à prouver que celui-ci est en possession du composant privé correspondant à la clé publique qu'il prétend certifier.

Finalement, afin d'assurer l'intégrité des données transférées à la CA, tout le message est signé. Comme nous l'avons décrit ci-dessus, l'élément *signKey*, du segment d'authentification, fournit la clé pour la vérification de la signature. La procédure de calcul de cette clé est égale à celle utilisée pour le calcul du PIN de l'utilisateur (voir 8.3.2.1). Son inverse *mod (p -1) (q -1)* est

utilisé pour chiffrer la signature du message. Alternativement, cette clé peut être une clé symétrique.

Puisque cette clé est protégée dans le segment d'authentification chiffré, ce message demeure infalsifiable. De plus, la CA de l'utilisateur est la seule à pouvoir déchiffrer ce segment puisqu'il est chiffré avec sa clé publique.

Accessoirement, ce message peut être imprimé sur papier pour que l'utilisateur puisse y attacher une signature manuscrite. Sur papier, le message est accompagné du résultat de son scellement en clair pour que la CA puisse le vérifier. L'utilisateur peut alors fournir cela à la CA avec d'autres preuves d'identité conforme demandé par la politique de sécurité de la CA en question. Ces preuves peuvent être un passeport, un permis de conduire, etc.

8.3.3.3.1.2. Vérification du Message Demande de Certification

A la réception du message électronique, la CA procédera à sa validation. La procédure de validation du message est la suivante :

(1) La CA déchiffre le segment d'authentification (*authSegm*) au moyen de sa clé privée de manière à obtenir l'identitification de l'utilisateur (*userName* et *userPass*) et les paramètres nécessaires à la vérification des signatures du message (*userSign* et *signKey*).

(2) Ensuite, la CA vérifie la signature de l'utilisateur (en utilisant les données *userSign* extraites du segment d'authentification) et la signature du message (en utilisant la clé *signKey*).

L'étape de validation peut détecter tout problème d'intégrité ou d'authenticité du message. Si la politique de sécurité de la CA l'exige, celle-ci validera les documents sur papier supplémentaires et vérifiera le résultat de scellement obtenu électroniquement avec celui fourni par le message d'enregistrement sur papier. Si la vérification réussit, la CA produira le certificat de l'utilisateur et le publiera dans l'entrée d'annuaire de l'utilisateur. Pour ce faire, la CA suit les étapes suivantes :

(1) La CA se connecte au DSA local en utilisant le protocole d'accès à l'annuaire (DAP) et recherche l'entrée de l'utilisateur dans son domaine de certification utilisant le nom de l'utilisateur fournit dans le segment d'authentification.

(2) Si l'entrée de l'utilisateur a pu être trouvée, la CA met cette entrée à jour avec le certificat de l'utilisateur.

(3) Finalement, la CA envoie un message à l'utilisateur, lui notifiant des résultats/erreurs de l'opération. Le contenu de ce message comprend une copie des certificats de l'utilisateur et de la CA.

Pour protéger sa clé publique, la CA envoie à l'utilisateur le résultat de scellement de son certificat, chiffré au moyen de la clé *signKey* reçue dans le segment d'authentification, et qui, à ce stade, constitue un secret partagé entre la CA et l'utilisateur.

La CA conserve une copie du message d'enregistrement comme preuve de non-répudiation au cas où l'utilisateur essaierait de démentir la demande de certification.

8.3.3.3.1.3. Certificats d'Utilisateurs Non-Humains

Dans de nombreux protocoles OSI, des processus (comme des DSA ou MTA) possèdent ses propres paires de clés, et donc des certificats utilisés pour leur identification. Dans de tels cas, le certificat identifie plutôt un service qu'une entité ou une personne.

Plusieurs des vérifications qu'une CA effectuerait pour certifier un utilisateur humain (par exemple, la présentation d'un passeport ou d'un permis de conduire) ne sont pas applicables lors de la certification d'un processus. Au lieu de cela, la CA doit exiger l'évidence qu'une organisation (ou personne physique) s'engage à fournir le service et à accepter la responsabilité des éventuels défauts de sa réalisation.

L'étendue des engagements demandés au fournisseur du service dépend de la politique de sécurité sous laquelle la CA opère. La politique de distribution de clés doit prendre ce cas en compte explicitement. Accessoirement, des CA spécifiques peuvent être désignées pour la certification des processus. Cela indiquerait aux utilisateurs de ces certificats que la sémantique en vigueur est différente.

8.3.3.3.2. Certification d'une CA

L'introduction d'une nouvelle CA dans le système et la certification mutuelle entre deux CA requièrent la participation directe et l'accord entre les deux CA, celle qui certifie et celle qui désire obtenir un certificat. Cet accord dépend de la politique de sécurité en vigueur. Par exemple, une organisation offrant le "service d'établissement de CA" pourrait exiger que les organisations qui prétendent offrir le service d'autorité de certification, lui soumettent leur politique de sécurité pour approbation. Cela dépend également du modèle de confiance dans lequel l'organisation désire s'insérer, car le modèle en question peut établir un ensemble de règles générales applicables à toutes les CA dans le modèle (comme par exemple, dans le modèle PEM).

Dans SecUcomx nous avons décidé de ne pas définir un modèle de confiance propriétaire mais de concevoir une architecture assez flexible de façon à pouvoir s'insérer dans tout modèle compatible avec l'approche X.509 avec un niveau d'assurance satisfaisante.

Nous avons conçu une procédure de certification de CA qui requiert la participation directe entre les CA (en fait entre les opérateurs responsables des CA). En effet, cette procédure consiste en le même échange d'informations que décrit ci-dessus hormis la structure du segment d'authentification, qui dans ce cas a le format suivant :

```
AuthenticationSegment ::= SEQUENCE {
    CASign          SignatureData,
    CAName          DistinguishedName,
    CAManagerName   DistinguishedName,
    CAPassword      EntityPassword,
    signKey         BIT STRING
}
```

Où,

- L'élément *CASign* est défini comme auparavant,
- L'élément *CAName* inclut le nom d'annuaire de la CA prétendante,

- L'élément *CAManagerName* correspond au nom d'annuaire du responsable de la CA postulante,
- L'élément *CAPassword* est un secret partagé entre les deux parties et qui doit être établit précédemment par des moyens sûrs,
- L'élément *signKey* est défini comme auparavant.

Ainsi, cette procédure exige la participation directe entre les CA, dans la mesure où un secret partagé doit être établi. Il est à la charge de l'organisation de définir les conditions sous lesquelles elle acceptera de certifier d'autres CA. Les éléments *CASign* et *signKey* sont définis de la même façon que leurs correspondants dans la procedure définie pour la certification des utilisateurs.

8.3.3.4. La Révocation de Certificats

La production d'un certificat implique l'engagement d'une autorité de certification qui atteste la validité des informations contenues dans ce certificat. Toutefois, ces informations peuvent devenir invalides après un laps de temps probablement indéterminé. Par exemple :

* le changement du nom d'annuaire de l'utilisateur, par exemple, suite à un changement d'affiliation organisationnelle,
* l'utilisateur peut vouloir obtenir une nouvelle paire de clés à cause d'une perte, ou de la compromission de sa clé privée ou de son PIN,
* l'algorithme de signature du certificat peut changer dans le cas où une faiblesse de l'algorithme utilisé auparavant a été constatée,
* l'utilisateur ou la CA peuvent demander une augmentation de la taille de clés ou le changement d'algorithme de chiffrement en conséquence du progrès de la cryptanalyse et/ou de l'augmentation des capacités de calcul.

8.3.3.4.1. Procédure de Demande de Révovation d'un Certificat

La procédure utilisée pour la demande de révocation des certificats ne peut pas être unique. Cette procédure doit inévitablement dépendre de la raison pour laquelle la révocation est souhaitée, car :

(1) si le nom de l'utilisateur change ou s'il désire remplacer sa paire de clés par une autre de plus grande assurance, il est souhaitable d'assurer l'intégrité et l'authenticité de ces requêtes, de façon à éviter qu'un attaquant puisse provoquer la révocation du certificat de quelqu'un d'autre,

(2) si la clé privée de l'utilisateur à fait l'objet de compromission, évidemment cette clé privée ne peut plus prouver son identité ; la procédure (1) ci-dessus ne peut donc pas être appliquée,

(3) de même, si la clé privée ou le PIN de l'utilisateur est perdu, l'utilisateur n'est plus capable de produire une requête signée.

Cependant, dans les cas (2) et (3), l'utilisateur serait encore capable de produire une requête signée de révocation s'il dispose d'une autre paire de clés dont le composant public est certifié avec le même nom d'annuaire.

De toutes façons, normalement l'utilisateur désire que sa clé soit révoquée le plus tôt possible. De plus, il peut être géographiquement distant d'un équipement à partir duquel il puisse signer la requête pour l'envoyer à la CA. Par conséquent, la CA (en fait son responsable) doit être capable d'accepter des requêtes manuscrites.

De même, la CA doit être capable d'informer le changement de ses clés à ses clients. Ayant considéré les sites opérateurs de CA comme bien protégés, la compromission de la clé privée de la CA doit être un événement rare. Ainsi, les raisons typiques pour le changement des clés d'une CA seront l'augmentation de la taille des clés ou l'adoption d'un algorithme cryptographique plus fort. Dans de tels cas, l'ancienne clé publique ne nécessite pas d'être révoquée immédiatement et la CA peut maintenir des certificats émis au moyen des deux clés durant la période de transition.

8.3.3.4.2. Gestion de Listes de Révocation

Toute CA doit émettre une nouvelle liste de révocation à des intervalles réguliers. Cela doit être fait même si aucun certificat n'a été révoqué dans la période et sert à assurer aux utilisateurs et aux applications que la liste récupérée est la plus récente.

Lors de la vérification de certificats, les applications doivent comparer la date-heure prévue pour mise à jour de la liste avec la date-heure courant. Si la liste est périmée, l'application doit récupérer une version plus récente de la liste, selon le niveau d'assurance exigé. Si cela échoue, le certificat peut être considéré comme invalide.

Dans SecUcomx nous recommandons que les listes soient remplacées un peu avant la date d'expiration de leur précédante pour que des procédures d'authentification ne soient pas amenés à échouer à cause de problèmes opérationnels de remplacement.

La fréquence de mise à jour des listes de révocation est définie par la politique de sécurité sous laquelle la CA opère. En contrepartie, cela doit prendre en compte le fait que la taille des listes peut avoir un impact défavorable sur la performance du service de noms. Pour minimiser cet impact, les certificats révoqués peuvent être supprimés de la liste dès que leur date d'expiration soit atteigne. La gestion de certificats doit également prendre cet aspect en considération. Si une longue validité est attribuée aux certificats, lorsque révoqués ceux-ci devront rester longtemps sur la liste, ce qui rend sa gestion laborieuse. A l'inverse, si les certificats sont émis avec une durée de vie excessivement courte, la CA devra dépenser un montant inacceptable de ressources et de temps pour le remplacement de certificats.

Dans notre architecture, la fréquence de mise à jour des listes est paramétrable pour pouvoir comporter une politique de sécurité quelconque. Le projet Password, n'a pas défini une fréquence de mise à jour de listes de révocation. Dans le cas particulier de SecUcomx, du fait du nombre limité d'utilisateurs pendant les tests pilotes, notre choix a été d'émettre une nouvelle liste à chaque mois.

Dans tous les cas, une liste doit être remplacée avant sa date d'expiration aussitôt que la CA reçoit une demande de révocation d'un certificat. De plus, si des services de non-répudiation doivent être rendus, les anciennes listes doivent être conservées pour une eventuelle vérification en cas de litige.

8.3.3.4.3. Format de Listes de Révocation

Des déficiences ont été constatés au niveau de la sécurité et de la syntaxe du format des listes défini par le cadre X.509 (voir 5.1). Ainsi, dans Password et aussi dans SecUcomx, nous avons décidé d'adopter le format de listes de révocation défini par PEM. Ce choix présente les avantages suivants :

* le format de liste PEM ne présente pas les redondances de celui défini par X.509. En outre, dans PEM une seule liste est utilisée tant pour les certificats de CA que pour ceux des utilisateurs. Cela résulte en la minimisation d'effort de codage et de ressources nécessaires à l'obtention et le stockage des listes dans le service de noms.
* le format de liste PEM contient l'indication de la date prévue pour la prochaine mise à jour de la liste. Cette indication n'est pas présente dans les listes conformes à X.509. Cela permet d'indiquer une date à partir de laquelle la liste doit être considérée invalide. Bien entendu, cette solution ne permet pas d'assurer complètement aux applications que la liste récupérée est la plus récente. Nous proposons une solution à ce problème dans la Partie IV de ce travail.

8.3.3.5. La Protection de la CA

Une Autorité de Certification ne peut remplir son rôle si elle n'est pas protégée de la compromission et du mauvais usage. Les services d'une CA ne sont activés que lors de la production ou de la révocation de certificats. Par conséquent, les CA ne nécessitent pas d'être disponibles en permanence. Pour garantir une sécurité effective, les CA peuvent être mises en place sur des machines isolées, où ses clés aussi bien que ses supports matériel et logiciel peuvent être verrouillés quand la CA n'est pas active.

Dans le cas où la CA doit interagir avec l'annuaire ou un autre service de noms, pour stocker des certificats d'utilisateur ou pour récupérer des certificats de CA pour la certification mutuelle, une solution serait de doter la CA d'une ligne de communication à travers laquelle la CA pourrait se connecter au service mais jamais accepter de demandes de connexion. L'intention est de réduire le plus possible les chances d'attaque contre la CA à travers le réseau.

Dans notre approche, les CA sont des agents "*off-line*". La production des certificats est implémentée comme une opération indépendante et non pas au moyen d'un mécanisme à question/réponse automatique. Cela évite la possibilité d'attaque par le réseau. Un autre avantage de cette méthode est que la clé privée de la CA n'est jamais connue si ce n'est de la CA isolée et matériellement sûre. Ainsi, le secret de la clé privée ne peut être percé que par une attaque contre la CA elle-même, ce qui rend sa compromission difficile.

8.4. L'Obtention de Données pour l'Authentification

La réalisation des services de sécurité, basés sur des crypto-systèmes asymétriques, exige la récupération et la vérification de clés publiques entre les parties communicantes, afin que celles-ci puissent s'authentifier. X.509 dit que l'obtention de l'information d'authentification d'un partenaire d'une communication à partir de l'annuaire est, avec cette approche, semblable à l'obtention d'une addresse. En pratique, cette procédure n'est pas si simple, comme nous le verrons par la suite.

On valide un certificat en vérifiant la signature générée par l'émeteur. L'usage de certificats transforme ainsi le problème d'obtention de la clé publique d'un utilisateur en un problème d'obtention de la clé publique de l'émetteur du certificat de cet utilisateur. Puisque les CA possèdent aussi des certificats, la procédure de vérification est récursive et définit implicitement un graphe de certification. Comme cela a été déjà mentionné, un chemin de certification forme une chaîne continue de points de confiance entre deux utilisateurs souhaitant s'authentifier. L'objectif de l'obtention et de la vérification des chemins de certification est de permettre à une entité de déterminer, de façon fiable, la clé publique (et eventuellement d'autres attributs de sécurité) d'une autre entité pour une utilisation future dans des procédures de sécurité.

La vérification d'un certificat peut ainsi demander la récuperation et la vérification de plusieurs autres certificats, éparpillés partout dans le monde et contenus dans plusieurs serveurs de noms. En outre, la vérification d'un certificat implique la récupération et la consultation des listes de révocation des CA pour assurer qu'aucun des certificats en question n'a été annulé. Ces procédures peuvent être réalisées avec plus ou moins de rigueur selon la politique de sécurité en vigueur.

Ce raisonnement s'applique à toute partie impliquée dans des procédures de sécurité, i.e., les serveurs et les agents utilisateurs de toutes sortes. Ainsi, les applications faisant usage des certificats pour leurs procédures de sécurité doivent disposer d'outils pour l'établissement et la vérification de chemins de certification. Outre la vérification des signatures et des listes de révocation, ces outils doivent également vérifier si les chemins de certification obéissent aux politiques de sécurité en vigueur.

Dans l'environnement des applications sécurisées interconnectées, il est attendu que le problème d'accès aux données nécessaires aux mécanismes d'authentification devienne une question suscitant un intérêt croissant. En effet, l'obtention de données pour l'authentification constitue une tâche assez complexe car cela requiert :

- la connaissance et le respect des modèles adoptés pour la distribution des CA,
- la connaissance et le respect de la politique de sécurité associée à l'application en question,
- plusieurs accès au service de noms,
- un effort non négligeable de vérification de données cryptographiques.

En l'absence de mécanismes globaux et spécifiques à l'accomplissement de telles tâches, les applications sécurisées sont forcées de les réaliser par leurs propres moyens. Or, cela ne reflète pas une approche modulaire car dès que le moindre détail concernant les données

d'authentification est modifié, ce qui est très probable dû à l'état récent de la technologie dans le domaine, une partie considérable de l'implémentation doit être modifiée.

Nous considérons que ces aspects sont fortement liés aux architectures de sécurité, plutôt qu'aux applications. Ainsi, l'un des facteurs le plus motivant de l'élaboration de notre architecture a été la conception d'un Serveur de Données pour l'Authentification, dit **DAS** (**D**ata **A**uthentication **S**erver). Le DAS est un élément de l'architecture de sécurité commun à toutes les applications sécurisées, chargé d'obtenir et de valider les chaînes de certificats nécessaires aux procédures d'authentification.

Cette section décrit le DAS tel comme il a été initialement conçu. Toutefois, le problème de l'obtention de données pour l'authentification nous a poussé à rechercher l'évolution du concept du DAS vers la conception d'une approche qui permette la mise en place d'une infrastructure globale pour les services de sécurité. Nous abordons cette évolution dans la Partie IV de ce travail.

8.4.1. Diversité contre Interopérabilité ?

A l'heure actuelle, les données d'authentification existent sous de nombreux formats ce qui pose d'évidents problèmes d'interopérabilité.

Dans le contexte des applications ouvertes sécurisées, il existe au moins trois types de structures de données actuellement définies pour représenter des chaînes de points de confiance. En fait, ces différentes structures (ou "*paths*") constituent des variations sur le même thème, à savoir, la chaîne de certificats. Ces variations sont introduites par des différentes applications. En pratique, les applications sécurisées introduisent ces chemins comme un moyen de permettre à un utilisateur de fournir à d'autre(s) utilisateur(s), des données nécessaires à son authentification.

- **Certification Path**

  Cette structure contient une chaîne complète de certificats permettant l'authentification de la clé publique d'une entité [X.509]. Ce type de donnée est utilisé essentiellement dans des contextes du type mono-destination, telles que le protocole d'accès à l'annuaire X.500. Cette structure des données a actuellement le format suivant :

```
CertificationPath ::= SEQUENCE {
    userCertificate          Certificate,
    theCACertificates SEQUENCE OF CertificatePair }

CertificatePair   ::=   SEQUENCE {
    forward   [0] Certificate OPTIONAL,
    reverse   [1] Certificate OPTIONAL
    -- at least one of the pair must be present --}
```

- **Certificates**

  Cette structure contient les certificats de toutes les CAs supérieures d'une entité souhaitant s'authentifier (par exemple, le signataire d'une portion quelconque d'information) [X.411]. Ce type de donnée ne représente pas une chaîne complète de certificats entre deux entités qui désirent s'authentifier, mais les niveaux de l'arbre de certification d'un signataire.

```
Certificates ::= SEQUENCE{
        certificate          Certificate,
        certificationPath  ForwardCertificationPath OPTIONAL}

ForwardCertificationPath ::= SEQUENCE OF CrossCertificates

CrossCertificates ::= SET OF Certificate
```

Chaque niveau du "*ForwardCertificationPath*" représente une CA et peut contenir tous les certificats de cette CA. Puisque cette structure de données a été définie pour des applications basées sur le cadre X.509 (qui suggère une distribution hiérarchique de CA), le dernier niveau de la chaîne contient le(s) certificat(s) appartenant à la CA la plus proche de la racine du DIT.

Ce type de donnée est utilisé essentiellement dans des contextes du type multi-destination, tel que les protocoles de messagerie électronique. Dans un tel contexte, le signataire *S* d'un message établirait son *ForwardCertificationPath* et l'enverrait avec son message à tous les destinataires. A la réception du message, chaque destinataire *D* compléterait le ForwardCertificationPath reçu avec son propre "*reverse certification path*" afin d'obtenir une chaîne de complète de certificats (D → S) qui permettrait l'authentification de l'identité de *S*.

- **PEM Forward Certification Path**

  Cette structure a été définie pour la messagerie sécurisée Internet [Ken93a]. Elle comporte une syntaxe identique à celle de la structure *CertificationPath* décrite plus haut et une sémantique similaire à celle définie pour la structure *ForwardCertificationPath* décrite ci-dessus. Dans le cas présent, chaque niveau du champ "*theCACertificates*" représente une CA dans l'arborescence de certification de l'utilisateur mais ne peut contenir qu'un certificat de cette CA. Le dernier élément de la chaîne contient le certificat appartenant à la CA la plus proche de la racine du DIT, qui dans le modèle de certification PEM correspond au certificat de la CA IPRA.

Un autre problème qui se pose est celui des modèles de certification. A l'heure actuelle, il existe plusieurs modèles de certification, comportant différents types de CA, dont chacun peut jouer un rôle différent (comme PEM, PASSWORD [Roe92], Chimæra [Alt92]) et même des modèles où le concept d'autorité de certification n'existe pas (comme PGP [Zim93] ou RIPEM [Rio92]).

Par conséquent, bien qu'actuellement de nombreuses applications fournissent des services de sécurité, la plupart ne permettent pas une interopérabilité comparable à celle d'un standard OSI. L'interopérabilité comme la sécurité ont un coût qui peut paraitre prohibitif pour la plupart des applications.

Toute cette diversité de modèles de certification et de formats de données, ainsi que leurs possibilités d'évolution dans un avenir proche, demande une approche modulaire et évolutive. Il devient judicieux d'élaborer des mécanismes capables de construire et de vérifier des chemins de certification à partir de la structure de base : le certificat. Actuellement, les procédures d'obtention et de vérification de ces données publiques nécessaires à l'authentification sont encapsulées dans les applications. Une approche modulaire doit isoler ces procédures de façon à alléger les applications et à faciliter la migration et l'évolution des architectures de gestion de clés.

8.4.2. Le DAS

Le rôle du DAS est de traiter les informations publiques d'authentification tels que les certificats ou les chemins de certification. La construction, vérification et le traitement de ces informations de sécurité sont complexes. La sécurisation des applications se trouve grandement facilitée si cette complexité est déportée dans une entité indépendante des applications comme le DAS.

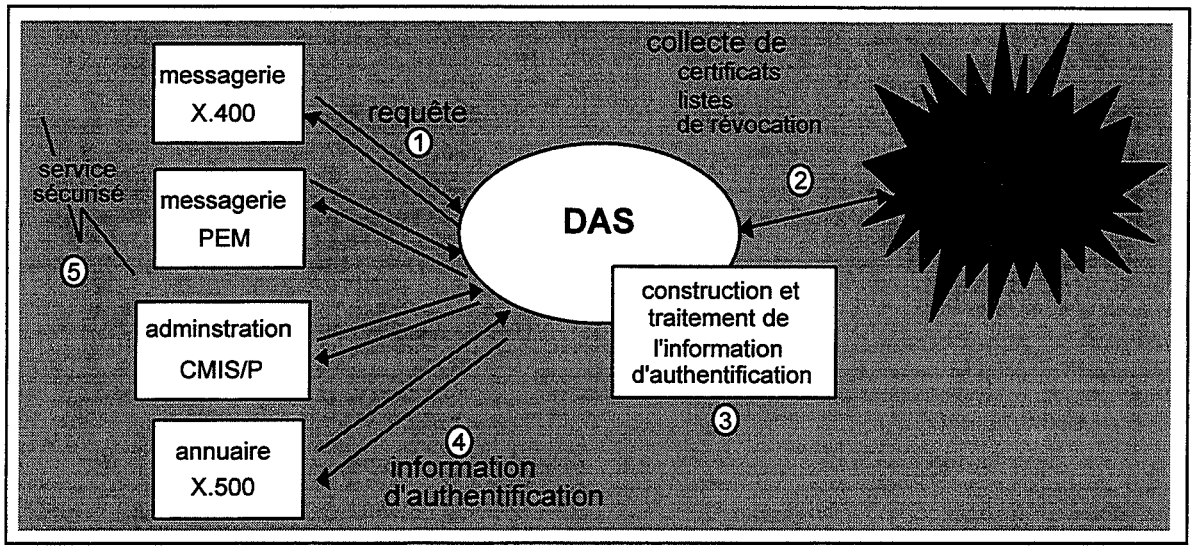

Figure 16. Le DAS

Le DAS constitue un composant indépendant de l'architecture de sécurité pouvant être utilisé par toute application pour la construction de ses propres procédures de sécurité, si celles-ci sont basées sur l'utilisation de certificats.

8.4.3. Services et Modèle Fonctionnel du DAS

Une exigence de base de la fonctionnalité du DAS est qu'elle soit proche des besoins des utilisateurs, et qu'elle réponde aux besoins des mécanismes d'authentification existants. Dans un premier abord, nous avons envisagé plusieurs services vis-à-vis des types de données existants :

- Etablissement d'un CertificationPath : ayant comme entrée le nom d'annuaire de deux entités (A et B), ce service construit des chemins de certification entre A et B (A→B and B→A), ou retourne une indication d'erreur.

- Vérification d'un CertificationPath : ayant comme entrée un chemin de certification, ce service vérifie ce chemin et retourne le résultat.

- Construction d'un Certificates : ayant comme entrée le nom d'annuaire d'une entité, ce service construit une structure de données du type "Certificates" (certificat de l'entité et forwardCertificationPath) relative à cet entité.

- Vérification d'un Certificates : ayant comme entrée une structure de données du type "Certificates" relative à l'entité A et le nom d'annuaire du récepteur B de cette donnée, ce service obtient la chaîne de certificats B → A vérifiée, ou retourne une indication d'erreur.

- Construction d'un PEM Forward Certification Path : ayant comme entrée le nom d'annuaire d'une entité, ce service construit une structure de données du type "PEM Forward Certification Path" (certificat d'utilisateur et une instance de l'arbre de certification de cet utilisateur jusqu'à la CA IPRA) relative à cet utilisateur.

- Vérification d'un PEM Forward Certification Path : ayant comme entrée une structure de données du type "PEM Forward Certification Path" relative à l'entité A et le nom d'annuaire du récepteur B de cette donnée, ce service obtient la chaîne de certificats B → A vérifiée, ou retourne une indication d'erreur.

- Récupération de Clés Publiques : ayant comme entrée le nom d'annuaire d'une entité, ce service récupère les certificats valides appartenant à cet entité.

- Récupération de Listes de Révocation : ayant comme entrée une liste de noms de CA, ce service récupère les listes de révocation appartenant à chaque CA.

Notre choix initial a été d'utiliser l'annuaire X.500 comme service de noms où les données publiques d'authentification sont conservées et d'où ils sont récupérées par le DAS. Ce choix est justifié par les raisons suivantes :

- le besoin d'une base initiale pour la validation du modèle X.509 et de toute l'architecture,
- le besoin d'intégration et de compatibilité avec le projet Password.

La fonctionnalité du DAS est offerte par des opérations accédées au moyen, soit d'une interface de programmation (API ou Application Programming Interface), soit d'un protocole du type client/serveur.

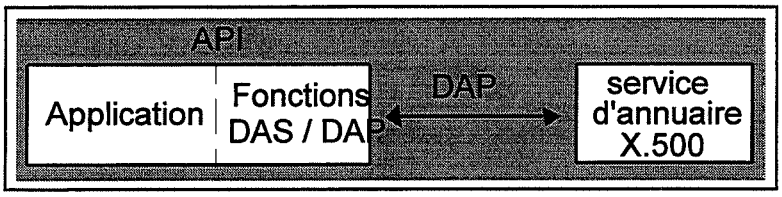

Figure 17. DAS sous forme d'API

Sous cette forme (Figure 17), le DAS consiste en un ensemble de fonctions qui sera intégré à l'application sécurisée.

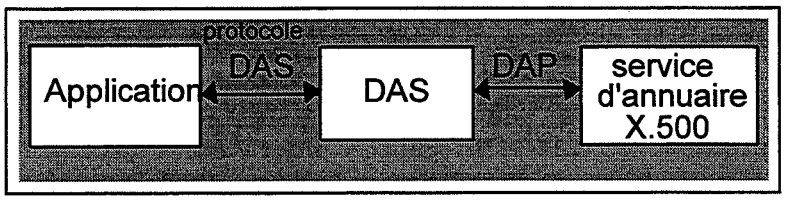

Figure 18. DAS sous forme de Serveur

Sous la forme d'un serveur (Figure 18), le DAS devient totalement indépendant des applications ; l'accès au service se fait par l'intermédiaire d'un protocole d'accès au DAS. La spécification initiale de ce protocole d'accès se trouve dans l'Annexe B.

Les opérations du DAS récupèrent une série de certificats, les listes de révocation concernées, vérifient les signatures de ces certificats et que la séquence constitue une chaîne valide de points de confiance. A partir de ces services, chaque application sécurisée peut entamer ses propres procédures d'authentification et bien d'autres services de sécurité (l'authentification étant à la base de tous les services de sécurité).

Les algorithmes pour la construction et la vérification de chemins de certification ont été conçus en conformité avec les modèles de certification Password et PEM. La description complète de ces algorithmes se trouve dans [Men93].

8.4.4. Clés d'Amorçage

L'assurance des opérations du DAS ne doit pas dépendre de la sécurité du service de noms. En particulier, le DAS ne doit pas faire confiance à l'annuaire en ce qui concerne l'intégrité et l'authenticité des informations qui y sont conservées.

Bien que les informations manipulées par le DAS soient elles-mêmes protégées par leur signature, certaines d'entre elles ne peuvent pas être vérifiées, à savoir, les certificats des CA de plus haut niveau d'une hiérarchie de certification. Pour éviter qu'une CA pirate puisse détourner des procédures d'authentification, le DAS maintient des informations relatives aux clés publiques des CA de plus haut niveau de la hiérarchie de certification d'une organisation ou domaine.

localRootCA ::= SET OF Certificate

Une organisation peut également accorder sa confiance à d'autres CA de haut niveau, indirectement liées à l'organisation, mais permettant l'authentification d'entités d'un autre domaine à travers la certification croisée. Le DAS offre donc la possibilité d'interfonctionnement fiable entre domaines de sécurité à travers la spécification des noms de ces autres CA de haut niveau :

trustedRootCA ::= SET OF DistinguishedName

Puisque les clés publiques des CA racines (*localRootCA*) sont conservées par le DAS, les certificats des autres CA *trustedRootCA* ne sont pas nécessaires car le DAS peut les vérifier. Il est possible de spécifier plusieurs paires (*localRootCA*, *trustedRootCA*).

CHAPITRE 9
LA SECURISATION DES APPLICATIONS

Après les étapes de conception et de développement de l'architecture, nous avons validé ses divers composants au sein du projet Password grâce aux tests avec les autres partenaires qui disposaient de différentes implémentations. Ayant accompli ces étapes, nous avons procédé à la sécurisation des applications. Nous avons intégré les éléments de notre architecture dans des implémentations de deux applications :

* l'annuaire réparti UCOMX.500, une évolution de PIZARRO développé à l'INRIA dans le cadre du projet européen THORN, en conformité avec les Recommandations X.500 1988 [E3X92a].
* la messagerie électronique UCOMX.400, développée en conformité avec les Recommandations X.400 1988 [E3X92b].

La modularité de l'architecture de sécurité SecUcomx a facilité considérablement le développement des applications sécurisées, ce que nous décrirons dans la suite de ce chapitre.

9.1. Un Service d'Annuaire Sécurisé

Les objectifs de l'intégration de mécanismes de sécurité dans l'annuaire réparti X.500 sont les suivants :

(1) protéger l'accès et l'interaction entre les serveurs d'annuaire (via DAP et DSP, respectivement),

(2) protéger les ressources et la base d'information de l'annuaire (DIB).

Au premier abord, notre travail s'est concentré sur les aspects considérés dans le projet Password, à savoir, la mise en place des services et mécanismes de sécurité normalisés. Le modèle de la sécurité de l'annuaire, défini par les normes X.500, concerne seulement le premier aspect cité ci-dessus.

En contrepartie, le deuxième aspect, i.e., la protection de l'accès local aux différentes parties de la DIB, l'intégrité et l'éventuel besoin de confidentialité, n'est pas pris en compte par les recommandations. Ceux-ci sont considérés comme des points ouverts de la sécurité de l'annuaire, mais comptent autant que l'aspect (1) si l'on souhaite avoir un service d'annuaire réellement sécurisé. Dans la Section 10.2 nous aborderons ces problèmes et proposerons des solutions.

9.1.1. Architecture et Services

Nous avons développé une version sécurisée de l'implémentation UCOMX.500 [E3X92a]. Le système résultant est composé des modules suivants :

- le serveur d'annuaire (DSA),
- l'agent utilisateur d'annuaire (DUA) permettant aux utilisateurs d'acceder aux services d'annuaire,
- le serveur de données pour l'authentification (DAS) offrant les services d'établissement et de vérification de chemins de certification,
- le gestionnaire de secrets (GS) permettant à chaque utilisateur d'acceder à sa clé privée.

La Figure 19 illustre le rapport entre les éléments de l'architecture et ceux de l'annuaire X.500.

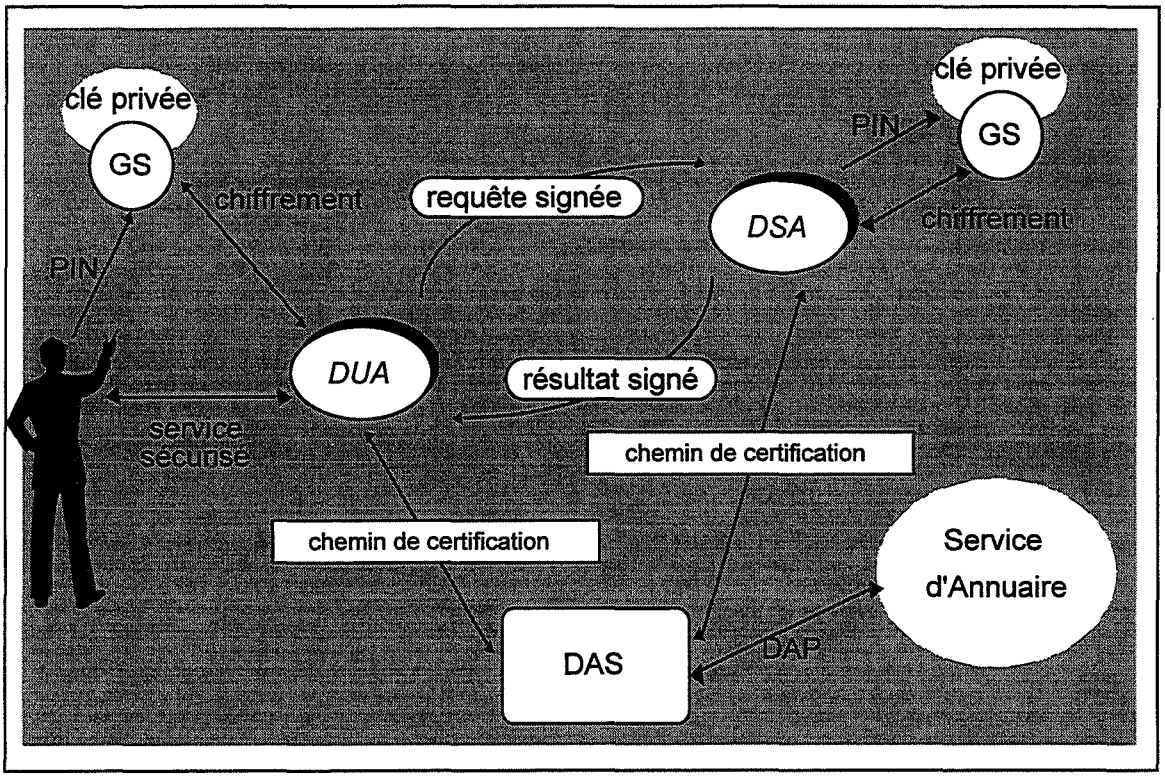

Figure 19. Rapport entres les éléments de l'architecture et l'annuaire X.500

Puisque cette implémentation disposait déjà d'un schéma de contrôle d'accès [Hui91b], nous avons pu utiliser ce schéma comme description des politiques d'autorisation sur lesquelles les méthodes d'authentification s'appliquent.

9.1.1.1. L'agent utilisateur d'annuaire

L'agent utilisateur d'annuaire (DUA) se compose de deux modules :

- une interface homme-machine (IHM), qui est une interface graphique permettant à un utilisateur de formuler ses requêtes,
- une API DAP permettant l'échange des opérations avec les serveurs d'annuaire via le protocole DAP.

Lorsque l'utilisateur utilise l'UA sécurisé, il lance son GS qui lui permettra d'accéder à sa clé privée pour la génération de signatures numériques. La boîte à outils de sécurité est intégrée dans l'API DAP pour permettre la construction de requêtes signées et l'authentification de résultats. En particulier, l'API DAS est intégrée dans la fonctionnalité du DUA pour qu'il puisse construire et valider des chemins de certification. Le DUA sécurisé est capable de réaliser les services de sécurité suivants :

- l'authentification d'un DSA durant l'établissement de l'association,
- la construction de requêtes d'opérations signées,
- l'authentification des résultats d'opérations,
- la détection de re-jeu de résultats,
- la détection de la manipulation de résultats.

9.1.1.2. Le serveur d'annuaire

Puisque le DSA participe activement des procédures de sécurité, il possède sa propre paire de clés et son certificat. Un GS est donc associé au DSA de manière à lui donner accès à sa clé privée. La boîte à outils de sécurité est intégrée dans la fonctionnalité du DSA pour lui permettre d'authentifier les requêtes d'opérations et de construire des résultats sécurisés. Le DSA sécurisé est capable de communiquer avec le DAS via son protocole d'accès afin d'obtenir et de valider des chemins de certification lors des procédures d'authentification. Le DSA sécurisé est capable de réaliser les services de sécurité suivants :

- l'authentification d'un DUA durant l'établissement de l'association,
- l'authentification des requêtes d'opérations,
- la construction de résultats d'opérations signées,
- la détection de re-jeu de requêtes,
- la détection de la manipulation de requêtes.

9.1.2. Authentification durant l'établissement de l'association

En ce qui concerne les opérations d'association de l'annuaire, nous avons ajouté les méthodes d'authentification simple protégée et d'authentification forte. L'authentification simple protégée à été implémentée pour que les mots de passe des utilisateurs et des DSA ne transitent pas en clair sur le réseau. Du côté du DUA, l'utilisateur choisi la méthode d'authentification qu'il désire utiliser pour s'associer à un DSA quelconque. Les procédures d'authentification de l'annuaire ont été décrites dans 6.2.1. Par la suite nous analysons la réalisation des deux methodes d'authentification.

9.1.2.1. L'authentification simple protégée

L'application effective des méthodes d'authentification simple se heurte à des problèmes tant au niveau opérationnel qu'au niveau de l'assurance des mécanismes d'authentification eux-mêmes. Nous pouvons dépeindre deux scenarios :

cas (1) : le DSA avec lequel l'usager établit l'association, gère l'entrée de ce dernier.

Dans un tel cas, la mise en vigueur des procédures d'authentification simple est plus facilement abordable car le DSA retient la copie du mot de passe de l'usager (normalement dans l'attribut "*password*" de son entrée).

Si la politique de sécurité établit que la méthode d'authentification simple *non-protégée* doit être employée, le DSA est capable de vérifier si le mot de passe fourni par l'utilisateur correspond à la valeur de sa copie locale.

De même, si la méthode d'authentification simple *protégée* doit être employée, le DSA sera capable de reproduire la valeur protégée du mot de passe fournie par l'usager, en utilisant sa copie locale de ce mot de passe. Cela permet le DSA d'authentifier cet usager.

cas (2) : le DSA ne gère pas l'entrée de l'utilisateur.

Dans un tel cas, le seul moyen de vérifier la valeur du mot de passe de cet utilisateur serait de déclencher une opération de comparaison vers le DSA qui detient l'entrée de cet utilisateur. Dans le cas de l'authentification simple non protégée, cette solution serait encore possible, malgré l'impact sur le niveau de performance et de sécurité car le mot de passe de l'utilisateur continuerait à passer en clair par le réseau, et probablement par plusieurs DSA.

Dans le cas d'authentification simple protégée, cette solution ne serait pas réalisable car l'opération de comparaison, telle qu'elle est définie actuellement, ne permet que la spécification de types et de valeurs d'attributs (*attributeValueAssertion*) pour la comparaison. Puisque ce type accréditations ne constitue pas un attribut de l'annuaire, le DSA ne disposant pas d'une copie du mot de passe de l'utilisateur n'est pas capable de l'authentifier.

Par conséquent, cette méthode d'authentification ne peut être utilisée qu'avec les DSA ayant accès à l'entrée de l'usager. En outre, le format des accréditations ne permet pas d'indiquer l'algorithme utilisé pour le scellement. Pour régler ce problème, nous avons fixé l'utilisation de l'algorithme MD5 [Riv92]. Cette anomalie a été corrigée dans la version 1994 des spécifications des protocoles d'annuaire.

9.1.2.2. L'authentification forte

Lorsque l'authentification forte est demandée, le DUA utilise l'API DAS pour établir le chemin de certification vers le DSA avec lequel l'utilisateur souhaite s'associer. Puisqu'un DUA intègre déjà le protocole DAP, il se sert seulement des algorithmes du DAS.

Une fois que le chemin de certification a été établi, le DUA rempli le jeton d'authentification et utilise la fonction de génération de signature. Puisque le DUA opère dans l'environnement de l'utilisateur, la fonction de génération de signature peut se communiquer avec le GS, qui est censé avoir être activé préalablement par l'utilisateur. A la réception d'une demande de chiffrement, le GS la présentera à l'utilisateur et n'effectuera le chiffrement que sur confirmation de ce dernier. Après confirmation de l'utilisateur, le GS donne suite au chiffrement pour que le DUA puisse envoyer sa demande d'association au DSA.

Lorsque le DSA reçoit un jeton d'association signé, il procédera tout d'abord à la vérification de la clé publique de l'utilisateur. Dans ce contexte, l'approche choisie est de permettre un maximum de flexibilité en ce qui concerne l'information fournie par l'utilisateur. Ainsi, le DSA permet que l'utilisateur fournisse :

- seulement son certificat,
- ou le chemin menant jusqu'au point commun de confiance entre le DSA et l'utilisateur,
- ou le chemin complet entre le DSA et l'utilisateur.

Pour valider la clé publique de l'utilisateur, le DSA essaiera d'abord de le faire avec les informations contenues dans la partie de la DIB qu'il gère. Dans les cas où le DSA ne detient pas toute l'information nécessaire pour le faire, il contacte le DAS en lui envoyant une requête d'établissement ou de vérification de chemin de certification selon l'information fournie par l'utilisateur.

Dans les cas où un chemin de certification valide n'a pas pu être trouvé, le processus d'authentification est arrêté et l'utilisateur est notifié du problème en question. Dans le cas contraire, i.e., la clé publique de l'utilisateur a pu être validée, le DSA l'utilise pour vérifier la signature du jeton. Finalement, le DSA vérifie si ce jeton ne fait pas objet d'un re-jeu. Le processus d'authentification est alors achevé.

9.1.3. Authentification des Opérations

En ce qui concerne la sécurisation des opérations, l'annuaire ne définit que la méthode d'authentification forte. Pour les requêtes, l'annuaire considère la spécification du nom d'annuaire de l'expéditeur comme une méthode d'authentification. Nous ne considérons pas cette mesure comme valable car un intrus pourrait s'identifier en utilisant le nom d'annuaire de n'importe quel utilisateur légitime.

Pour garantir un ensemble complet de services, nous avons développé le mécanisme d'authentification forte pour toutes les opérations d'annuaire, i.e., les opérations de lecture, recherche et de modification. L'interface homme-machine du DUA permet à l'utilisateur de sélectionner des services de sécurité pour chaque requête à l'annuaire de façon indépendante.

Lorsque l'authentification forte est demandée, le DUA rempli les paramètres de sécurité[23] en spécifiant le nom du DSA vers lequel la requête s'adresse, une estampille de temps spécifiant la durée de validité de la requête et un nombre aléatoire. Cet ensemble d'informations protège le destinataire d'une attaque par re-jeu. Ensuite le DUA appelle la fonction de génération de signature, qui envoie celle-ci au GS pour le chiffrement. Après confirmation de l'utilisateur, le GS complète le chiffrement et le DUA peut alors envoyer sa requête signée au DSA.

Lorsque le DSA reçoit une requête signée, il vérifie les paramètres de sécurité pour s'assurer qu'il correspond au serveur vers lequel la requête s'adresse, et que celle-ci ne fait pas l'objet d'un re-jeu. Ensuite, il valide la clé publique de l'utilisateur. Si la validation réussit, le DSA utilise cette clé pour vérifier la signature de la requête. Cela garantit le DSA de l'origine de la requête et que celle-ci n'a pas été manipulée pendant le transfert.

A ce stade le DSA peut consulter sa politique d'autorisation (par exemple, sa liste de contrôle d'accès) pour vérifier si l'utilisateur à les droits d'accéder au service et aux informations de demandés.

En ce qui concerne la procédure de validation de clés publiques, il reste le même que celui décrit plus haut pour l'authentification de l'association. Bien entendu, une même procédure ne sera pas répétée dans une même session. Une fois le chemin de certification établi et la clé publique d'un partenaire (DUA ou DSA) validée, pour les requêtes subséquentes, le DUA ou DSA vérifiera seulement que le certificat du partenaire en question est toujours valide et qu'il n'a pas été révoqué.

En ce qui concerne les politiques de sécurité, notre choix a été de permettre une certaine flexibilité, i.e., de permettre au DSA d'opérer sous une politique de sécurité quelconque définie par son domaine ou son administrateur. Ainsi, de façon générale,

* la possibilité d'emploi des procédures d'authentification forte pour les opérations est indépendante de la méthode utilisée l'authentification de l'établissement de l'association,
* le niveau de sécurité requis pour chacune des opérations dépend de la politique de sécurité adoptée, ainsi, les opérations sécurisées sont indépendantes les unes des autres,
* l'authentification des résultats peut être demandée indépendamment de la méthode utilisée pour authentifier la requête correspondante.

Bien sûr, ces mesures peuvent être modifiées de façon plus restrictive selon la politique d'autorisation sous laquelle le DSA doit fonctionner.

9.2. Un Service de Messagerie Sécurisé

Durant le projet Password, nous avons intégré les services de sécurité de la classe S0a dans l'implémentation UCOMX.400. Les détails de cette implémentation peuvent être trouvés dans

---

[23]Les paramètres de sécurité des protocoles d'annuaire ont été décrits dans la Partie II, Section 6.2.1de ce travail.

[E3X92b]. Dans cette section nous décrivons l'intégration des composants de l'architecture SecUcomx dans l'implémentation de la messagerie sécurisée.

9.2.1. Architecture et Services

L'application résultante est composée de modules qui collaborent pour offrir les services de sécurité de bout-en-bout correspondants à la classe S0a. Ces modules sont les suivants :

- l'agent de transfert de messages X.400 (MTA),
- l'agent utilisateur (UA), permettant aux utilisateurs d'accéder aux services de messagerie,
- le serveur de données pour l'authentification (DAS) offrant les services d'établissement et de vérification de chemins de certification,
- le gestionnaire de secrets (GS) permettant à chaque utilisateur d'acceder à sa clé privée.

Les services de la classe S0a n'exigent pas la participation active du MTS, pour la réalisation de services sécurisés. Ainsi, nous nous sommes concentrés sur les fonctionnalités d'un UA sécurisé et donc capable de fournir les services de la classe S0a par l'intermédiaire du protocole P3. L'UA lui-même est composé de deux modules :

- une interface homme-machine (IHM), qui est une interface graphique permettant à un utilisateur de composer ses messages,
- le Client P3, un module chargé de réaliser le protocole d'accès au MTS, i.e., le protocole P3

L'UA sécurisé offre une interface utilisateur à fonctionnalités réduites pour un service de messagerie sécurisé. Le Client P3 est chargé d'expedier et de réceptionner les messages pour un utilisateur donné. Il joue un rôle d'intermediaire entre l'IHM et le MTA. Tournant en tâche de fond sur la machine de l'utilisateur, le Client P3 peut recevoir les messages d'un utilisateur même lorsque celui-ci n'a pas lancé son UA. Notre implémentation est capable d'offrir les services de sécurité suivants :

- l'authentification de l'origine de messages,
- la détection du re-jeu de messages,
- la détection de la manipulation de messages,
- la confidentialité de messages.

Puisque les services de sécurité de la classe S0a sont fournis au moyen des éléments du protocole P3, nous avons choisi d'implementer ces services dans le module Client P3. Cette tâche a nécessité l'intégration de l'infrastructure cryptographique pour la génération et la vérification des *authentications checks* (à savoir les types MOA et CIC) ainsi que les jetons de messages définis par le modèle de sécurité de X.400. De plus, nous avons fourni au Client P3, l'accès au service du DAS à travers l'intégration de le son protocole d'accès.

9.2.2. Fonctionnement de l'Impléméntation

Lorsque l'utilisateur souhaite envoyer des messages sécurisés, il commence par s'assurer que le GS est lancé dans son environnement. Ensuite, il active son UA qui, dans sa phase d'initialisation, ira contacter le DAS de manière à obtenir les certificats nécessaires à la validation de sa clé publique. En fait, le DAS lui retourne un chemin de certification dans le format conforme au modèle de sécurité de X.400, i.e., une structure du type *Certificates* contenant le certificat de l'utilisateur et les certificats de toutes les CA supérieures jusqu'à la racine de son arbre de certification (voir 8.4.1).

A ce stade, l'utilisateur est capable d'envoyer des messages sécurisés comportant n'importe quel service de la classe S0 et d'inclure dans ses messages la structure *Certificates* retournée par le DAS. Au moment de la production de signatures, la fonction *gs-client* intégrée dans l'UA communiquera avec le GS qui, à son tour, demanderá une confirmation à l'utilisateur selon le mode d'opération choisi. Le rapport entre les élements de l'architecture SecUcomx et la messagerie X.400 est illustré par la Figure 20.

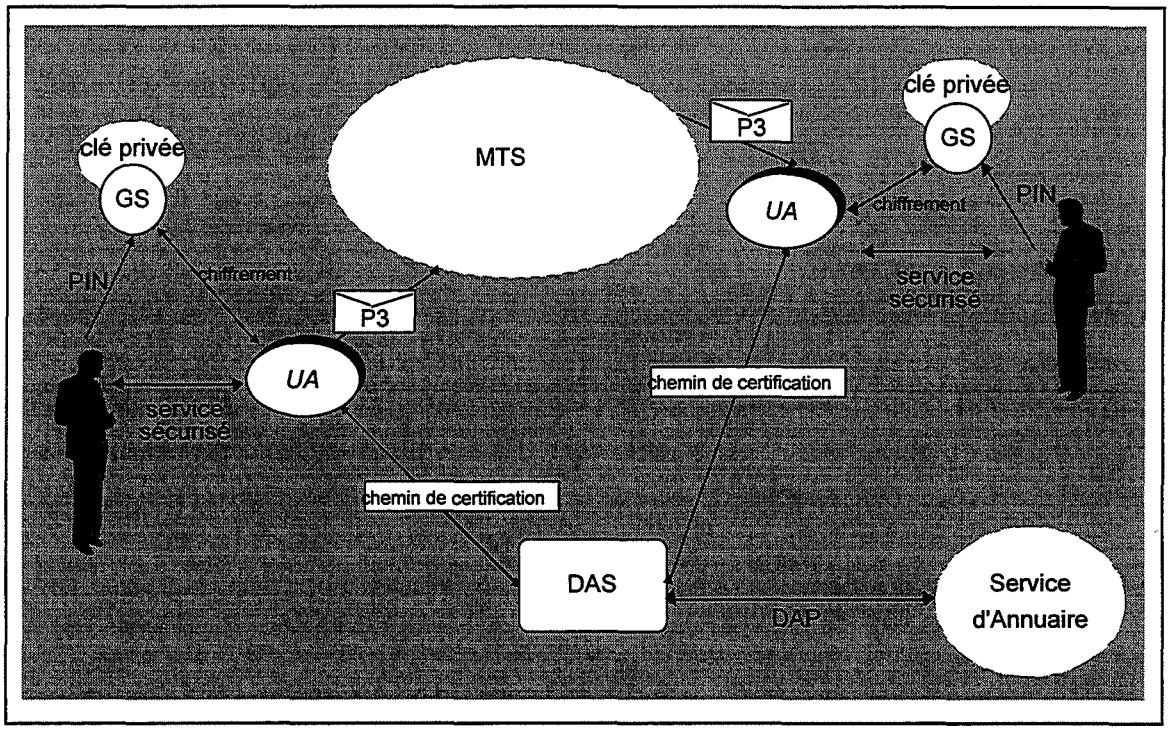

Figure 20. Rapport entre les éléments de l'architecture et la messagerie X.400

Pour avoir le service de confidentialité de messages, l'expéditeur doit disposer du certificat de chaque destinataire de manière à extraire la clé publique nécessaire au chiffrement de la clé symétrique utilisée pour chiffrer le message. En pratique, il ne suffit pas de récupérer seulement le certificat de chaque destinataire, car si un intrus réussi à tromper l'expéditeur en lui fournissant un faux certificat au lieu d'un certificat valide d'un destinataire quelconque du message, alors la confidentialité du message est violée. Il en est de même, si l'expéditeur utilise une clé publique dont le certificat à été révoqué à cause de la compromission de la clé privée correspondante.

Pour éviter ce danger, lorsque l'utilisateur souhaite envoyer des messages confidentiels, l'UA lance, pour chaque destinataire, des requêtes au DAS en demandant le chemin de certification

menant de la CA racine de l'arbre de certification de l'expéditeur jusqu'au destinataire. Cela permet à l'utilisateur d'avoir confiance en la procédure de confidentialité.

A la reception d'un message, le Client P3 verifie si celui-ci contient des elements de sécurité. En cas affirmatif, celui-ci contactera le DAS afin d'obtenir et verifier le certificat de l'expediteur du message. Avec la clé publique extraite de ce certificat, l'UA est capable de verifier les signatures (MOA et CIC) associés au message.

L'utilisateur a la possibilité de "refaire" les vérifications sur de messages reçus précédemment. Cette fonctionnalité est utile notamment lorsque le certificat de l'expediteur n'a pas pu être vérifié à cause d'une manque de disponibilité des services d'annuaire. Lorsque les opérations de sécurité sont rejouées par l'UA, celui-ci se connecte au DAS pour vérifier le certificat, les vérifications proprement dites étant réalisées par l'UA.

Lorsqu'un message confidentiel est reçu, l'UA de l'utilisateur communique avec le GS afin d'accéder à sa la clé privée nécessaire à la découverte de la clé de déchiffrement utilisée pour chiffrer le message. A la réception de la demande de chiffrement, le GS demandera une confirmation de l'utilisateur, selon le mode d'opération choisi. Après cette opération, l'utilisateur est donc capable de déchiffrer le contenu du message reçu.

Notre implémentation offre ainsi des services de base pouvant être utilisés pour sécuriser l'échange de messages interpersonnels ou d'autres applications de messagerie. Toutefois, comme nous l'avons déjà mentionné, le modèle de sécurité de X.400 présente un nombre de problèmes ouverts et d'aspects négligés. Dans le chapitre suivant, nous aborderons ceux qui nous avons pu identifier par l'expérimentation et pour lesquels nous proposerons des solutions.

CHAPITRE 10
DES PROBLEMES OUVERTS

Le développement de l'architecture SecUcomx et son utilisation pour la sécurisation de l'annuaire X.500 et de la messagerie X.400 nous a confronté à des problèmes négligés non seulement au niveau du cadre d'authentification X.509 mais aussi au niveau des modèles de sécurité de ces applications. Dans ce chapitre nous allons exposer ces problèmes ouverts en essayant d'apporter des solutions viables.

Ce chapitre se décompose donc en quatre sections. La première questionne le cadre X.509 en tant que base pour l'implémentation d'une infrastructure commune pour les services de sécurité pour les systèmes ouverts. La deuxième section expose des lacunes de la sécurité de l'annuaire X.500. La troisième section explore des problèmes liés à la sécurité de la messagerie électronique. La dernière section de ce chapitre apporte des remèdes à un aspect négligé dans Password en ce qui concerne l'utilisation du ACSE pour la sécurisation d'applications ne disposant pas d'extensions de sécurité normalisées.

En particulier, les problèmes abordés dans la première section de ce chapitre constituent les motivations pour la suite de notre travail, à savoir l'évolution du concept du DAS, où nous chercherons à proposer des solutions permettant la mise en place d'une infrastructure globale pour les services d'authentification dans les systèmes ouverts. Nous consacrerons toute la Partie IV à l'étude de ce sujet.

10.1. Le Cadre X.509 comme base de l'Infrastructure de Sécurité

Le cadre X.509 considère l'annuaire X.500 comme une partie de l'infrastructure de sécurité dans la mesure où il peut conserver des données publiques pertinentes aux procédures d'authentification, telles que les certificats et les listes de révocation.

La question qui se pose aujourd'hui est de déterminer si l'annuaire X.500 et le cadre d'authentification X.509 peuvent être adoptés comme les pivots d'une infrastructure globale pour l'implémentation des services de sécurité dans les systèmes ouverts.

10.1.1. A propos du Schéma de Nommage

La question du nommage est très importante pour l'authentification dans la mesure où les certificats associent un nom à une clé publique. Nous avons déjà soulevé le besoin de noms uniques dans la Section 8.3.3. Outre cela, il est essentiel que les noms utilisés dans des certificats soient non ambiguës et descriptifs. Ceci pour les raisons suivantes :

- permettre aux utilisateurs d'établir facilement le rapport entre le nom contenu dans le certificat et l'information d'identification utilisée dans la vie réelle,
- minimiser la probabilité de conflit entre deux noms similaires. Par exemple, fréquemment les utilisateurs se rendent compte de l'envoi d'un message, par erreur, à un certain destinataire à cause d'une confusion entre deux adresses similaires.

Deux autres qualités souhaitables pour le schéma de nommage utilisé pour la certification sont de :

- fournir une forme universelle d'identification,
- être indépendant d'un ou d'un ensemble particulier de services.

En prenant comme exemple la messagerie électronique, un utilisateur peut avoir plusieurs boîtes aux lettres (par exemple, mendes@emse.fr, mendes@osi.e3x.fr) et souhaiter disposer d'un service sécurisé en utilisant une même identité. En contrepartie, un utilisateur peut avoir une seule boite aux lettres et souhaiter échanger des messages en utilisant des différentes formes d'identification, par exemple, comme individu, citoyen ou bien en tant qu'employé d'une organisation.

De plus, il est nécessaire d'avoir une infrastructure capable de supporter la désignation de noms et la récupération de certificats basée sur les noms utilisés dans ces certificats ou bien à partir d'autres attributs de recherche. De plus, un système de certification devant accommoder un grand nombre d'utilisateurs et admettre une gestion distribuée doit avoir un schéma de nommage en rapport avec ces mêmes caractéristiques.

Le format des certificats X.509 exige l'utilisation des noms d'annuaire. Cependant, à l'heure actuelle les services d'annuaire ne sont pas largement déployés et ne représentent pas la majorité des utilisateurs. En particulier, l'utilisation des noms et de la structure **DNS** (Internet **D**omain **N**ame **S**ystem) [Moc87a, Moc87b] a été suggéré pour la mise en place du système de certification de l'Internet car ce système offre une infrastructure déjà établie et partage certaines caractéristiques avec les noms X.500.

Par la suite nous étudions ces deux systèmes en tant qu'approches de nommage pour la certification.

10.1.1.1. Les Noms DNS

Le système DNS [Moc87a] fournit une infrastructure pour le nommage de machines et pour la correspondance (ou *mapping*) entre des noms de machines et leur adresse Internet. Les bases de données DNS ne contiennent pas d'information sur des utilisateurs, mais, à l'heure actuelle, des millions d'utilisateurs s'échangent des messages en utilisant des adresses dérivées de noms DNS. Ainsi, le DNS représente une infrastructure considérable pour la désignation et la récupération de noms de machines.

L'approche DNS emploie des domaines de haut niveau représentant des pays ou des grandes communautés d'utilisateurs. Les pays sont représentés par des codes sur deux caractères extraits de la norme ISO 3166, le même standard adopté par X.500 pour l'identification des pays. Les autres domaines de haut niveau (actuellement au nombre de sept) sont identifiés par des codes sur trois caractères, par exemple, COM (commercial), EDU (éducationnel), ORG (organisationnel), NET (networks), MIL (militaire), GOV (gouvernements civils) et INT (organisations internationales).

Probablement, le contretemps plus important à l'utilisation de noms DNS pour la certification repose sur le fait que les noms DNS sont très concis (chaque composant a typiquement entre trois et six caractères). Cela est voulu pour éviter la création de noms longs et donc difficiles à saisir. Cependant, l'utilisation de noms courts augmente la possibilité de collision et de confusion. Comme nous l'avons déjà mentionné, les noms peu descriptifs et qui ressemblent à d'autres noms peuvent plus facilement confondre les utilisateurs.

Cette possibilité de conflit constitue aussi un problème potentiel au niveau des domaines. Par exemple, si toutes les organisations commerciales s'enregistraient sous "COM", on pourrait s'attendre à de nombreuses collisions, étant donné la tendance aux noms concis. Ainsi, bien que la structure du DNS se soit montrée suffisante face à l'expansion de l'Internet ces dernières années, la question se pose de savoir si cette structure restera suffisante pour la continuation de cette expansion, sans que la structure et les conventions de nommage soient révisées.

De toutes façons, bien que les noms DNS soient utilisés pour dériver des noms de boîtes aux lettres d'utilisateurs, ceux-ci ne correspondent pas à des noms d'utilisateurs. Par conséquent, le principal avantage de l'utilisation de noms DNS dans les certificats est l'existence d'une infrastructure distribuée pour la désignation de ces noms. Tels qu'ils sont sélectionnés actuellement, les noms DNS ne sont pas les candidats idéaux pour l'utilisation dans des certificats.

10.1.1.2. Les Noms d'Annuaire

Les structures des noms d'annuaire ou DN (**D**istinguished **N**ame) et celle des noms DNS partagent des caractéristiques. Dans ces deux structures les noms sont formés d'une séquence de composants dont chacun représente un noeud d'un arbre de nommage. Chaque noeud constitue un point à partir duquel le nommage peut être délégué à une autorité de désignation (qui peut utiliser

pour cela une approche géopolitique ou organisationnelle). Cela implique que chaque autorité de désignation attribuera des noms subordonnés à son point de l'arbre, sans risque de collision avec d'autres branches de l'arbre. Une telle approche permet ainsi la définition d'un schéma de nommage pouvant être géré de façon distribuée.

La différence entre ces deux approches de nommage est que les composants des noms X.500, appelés RDN (**R**elative **D**istinguished **N**ame), ne sont pas constitués de simples chaînes de caractères "*caseless*". Chaque RDN est une paire (attribut, valeur) permettant l'association explicite de sémantiques à chaque composant. Des exemples de types d'attributs définis actuellement comprennent : pays (C), état ou province (S), localité (L), organisation (O), unité organisationnelle (OU), nom courant (CN).

X.500 n'impose pas de restrictions à la structure des DN, mais permet un schéma de nommage avec plusieurs racines où des pays et des organisations forment l'équivalent des domaines de haut niveau du DNS.

Les DN peuvent ainsi être fortement descriptifs, en comparaison avec les noms DNS. La possibilité d'utiliser différents jeux de caractère et d'associer des attributs à chaque RDN rend le DN un bon choix pour utilisation dans des certificats. L'incovenient de cette approche est que les DN sont typiquement longs et donc peu commodes à saisir à travers un clavier, notamment pour des utilisateurs non habitués à l'approche OSI.

En outre, le DN est indépendant d'un service ou d'un système d'exploitation particulier. En contrepartie, cette indépendance peut impliquer la difficulté de convertir un nom d'annuaire vers une autre forme d'identification (un nom DNS, par exemple), et vice-versa.

Par conséquent, d'une part, les noms d'annuaire sont, à certains égards, des candidats idéaux pour utilisation dans des certificats, d'autre part il y a des obstacles à leur adoption, parmi lesquels :

- le manque d'une infrastructure pour la désignation de noms,
- le manque d'un déploiement répandu des services d'annuaire,
- la résistance de communautés non OSI, qui ne sont pas familiarisées avec la structure des noms d'annuaire.

Cependant, les problèmes soulevés ci-dessus peuvent être justifiés par l'état récent de la technologie d'annuaire. De plus, l'utilisation de noms d'annuaire et du cadre X.509 comme approche commune n'implique pas la dépendance sur les systèmes d'annuaires pour le stockage et la récupération de certificats. Il est possible de stocker des certificats X.509 dans d'autres bases de données et services de noms.

Etant donné que les DN permettent la création de noms descriptifs et admettent une gestion distribuée pouvant assurer l'unicité et la non ambigüité de noms, les noms d'annuaire peuvent être utilisés pour identifier les partenaires d'une communication et coexister avec les autres formes d'identification spécifiques à chaque service. Il nous faut donc trouver un moyen d'associer, de façon sûre, le DN d'un utilisateur à ses autres identificateurs. Dans la Partie IV (13.3) nous proposons une solution à ce problème.

10.1.2. A propos du Format de Certificats

Le format de certificat défini par la Recommandation X.509 associe une clé publique au nom d'annuaire d'un utilisateur. Néanmoins, ce format de certificat, tel que défini actuellement ne permet pas la réalisation d'une infrastructure commune et globale pour les services d'authentification, comme on le verra par la suite.

10.1.2.1. Un format de certificat pour divers services sécurisés

Comme nous l'avons explicité ci-dessus, bien que le DN puisse constituer une forme descriptive de nommage, il n'offre pas une méthode d'identification globale vis-à-vis de tous les services existants.

Le DN identifie un utilisateur vis-à-vis de l'annuaire mais ne l'identifie pas vis-à-vis d'autres services, parmi lesquels nous pouvons citer le nom de login, l'adresse électronique X.400, la messagerie Internet, etc. A titre d'exemple, dans le cas de CA organisationnelles, où la CA coïncide avec l'employeur d'un groupe de personnes, la CA peut préférer certifier un "numéro de registre" de ces employés plutôt que leur nom d'annuaire.

Il est clair que nous ne disposons pas d'un moyen universel d'identification. Si cela était vrai, chacun de nous n'aurait pas autant de numéros et de codes d'identification. En contrepartie, il ne serait pas souhaitable d'avoir un certificat différent pour chaque sorte de service sécurisé.

Par conséquent, pour qu'un format unique de certificat puisse être adopté, il doit être *extensible* pour pouvoir comporter différents identificateurs d'un utilisateur. De cette sorte, lorsqu'un utilisateur souhaiterait obtenir un certificat, il présenterait à la CA sa clé publique, son nom d'annuaire, ainsi que d'autres identificateurs, peut-être moins descriptifs, mais qu'il doit utiliser pour s'identifier auprès des divers services sécurisés. La CA doit alors employer tous les moyens pour s'assurer de la véracité de ces informations. Le format de certificat PKCS #6, par exemple, permet la réalisation d'une telle approche.

10.1.2.2. Un format de certificat pour divers types d'attributs

Une autre déficience de ce cadre d'authentification est que le même format de certificat est utilisé dans la définition de trois types différents d'attributs comportant différentes sémantiques, à savoir, le certificat d'utilisateur, le certificat de CA et les certificats croisés. Toutefois, le cadre X.509 ne permet pas d'établir, de façon sûre, la différence entre ces types de certificat.

Cela constitue une source de compromission car un attaquant peut se faire certifier par une CA et puis essayer de jouer le rôle d'autorité de certification en produisant de faux certificats sans que cela soit détecté par des moyens cryptographiques. Cela peut entraîner des conséquences graves puisque des utilisateurs peuvent se servir des clés publiques contenues dans de tels certificats pour envoyer de messages confidentiels.

En particulier, en ce qui concerne la certification croisée, son objectif est de rendre plus facile la transition et l'interfonctionnement entre deux domaines ou politiques différentes. Cependant, les sémantiques liées à la certification peuvent varier considérablement entre un domaine de gestion

et l'autre. Ainsi, les procédures d'authentification ne doivent pas omettre l'éventuel interfonctionnement entre deux domaines.

Pour remédier à ces problèmes, il serait préférable que le type d'attribut (certificat d'usager, de CA ou certificat croisé) soit également signé dans le certificat pour permettre l'identification immédiate de la semantique associée au certificat. Nous discuterons d'avantage le manque de précision du cadre X.509 dans 11.1 (Partie IV).

10.1.2.3. Un format de certificat pour divers types de CA

Bien que le cadre X.509 ne spécifie pas un modèle de distribution de CA, le besoin d'une infrastructure de certification, globale à plusieurs communautés différentes, avec des politiques et sémantiques de certification diverses a amené la définition de systèmes de certification comportant différents types de CA, dont chacun peut jouer un rôle différent.

En absence de mécanismes spéciaux permettant la distinction entre les divers types de CA, les applications chargées de la validation des certificats doivent distinguer les CA par leurs propres moyens. A titre d'exemple, dans les modèles Password et PEM, il n'est pas possible de distinguer une PCA d'une CA organisationnelle. Pour pallier au manque de moyens d'identification, en plus des mésures (1) et (2) décrites dans 10.1.2.2 ci-dessus, les applications doivent maintenir une liste protégée de noms de CA associée au type des CA en question. En plus de ne pas être une pratique très propre, cela ne permet pas de distinguer les diverses politiques de certification.

Notre expérience de l'approche X.509 nous a fait constater que la création de modèles de confiance pouvant regrouper des CA, des politiques et des sémantiques de certification hétérogènes exige la définition d'un format commun por la description de telles politiques. A travers un tel format chaque modèle et chaque domaine de sécurité pourrait mettre à disposition des informations permettant d'identifier sans ambiguïté :

* quelles sont les entités chargées de la certification, quels types de certificat celles-ci sont autorisées à émettre et quels types d'utilisateur celles-ci sont autorisées à certifier,
* le niveau de confiance que l'on doit placer à l'identité d'un utilisateur ayant un certificat,
* les sémantiques liées à la certification.

Ce constat nous a amené à rechercher les besoins et la définition d'une méthode permettant de rendre explicite les politiques de certification ainsi que leur rapport avec la validation de certificats. Nous décrivons les résultats de ces études dans la Partie IV.

10.1.2.4. Un format de certificat pour divers types d'utilisateur

La fonction d'un certificat est de garantir l'intégrité et l'authenticité de certains attributs de sécurité. La distribution répandue de certificats rend facile l'obtention de ces attributs, même à un intrus au système. Cela ne devrait donner aucun avantage à cet intrus car l'information contenue dans un certificat est censée être publique.

Toutefois, il peut y exister des attributs de sécurité pour lesquels la confidentialité peut s'avérer nécessaire. Par exemple, la protection de l'intégrité des contrôles de sécurité garantit qu'un usager possède vraiment les droits d'accès qu'il prétend avoir. Cependant, la connaissance des privilèges de tous les usagers donne à un intrus une bonne indication de qui il peut attaquer, ou bien de quelle personne il doit essayer d'intercepter le mot de passe. De tels types d'information ne peuvent pas être distribués en utilisant des certificats du type X.509.

De même, il existe le cas où l'identité réelle d'un utilisateur doit rester confidentielle. Dans ce cas, le certificat doit comprendre des attributs qui n'incluent pas le nom réel de l'utilisateur. De tels certificats sont connus comme du type "*Persona*". L'idée de ces certificats peut paraître paradoxale, puisque la fonction principal d'un certificat est de véhiculer l'identité de l'utilisateur. Toutefois, outre l'identité, d'autres attributs peuvent être utilement inclus dans un certificat. Ceux-ci comprennent :

- l'affiliation organisationnelle,
- l'occupation d'un rôle particulier,
- un identificateur de contrôle d'accès.

En particulier, les identificateurs utilisés pour le contrôle d'accès peuvent être constitués de chaînes aléatoires de bits, sans rapport avec le nom réel de l'utilisateur associé.

En plus de cela, un autre trait souhaitable dans le cas des certificats d'utilisateur est que ceux-ci puissent comporter des informations délimitant le champ de validité des signatures produites au moyen du composant privé correspondant. Par exemple, les certificats d'usager pourraient contenir des informations du genre : "Ce certificat n'autorise pas son porteur à signer quoi que ce soit au nom de la Société W" ou "Cet utilisateur déclare ne pas utiliser ce certificat à des fins personnelles".

10.1.3. Le Format de Listes de Révocation

Une autre déficience du cadre d'authentification X.509 concerne le format des listes de révocation, dont la définition ASN.1 est la suivante :

```
CertificateList ::= SIGNED SEQUENCE {
    signature               AlgorithmIdentifier,
    issuer                  Name,
    lastUpdate              UTCTime,
    revokedCertificates     SIGNED SEQUENCE OF SEQUENCE {
        signature               AlgorithmIdentifier,
        issuer                  Name,
        subject                 CertificateSerialNumber,
        revocationDate          UTCTime } OPTIONAL
}
```

Cette définition est considérée incorrecte pour de nombreux aspects dont les détails peuvent être trouvés dans [Men93]. Les défaillances ont fait l'objet de rapports d'anomalie [Roe92]. Les déficiences plus importantes sont les suivantes :

(1) Ce format ne contient pas d'indication de la prochaine date prévue pour la mise à jour de la liste. Sans cette information, il devient difficile de déterminer si une certaine liste est la plus récente. De plus cela constitue une source de compromission.

Par exemple, un adversaire A qui connaît une clé privée d'un usager B, dont le composant publique a été déjà révoquée, peut essayer de faire valoir cette clé révoquée en remplaçant la liste de révocation courante par une plus ancienne. En procédant ainsi, A aurait accès à des informations confidentielles destinées à B et pourrait se fairer passer pour B dans des procédures d'authentification.

(2) Dans le composant *revokedCertificates*, de type SIGNED SEQUENCE OF SEQUENCE, chaque composant de la dernière séquence spécifie un identificateur d'algorithme (AlgorithmIdentifier), alors qu'en fait une seule signature est appliquée à la double séquence (SEQUENCE OF SEQUENCE). Cela constitue une redondance et entraîne une ambiguïté par rapport à la macro SIGNED dont l'expansion comprend également un identificateur d'algorithme pour la signature.

En contre partie, il serait utile de remplacer la définition SIGNED SEQUENCE OF SEQUENCE par SEQUENCE OF SIGNED SEQUENCE. Cela permettrait que l'information de révocation soit disponible individuellement pour chaque certificat annulé. Une telle fonctionnalité serait particulièrement utile pour des fins de non répudiation de révocation.

Selon la version des nouvelles spécifications dont nous disposons à ce jour [IS9594-8], la recommandation X.509 n'a pas évolué dans ce sens, malgré les nombreux rapports d'anomalie envoyés aux organismes de normalisation.

10.1.4. La Dépendance vis-à-vis de l'Annuaire X.500

Un dernier point qu'il semble utile d'aborder est celui lié à la dépendance vis-à-vis de l'annuaire X.500 comme dépositaire des certificats et des listes de révocation. Pendant le projet Password, nous avons constaté que la fourniture et l'utilisation de services sécurisés peuvent être limitées par le manque de disponibilité du service d'annuaire. Lorsqu'un chemin de certification doit être établi et/ou vérifié, il suffit qu'un DSA, qui conserve l'un des certificats ou l'une des listes de révocation nécessaire, soit indisponible, pour que le processus d'authentification soit interrompu.

Susceptibles d'être très sophistiquées, les opérations de recherche peuvent être coûteuses en temps et en ressources. Si d'une part il est possible de rechercher une entrée d'annuaire utilisant une partie ou la totalité d'un DN, ou même d'autres attributs ; d'autre part, une opération dont les attributs de recherche ne sont pas suffisamment spécifiés peut exiger l'interaction entre de nombreux DSA disséminés à l'échelle mondiale.

Etant donné que d'autres services de noms existent et que la technologie de certification ne dépend pas de celle de l'annuaire X.500, différentes communautés d'utilisateurs peuvent adopter différents services de noms pour stocker leur données publiques d'authentification.

En ce qui concerne le stockage et la récupération des certificats, la comparaison entre DNS et X.500 est impropre. Le système DNS n'a pas été conçu pour offrir des services du type "pages blanches". Cela demanderait que la structure de ce système soit étendue pour inclure des types de

registres permettant de désigner des utilisateurs ainsi que des CA. Ces registres devraient être définis de manière à contenir des certificats et des listes de révocation. Pour obtenir un tel service, une meilleure approche serait d'adapter la structure de pages blanches déjà existante dans l'Internet, tel que le service *Whois++* [GW93]. L'utilisateur déclenche des opérations de recherche dans les bases de données Whois à partir des informations descriptives, examine le résultat de la requête et la re-formule jusqu'à ce que l'entrée de l'utilisateur demandé soit identifiée.

En effet, à l'heure actuelle de nombreux systèmes d'information sont disponibles. Certains d'entre eux sont conçus pour offrir des fonctionalités du type "pages blanches" et d'autres incluent cette fonctionalité. Parmi ces systèmes nous pouvons citer, Netfind [PA94], Gopher [AML93]. En particulier, le système WWW (**W**orld **W**ide **W**eb) offre un service d'information hyper-media, pouvant accèder à plusieurs service de noms, y compris X.500 [PA94]. De même, SOLO (**S**imple **O**bject **LO**ok-up) [HPZW93], actuellement en étude, servira comme un *front-end* pour des services de noms du type pages blanches, notamment l'annuaire X.500.

Du fait de la demande, de plus en plus croissante, des services d'authentification, il est probable que les utilisateurs n'attendront pas que les systèmes d'annuaire deviennent largement disponibles et opérationnels.

10.2. La Sécurité de l'Annuaire X.500

La mise en place des services fiables d'annuaire exige non seulement la protection de l'accès à ce service et de l'interaction entre ses serveurs, mais aussi la protection de ses ressources internes contre l'accès non autorisé. Dans cette section, nous discutons des aspects négligés par le modèle de sécurité de l'annuaire défini par les Recommandations X.500.

10.2.1. Protection des Informations de Connaissance

La DIB est potentiellement répartie entre plusieurs DSA dont chacun en gère un fragment. Une caractéristique impérative de l'annuaire est la transparence de la répartition de cet système, i.e., l'impression que l'ensemble de la DIB est à l'intérieur de chaque DSA. Pour assurer cette caractéristique, il est nécessaire que chaque DSA soit capable d'identifier et d'interagir avec d'autres DSA détenant d'autres fragments du DIT. Chaque DSA conserve donc des informations de connaissance étant, en fait, une description de la portion de l'arbre dont ce DSA est responsable couplée à des références vers d'autres DSA. Cet ensemble d'informations permet aux DSA de naviguer dans l'espace de noms.

L'efficicacité de l'opération répartie de l'annuaire réside non seulement dans la maintenance de la *cohérence* des informations de navigation mais aussi dans leur *intégrité*. Si d'une part la maintenance de la cohérence de ces informations est prévue par les normes, d'autre part les aspects d'intégrité ne sont pas considérés par ces recommandations.

Ce problème a été soulevé par Altarah [Alt92], qui propose que les CA soient chargées de signer chaque référence croisée d'un DSA, Or, une telle solution a un impact sur la performance de l'opération repartie de l'annuaire car le DSA est obligé de vérifier la clé publique de la CA avant de pouvoir vérifier l'intégrité de ses données de navigation.

De plus, cette solution n'est pas adéquate car l'ensemble de références croisées d'un DSA peut être étendu et mis à jour dynamiquement en utilisant les résultats de l'opération distribuée. Cette solution demanderait que chaque référence modifiée ou ajoutée soit soumise à la CA pour la certification. Cela implique un effort additionnel de maintenance, d'autant plus que les CA sont normalement des entités "*off-line*".

En fait, ce problème peut être résolu de façon plus simple et efficace si le DSA signe lui-même ses informations de navigation au moyen de sa propre clé privée. Le DSA utiliserait alors sa clé publique pour vérifier l'intégrité de ces données et sera capable de s'apercevoir d'une modification quelconque.

10.2.2. La Confidentialité et la Protection de la DIB

L'utilisation des procédures d'authentification au niveau du DAP et du DSP permet l'implémentation de mécanismes de contrôle d'accès à des informations sensibles conservées dans l'annuaire. Si d'une part les spécifications du service d'annuaire ont évolué dans le sens de la spécification d'un moyen propre pour la définition des politiques d'autorisation à travers un

schéma de contrôle d'accès [X.50093], d'autre part aucune mesure de protection n'a été définie pour protéger ces informations durant leur transit sur le réseau.

Or, il ne semble pas raisonnable de mettre en place des mécanismes sophistiqués de contrôle d'accès aux informations de l'annuaire si par la suite ces informations sont transmises en clair par le réseau. Cela rend inutile le mécanisme de contrôle d'accès car un intrus peut surveiller la communication et obtenir des informations qui, par principe, devraient être d'accès restreint. Cet aspect n'est même pas mentionné par les nouvelles spécifications de l'annuaire.

Comme réponse à ce problème, nous proposons qu'un mécanisme de confidentialité soit couplé au mécanisme de contrôle d'accès. Dans ce contexte, deux types différents de solution peuvent être envisagés :

(1) *durant établissement de l'association* : l'utilisateur et le DSA établissent une clé de session symétrique qui servira, tout au long de l'association, à chiffrer des données sujettes au contrôle d'accès. Pour l'établissement de la clé de session, l'un des mécanismes décrits dans la Section 7.3.3.1 peut être utilisé.

(2) *durant l'exécution de l'opération* : A la réception d'une première requête concernant des informations sous contrôle d'accès, et après avoir réalisé les procédures d'authentification et d'autorisation, le DSA génère une clé de session, au moyen de laquelle il chiffre les données qu'il envoie à l'utilisateur avec la clé de chiffrement elle-même chiffrée au moyen de la clé publique de cet utilisateur. Bien entendu, cette clé de session peut être réutilisée pour chiffrer les résultats des requêtes subséquentes du même genre.

En effet, nous ne pouvons pas considérer la solution (1) pour les raisons suivantes :

* la solution implique la modification de la spécification des accréditations d'association de l'annuaire, en plus de la spécification des résultats,
* la solution exige que l'utilisateur puisse déterminer à l'avance si une ou plusieurs de ses requêtes concernent des informations sous contrôle d'accès,
* pour assurer une sécurité effective, cette solution exige que l'utilisateur puisse déterminer à l'avance le DSA qui gère l'information en question pour pouvoir établir la clé de session avec ce serveur.

En contrepartie, la solution (2) ne se heurte à aucun des problèmes mentionnés ci-dessus et permet que ces informations soient obtenues de façon sûre même si celles-ci transitent par plusieurs DSA. Par contre, quelle que soit la solution retenue, elle demanderait la modification de la spécification des résultats de manière à supporter le service de confidentialité de données. La nouvelle spécification, en ASN.1, pourrait être la suivante :

```
NewOpResult ::= CHOICE {
  clearResult                [0]  OperationResult,
  encryptedResult            [1]  SEQUENCE {
        encryptedRes         [0]      ENCRYPTED OperationResult,
        encryptedKey         [1]      ENCRYPTED EncryptingKey OPTIONAL,
        publicKeyId          [2]      SEQUENCE {
                                        serialNumber   INTEGER,
                                        issuer         DistinguishedName
                                      } OPTIONAL
```

```
                    }
}
```

Avec cette définition, chaque résultat peut être retourné en clair (*clearResult*) ou chiffré (*encryptedResult*). Le type *OperationResult* représente la définition actuelle des résultats de l'annuaire [X.511]. Si le résultat doit être chiffré, les composants de *encryptedResult* sont interprétés la façon suivante :

- *encryptedRes* correspond au résultat chiffré,
- *encryptedKey* contient la clé et l'algorithme utilisés pour chiffrer le résultat, eux-mêmes chiffrés au moyen de la clé publique de l'utilisateur ; ces données en clair ayant la définition suivante :

```
EncryptingKey :: = SEQUENCE {
        algorithm   AlgorithmIdentifier,
        key         BIT STRING
}
```

- *publicKeyId* contient des informations permettant à l'utilisateur d'identifier parmi ses clés publiques, celle qui a été utilisée par le DSA pour chiffrer les données *encryptedKey*.

Comme mesure complémentaire, des attributs dont la valeur est sensible, peuvent être conservés chiffrés dans l'annuaire de façon à ce qu'ils soient "invisibles" à des utilisateurs, ou même à des administrateurs, non-autorisés. Une façon de normaliser une telle procédure est de définir une "enveloppe électronique sécurisée", du type PEM, X.400(88) ou PKCS #7, dans laquelle des attributs peuvent être conservés.

En faisant partie d'une politique de sécurité, des attributs peuvent être envoyés aux utilisateurs autorisés sous un tel format. Un résultat authentifié de l'annuaire aurait en fait deux signatures : une appartenant au DSA, se portant garant de son identité et de la source de l'information, et une autre appartenant à l'autorité administrative responsable de l'information en question. Cette dernière signature atteste donc de la véracité et de la maintenance de l'intégrité de l'information dans l'annuaire.

L'utilisation de ces enveloppes peut également contribuer à la sécurisation de la DIB en ce qui concerne son intégrité. Ceci est notamment important dans le contexte des informations de navigation (de l'annuaire lui-même mais aussi d'autres applications comme la messagerie) qui peuvent être conservées dans l'annuaire pour optimiser le routage et l'administration. Le même mécanisme s'applique aux listes de distribution dont la composition peut être stockée dans l'annuaire.

10.3. La Sécurité de la Messagerie X.400

La mise en oeuvre de mécanismes de sécurité dans des services de l'apport et de la complexité de la messagerie électronique X.400 exige un modèle de sécurité, d'une part cohérent avec le modèle du service, et d'autre part, qui prenne en compte le passarellage avec d'autres approches du même service. Ces caractéristiques ne se vérifient pas dans le modèle de sécurité défini pour X.400.

Dans cette section, nous exposerons les problèmes qui nous avons rencontré dans ce modèle ; problèmes pour lesquels nous essayons d'apporter des solutions plus viables et implémentables.

10.3.1. Des Lacunes du Modèle de Sécurité de X.400

Outre le manque de clarté et la redondance retrouvés parmi les mécanismes du modèle de sécurite de X.400, tels que le MOA et le CIC décrits dans la Section 7.3.2.3 de ce travail, ce modèle spécifie un service de preuve de remise qui ne peut pas être implémenté à cause d'incohérences par rapport à l'architecture de X.400. Nous avons abordé ce problème dans la Section 7.3.2.3.4. Nous analyserons les solutions possibles dans 10.3.3.

Une lacune peut-être "mineure" des spécifications de la sécurité de X.400 réside dans fait que le nom de l'expéditeur d'un message n'est pas signé. Par conséquent, il est possible de modifier le nom d'O/R d'un message de la sorte qu'un intrus peut apparemment prendre l'identité de quelqu'un d'autre en tant qu'expéditeur X.400. L'intrus peut même signer le message correctement avec sa clé privée. Cette lacune autorise donc un intrus à tromper les destinataires d'un message en leur faisant croire qu'un message est signé par quelqu'un d'autre. Au passage cela montre une faiblesse ou tout au moins une fonctionnalité ambigüe dans le modèle de sécurité de X.400.

Dans le cadre des tests pilotes du projet Password nous avons fait le test suivant : nous avons expédié aux participants du pilote français, un message ayant le nom d'expéditeur "*C=FR; ADMD=Atlas; PRMD=Inria; S=Paul-Andre Pays*" mais signé par "*C=FR; O=E3X; OU=OSI; CN=Alain Zahm*".

A la réception de ce message, le prototype d'UA sécurisé, qui nous avons mis à disposition des participants, a vérifié avec succès la clé publique du signataire, puis la signature du message. Ensuite, l'UA a affiché à chaque destinataire, le nom de l'expéditeur du message (un nom d'O/R, dans la terminologie X.400) comme s'il était le signataire du message, car le nom d'O/R constitue le mode d'identification des utilisateurs dans la messagerie X.400. Bien entendu, il existait un 'bug' dans l'UA prototype qui résidait dans le fait que l'interface[24] ne présentait pas le nom du signataire (un nom d'annuaire), mais uniquement le nom d'O/R. En fait, la confiance que l'on accorde au fait que la signature est correcte est trahie par le fait que le nom de l'expéditeur d'un message n'est pas signé.

Ce problème n'est pas exclusif à X.400. Le format de jetons d'authentification suit le modèle d'authentification spécifié par le cadre X.509. Par conséquent toute application basée sur ce cadre est exposée au même problème. Cela ne représente pas une lacune dans le cadre de la sécurité de l'annuaire car dans X.500 les utilisateurs sont identifiés vis-à-vis de ce service de la même façon dont ils sont identifiés dans leur certificat, i.e., le nom d'annuaire. Toutefois, ce problème se pose dans le cadre de toute autre application où les utilisateurs ne sont pas forcement identifiés par leur nom d'annuaire.

En fait, le problème est plus grave. La question qui se pose est la suivante : Peut-on garantir qu'un nom d'annuaire Y correspond au nom d'O/R W ? Cette question n'est pas exclusive à la

---

[24]Au passage, cela nous fait réfléchir à un autre problème potentiel qui est l'importance des interfaces homme-machine fiables.

messagerie X.400. Pour d'autres applications, il suffit de substituer "nom d'O/R" par "le mode d'identification" de l'application en question.

Peut-on faire confiance aux règles de "mapping", dans le cas où elles existent[25] ?

Ce problème est considéré comme une lacune mineure dans la mesure où la communauté où ces services pilotes sont déployés est constituée d'experts en réseaux, X.400, X.500 ou en sécurité. En contrepartie, dès que ces mêmes services seront mis en place dans une communauté plus répandue et non nécessairement versée en ces matières, cette même lacune peut entraîner des conséquences beaucoup plus importantes.

Outre les aspects décrits ci-dessus, le placement de tous les services de sécurité dans les enveloppes de message (i.e., dans les éléments des protocoles X.400) implique qu'une implémentation sécurisée de la messagerie X.400 ne puisse interfonctionner qu'avec d'autres implémentations X.400.

La réalisation décrite dans ce travail, bien que conforme aux Recommandations X.400, constitue seulement une des possibilités. En effet, de nombreux services de sécurité peuvent être fournis en utilisant les éléments de sécurité définis dans les standards X.400, et beaucoup d'entre eux ne sont pas appropriés à tous les systèmes. De plus, dans bien des cas, un même service peut avoir plusieurs alternatives de réalisation. Par conséquent, pour que deux systèmes X.400 sécurisés puissent interopérer, ils doivent être en concordance avec un sous-ensemble commun et détaillé des standards.

10.3.2. Un Service Complémentaire à la Classe S0a

L'utilisation d'UA déportés accédant à des messages à travers un Message Store (MS) connaît actuellement un intérêt croissant dû à l'expansion de l'utilisation de micro-ordinateurs portables ou domestiques qui sont normalement des machines de capacité de stockage réduite. Ainsi, en plus des services de sécurité de bout-en-bout de la classe S0a, un autre service de sécurité serait utile afin d'offrir un ensemble minimal de services sécurisés de messagerie, à savoir, le contrôle d'accès au MS.

La liaison UA ↔ MS peut être considérée comme l'une des plus vulnérables [MRW88]. Le modèle de sécurité de X.400 permet l'implémentation d'un mécanisme d'authentification forte pour l'accès aux entités du MHS. En particulier, ce mécanisme peut être utilisé pour sécuriser l'accès au MS.

Le service d'accès sécurisé au MS peut être réalisé à travers l'utilisation de jetons d'association[26]. Toutefois, ce mécanisme est restreint à l'établissement de l'association et donc n'offre pas des services de sécuité pendant le restant de l'association. Cependant, le format de jetons

---

[25]Cette réflexion vient confirmer l'observation qui nous avons fait dans la Section10.1.2, à propos du format unique de certificats X.509 vis-à-vis des plusieurs types de services et d'utilisateurs.

[26]Le format de ces jetons est dérivé du format de jeton asymétrique, définit dans la Partie II, Section6.2.2, de ce travail.

d'association permet l'échange de clés qui peuvent être utilisées pour fournir ces services pendant la durée de l'association. Bien que les normes X.400 ne fournissent pas les moyens de le faire, l'adoption d'une solution particulière au niveau du protocole P7 [X.413], entre UA ↔ MS, n'a pas d'impact sur le fonctionnement réparti du MHS. Cette solution particulière consiste à utiliser le jeton d'association de la manière suivante :

(1) lors de l'établissement de l'association avec le MS, l'UA choisit deux nombres aléatoires N1 et N2, de 64 octets chacun, et différents pour chaque association pour prévenir les attaques par re-jeu,

(2) *N1* est alors inclus dans le champ "*signed-data*" du jeton,

(3) *N2* est chiffré au moyen de la clé publique du MS et ensuite inclus dans le champ "*encrypted-data*" du jeton,

(4) le jeton est ensuite signé au moyen de la clé privée de l'utilisateur.

Le nombre N1 sert comme "challenge" et le nombre N2 comme secret partagé et qui sera utilisé pour authentifier le MS et l'utilisateur au cours de la session.

A la réception d'un tel jeton, le MS vérifie la signature du jeton et vérifie que le nombre aléatoire N1 ne fait pas l'objet d'un re-jeu. Le MS vérifie également la date-heure indiquée dans le jeton à fin de s'assurer que celui-ci est "frais". Si le jeton est valide, le MS récupère la valeur de N2 en le déchiffrant avec sa clé privée.

Pour répondre à l'utilisateur, le MS produit un autre jeton, contenant N1 dans "*signed-data*", signé avec sa clé privée. A la réception du jeton de réponse, l'UA de l'utilisateur vérifie la valeur de N1 et la signature du jeton. Si la vérification réussit, la session peut alors procéder.

Pour garantir l'authenticité et l'intégrité des données par la suite de la session, nous proposons l'utilisation d'une méthode d'authentification/intégrité sans chiffrement, mais qui utilise N2 comme secret partagé, couplé à une fonction de scellement de haute assurance. Le secret sert à authentifier les parties et la fonction de scellement sert à protéger le secret et à garantir l'intégrité des données. Pour ce faire, l'UA et le MS attachent à chaque requête, résultat ou erreur, une signature calculée de la façon suivante :

(1) Soit D les données originelles à échanger, i.e., une requête, un résultat ou une indication d'erreur. Le nombre N2 est ajouté à D, à la fois comme préfixe et suffixe, formant D' tel que :

$$D' ::= N2 \,||\, D \,||\, N2$$

(2) Ensuite, une fonction de scellement à sens unique et de haute assurance *h* est appliqué à *D'*, formant ainsi la signature *S* telle que :

$$S ::= h\,(D') = h\,(N2 \,||\, D \,||\, N2)$$

L'idée de remplacer le chiffrement par une méthode de couplage d'un secret à une fonction de scellement, vise à réduire les coûts, liés au temps et à la bande passante, inhérents au chiffrement. Une telle approche semble intéressante dans le cas de la liaison UA ↔ MS, notamment dans le contexte de communications mobiles.

La méthode originelle utilise le secret partagé seulement comme préfixe des données à sceller et a été proposée par Crocker et Kent pour utilisation dans le protocole SNMP (**S**imple **N**etwork **M**anagement **P**rotocol), le protocole pour l'administration des réseaux Internet [CFSD90, GMD92, GM93]. La méthode hybride, i.e., utilisant le secret comme suffixe et préfixe, a été proposée par Tsudik [Tsu92] dans le but de minimiser la vulnérabilité à des attaques connues. La méthode hybride double le montant des opérations nécessaires à une attaque réussie contre la fonction de scellement.

Dans notre variante, la vulnérabilité est encore réduite car le secret partagé est différent à chaque session. Le niveau de sécurité peut être renforcé si les parties s'accordent sur deux valeurs différentes pour le suffixe et le préfixe. Ainsi, dans notre exemple, l'UA choisirait, en fait, *N1*, *N2* et *N3*, dont *N2* et *N3* seraient inclus dans "*encrypted-data*" pour être utilisées comme préfixe et suffixe, respectivement. Un exemple de fonction de scellement à sens unique de haute assurance est le MD5 [Riv92].

Pour minimiser le temps et l'effort de vérification initiale, l'utilisateur et son MS peuvent être certifiés par la même CA. Ainsi, les parties n'auront qu'un certificat à valider.

10.3.3. Le Service de Preuve de Remise

Les problèmes conceptuels de ce service ont été detectés au sein du groupe Password et décrits dans la Section 7.3.2.3.4. Toutefois, aucune solution n'a été définie. Dans cette section, nous discutons des problèmes liés à la réalisation de ce service ainsi que différentes solutions.

La réalisation de ce service se heurte à de nombreux problèmes. Premièrement, le destinataire ne peut pas être forcé à signer la preuve de remise. Deuxièmement, ni le destinataire ni l'UA ne sont systématiquement disponibles lors de la réception d'un tel message. Dans un tel cas, le MTA pourrait, en cas extrême, retenir le message pour une période de temps donnée au bout de laquelle il considère l'opération de remise comme incomplète et retourne un avis de non-remise à l'expéditeur.

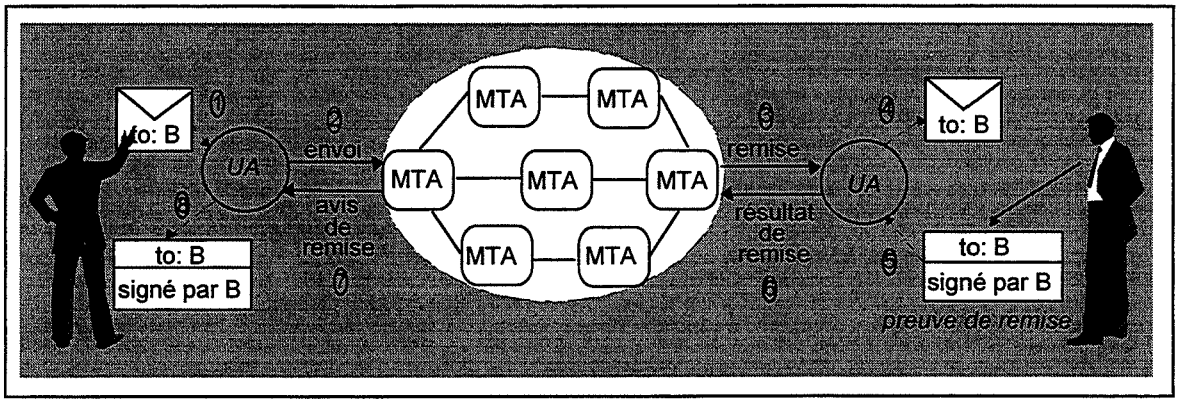

Figure 21. Mécanisme de Preuve de Remise

Les problèmes de réalisation du service de preuve de remise sont amplifiés dans le cas où l'utilisateur reçoit ses messages via un Message Store (MS). Dans un tel cas, ce n'est pas l'UA du destinataire qui génère l'avis de remise mais le MS, car c'est celui-ci qui dialogue avec le MTS.

Or, en principe, le MS n'a pas d'accès à la clé privée de l'utilisateur et donc ne peut pas signer la preuve de remise.

Dans un tel cas, où même dans le cas ou le destinataire ne possède pas un MS mais un UA demeurant disponible, deux solutions différentes pourraient être envisagées :

(1) Donner au MS ou à l'UA l'accès à la clé privée de l'utilisateur.

Or, faire partager la clé privée de l'utilisateur avec le MS (ou l'UA) pour que celui-ci signe la preuve de remise à la place de l'utilisateur constituerait une énorme atteinte à la sécurité de ce dernier. De plus, cela entraînerait une dégradation de la nature du service, surtout dans le cas de messages confidentiels, car le MS (ou l'UA) serait capable de déchiffrer le contenu du message et de produire la preuve de remise à partir du contenu en clair. Ainsi, la preuve de remise ne pourrait pas être considérée comme un service de non-répudiation de remise.

(2) Attribuer une paire de clés, et donc un certificat, au MS ou à l'UA.

Cette solution nécessite un moyen de distinction entre un certificat de MS, un certificat d'UA et un certificat d'utilisateur qui, dans la quasi-majorité des implémentations, partagent le même nom d'O/R. Comme nous l'avons exposé dans la Section 10.1.2.4, le format de certificat X.509 ne permet pas une telle distinction.

De plus, même si une paire de clés et un certificat sont attribués au MS (ou UA) pour qu'il soit chargé de signer[27] la preuve de remise, cela ne signifie pas que le message a été lu par l'utilisateur final. La politique de sécurité doit distinguer clairement les deux cas.

Là aussi on peut constater une dégradation de la nature du service car le service de non répudiation ne peut pas être offert.

Ainsi, la solution (1) n'est pas acceptable et la solution (2), d'une part, nécessite la modification du cadre X.509 et d'autre part provoque une dégradation de la nature du service. En fait, le problème vient du fait que le service de preuve de remise soit lié au protocole P3, qui en fait constitue un protocole d'accès au MTS et non un protocole de bout-en-bout.

La solution souhaitable serait de placer ce service au niveau du protocole P2, le protocole X.400 défini pour l'échange de messages interpersonnels. Cela dissocierait ce service du dialogue entre UA ↔ MTA et MS ↔ MTA, et permettrait la réalisation du service sans autant d'impact sur le système résultant et sur la nature du service lui-même. Effectivement, une solution élégante serait de retourner une "preuve de réception" dans la notification de réception (IPN - **I**nter**P**ersonal **N**otification) propre au protocole P2. Cependant, la spécification de l'IPN ne permet pas l'inclusion d'un tel élément. Ainsi, la solution qui reste est la définition d'une partie de corps spécifique à cet effet. Nous discuterons cela dans la Section 10.3.4.

Dans tous les cas, il faut rappeler que pour compléter la réalisation de ce service, l'UA de l'expéditeur est obligé de garder des informations du message originel pour pouvoir, plus tard, vérifier la signature de la preuve de remise.

---

[27]Bien entendu, dans le cas de messages confidentiels, le MS (ou l'UA) signerait la preuve de remise à partir du contenu chiffré.

10.3.4. X.400 *versus* PEM

Les services de sécurité définis dans la version 1988 des Recommandations X.400 sont fournis à travers les éléments de service des protocoles P3 et P1. Ce choix d'emplacement de services de sécurité serait convenable du point de vue d'un opérateur. Par contre, la vision et les besoins d'un utilisateur final sont différents. L'utilisateur n'a pas d'accès ni connaissance directe des enveloppes P3 et P1 d'un message. De plus, l'utilisateur peut vouloir envoyer, dans un même message, des données confidentielles mélangées à des données publiques. Ceci n'est pas possible dans le modèle de sécurité de X.400. Les utilisateurs ont besoin d'une vraie vision "bout en bout" de la sécurité.

De plus, la sécurisation des divers types de message peut dépendre du type de données concernées, par exemple, la sécurisation de messages du type télécopie peut être différente de celle appliquée à des messages du type IA5.

Tous ces aspects justifient le besoin d'application de mécanismes de sécurité au niveau du contenu de messages. Dans ce paragraphe nous discutons les caractéristiques principales dont une telle solution doit disposer.

En fait, une solution souhaitable serait la définition d'une *partie de corps sécurisée*, offrant des services de sécurité de bout en bout et permettant la sécurisation de tout type de document électronique. D'une part, une telle solution permettrait d'appliquer des services de sécurité de manière générique, d'autre part, elle admettrait l'inclusion de données spécifiques au type de corps, permettant ainsi la spécification de types, sémantiques et politiques propres à chaque type de corps ou application sécurisée.

L'application de mécanismes de sécurité au niveau du contenu a été déjà définie dans la messagerie Internet, à travers l'approche PEM. La définition d'une partie de corps sécurisée dans X.400 pourrait s'inspirer de l'approche PEM pour les raisons suivantes :

- prendre l'avantage d'une définition existante,
- permettre la conversion entre les messages sécurisés X.400 et PEM.

Cette partie de corps sécurisée doit permettre l'utilisation d'algorithmes cryptographiques symétriques et asymétriques et offrir les services suivants :

(1) l'authentification de l'expéditeur,

(2) l'intégrité de messages,

(3) la confidentialité des contenus,

(4) la détection de la manipulation de messages (remplacement, insertion, suppression ou mise en désordre d'une séquence de messages),

(5) l'adjonction de "multi-signatures",

(6) la preuve de réception de messages.

En particulier, l'utilisation de techniques cryptographiques asymétriques permet aux services d'authentification de l'expéditeur et de preuve de réception de fournir également les services de non-répudiation d'origine et de réception, respectivement.

Dans cette nouvelle partie de corps, les éléments de sécurité doivent être signés par l'expéditeur. Dans PEM les éléments de sécurité ne sont pas tous protégés. Ainsi, il suffit qu'un intrus modifie un identificateur d'algorithme pour causer un refus de service.

Les services (1) à (3) sont offerts tant par X.400 que par PEM. Nous devons donc chercher à trouver un format permettant la conversion entre ces deux approches.

Les services (4) à (6) ne sont pas offerts par PEM. La fourniture du service (4) demanderait la définition d'un nouvel élément de service de sécurité, par exemple un jeton, permettant la spécification de numéros de séquence des messages associé à chaque message d'une serie, au niveau de chaque destinataire. La fourniture du service (6) nécessiterait d'un service d'avis (non offert dans RFC-822) ou la définition d'un type particulier de message.

Le service (6), i.e. de preuve de réception de messages, est proposé comme une solution aux problèmes liés à la réalisation du service de preuve de remise défini dans X.400 (voir 7.3.2.3.4 et 10.3.3). En pratique, ce service peut exiger la définition d'une partie de corps spéciale. Cette solution doit permettre de spécifier une preuve de réception pour chaque partie de corps d'un message reçu.

En ce qui concerne le service (5), il n'est offert ni par X.400 ni par PEM. Toutefois, dans bien des circonstances, il est nécessaire qu'un même document soit apprécié et signé par plusieurs parties. Le service d'adjonction de multi-signatures permettrait donc que chaque partie responsable de l'approbation d'un document puisse y joindre sa signature. De plus, il doit être possible de rendre ce service de deux manières différentes (voir Figure 22) :

(1) la première consiste à permettre que les signatures soient indépendantes les unes des autres,

(2) la deuxième consiste à permettre que les signatures soient imbriquées, impliquant ainsi l'engagement entre les plusieurs signataires.

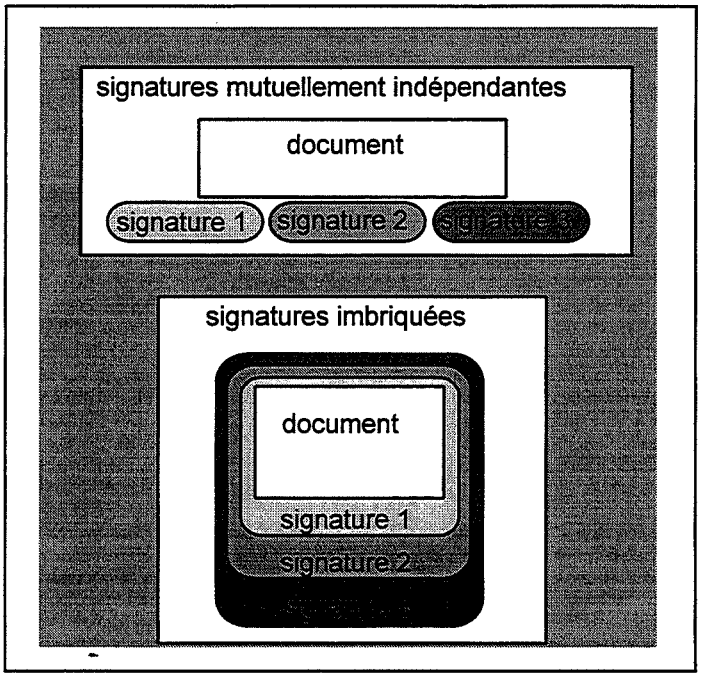

Figure 22. Service de Multi-Signatures

Une des grandes sources d'incompatibilité entre le modèle de sécurité de X.400 et PEM réside sur le modèle de gestion de clés publiques. PEM a son propre modèle de certification et définit un format et la sémantique des chemins de certification. X.400 ne définit aucun modèle de certification mais spécifie un format de chemin de certification qu'exige l'utilisation de certificats du type X.509 et qui peut rendre difficile le passerellage avec PEM et bien d'autres systèmes.

Un trait souhaitable de la partie de corps securisée serait de permettre l'indépendance vis-à-vis des modèles de gestion de clés et des formats des certificats échangés. Nous allons montrer dans la Partie IV de ce travail, qu'à travers l'utilisation des identificateurs d'objet, les utilisateurs peuvent spécifier le format de certificat et le modèle de certification auquel ils sont soumis, ainsi que la sémantique associée aux clés utilisées (par exemple, si l'utilisation d'un schéma particulier implique le service de non-répudiation).

Cela faciliterait non seulement le passarellage entre X.400 sécurisé et PEM, mais aussi l'utilisation de tout type de certificat (X.509, PKCS #6, ANSI9.17) et tout modèle de certification (Password, PEM, Chimæra), même ceux qui assument un espace de noms non structuré, comme PGP.

Un autre avantage de la solution partie de corps sécurisée par rapport à PEM est qu'elle permet la sécurisation de tout type de document, tandis que PEM ne véhicule que des messages en format textuel. Toutefois, il doit être noté que l'intégration des fonctionnalités MIME (Multipurpose Internet Mail Extensions) [BF92] dans la messagerie INTERNET peut permettre une telle évolution.

Il doit être noté que la mise en place d'une nouvelle partie de corps sécurisée n'empêche pas l'utilisation des éléments de sécurité spécifiés dans les protocoles P3 et P1. Ceux-ci demeurent utiles à la sécurisation de la structure de la messagerie elle-même et à la protection de ses ressources contre l'accès non autorisé.

10.4. D'autres Applications : L'ACSE et le restant de l'association

Comme nous l'avons déjà décrit, Password a choisi l'ACSE comme élément commun de la couche application à travers lequel des procédures de sécurité sont implémentées. Le problème qui se pose est que l'ACSE est appelé à chaque fois qu'une association de présentation est établie ou terminée, mais il n'est pas appelé pendant le transfert des données au cours de l'association. Ainsi, les services de sécurité pouvant être fournis à travers l'ACSE sont déterminés, voire limités, par cette caractéristique.

L'implémentation de services de sécurité applicables au cours d'une association n'a pas été définie au sein du groupe Password. Or, authentifier les parties au moment de l'etablissement de l'association et ne pas assurer la continuité de ce service, rend inutile l'effort initial d'authentification car un intrus peut observer l'établissement de l'association et usurper l'identité d'une des parties communicantes par la suite. Nous avons étudié ce problème et comme résultat nous proposons une solution, qui est décrite dans la suite de cette section.

Notre solution implique la définition du concept d'unité fonctionnelle dans les autres éléments de service commun de la couche application responsables de la gestion des opérations et du transfer fiable d'informations entre processus d'application, à savoir, le ROSE (**R**emote **O**pérations

Service Element) [X.219, X.229] et le RTSE (Reliable Transfer Service Element) [X.218, X.228]. Ce concept a été déjà défini et utilisé dans l'ACSE et dans les couches session et présentation. De façon générale, l'utilisation du concept d'unité fonctionnelle dans le ROSE et RTSE offrirait, à ces éléments de service, un mécanisme dynamique d'extensibilité. En particulier, cela permettrait la définition et l'utilisation d'une "unité fonctionnelle sécurité" dans ces modules.

Cette unité fonctionnelle servirait à assurer le transfert de paramètres de sécurité, de signatures et/ou de données chiffrées dans le cadre des primitives de services du ROSE et du RTSE (par exemple, RO-INVOKE, RO-RESULT et RO-ERROR, dans le cas du ROSE). Ces valeurs de sécurité seraient traitées comme un élément atomique par ces éléments de service. Leurs sémantiques seraient transparentes pour le fournisseur du service ROSE et RTSE.

La méthode décrite dans 7.3.3.1, dans le cadre de l'ACSE, pourrait être utilisée pour permettre la spécification d'identificateurs de mécanismes de façon à identifier, explicitement et de façon unique, l'ensemble de valeurs, d'options associées et de toute autre information nécessaire à l'interfonctionnement sécurisé des entités d'application sur une association d'application.

Une telle approche faciliterait la mise en oeuvre de mécanismes d'encapsulation à travers la définition d'une "enveloppe de sécurité" où les données spécifiques à chaque application seraient insérées, chiffrées et/ou accompagnées d'une signature et des paramètres de sécurité. Le format général d'une telle enveloppe pourrait avoir la définition suivante :

```
SecureEnvelope ::= OPTIONNALY-SIGNED  SEQUENCE {
  mechanism-name    [0]   IMPLICIT OBJECT IDENTIFIER OPTIONAL,
  mechanisme-value  [1]   CHOICE {
                            signed        [0] SecuredUserData,
                            encrypted     [1] ENCRYPTED SecuredUserData }
}

SecuredUserData ::= ANY DEFINED BY mechanism-name
```

Dans la définition ci-dessus, *SecuredUserData* correspond aux paramètres spécifiques aux applications. A titre d'exemple, nous pouvons citer le format d'accréditation défini par Password pour mettre en oeuvre les protocoles d'authentification à travers l'ACSE (voir 7.3.3.1).

Cependant, il doit être noté que le format de jeton défini par Password ne admet pas l'echange de données spécifiques aux applications, ni l'échange de données confidentielles. Une telle fonctionnalité serait interessant, notamment pour l'établissement de clés de session. Pour contourner cette déficience, nous proposons un nouveau format de jeton, *ExtendedToken*, ayant la définition suivante :

```
ExtendedToken ::= OPTIONNALY-SIGNED SEQUENCE {
    algorithm           [0]   AlgorithmIdentifier,
    claimant            [1]   DistinguishedName,
    verifier            [2]   DistinguishedName OPTIONAL,
    time1               [3]   UTCTime,
    random1             [4]   BIT STRING,
    time2               [5]   UTCTime OPTIONAL,
    random2             [6]   BIT STRING OPTIONAL,
```

```
    sequenceNumber      [7]   INTEGER OPTIONAL,
    toBeSigned          [8]   IMPLICIT EXTERNAL OPTIONAL,
                              – application specific additional data to be signed
    encryptionId        [9]   AlgorithmIdentifier OPTIONAL,
    encrypted           [10]  ENCRYPTED confidentialData OPTIONAL
}

confidentialData::= IMPLICIT EXTERNAL  – e.g., a BIT STRING indicating a session key
```

Cette nouvelle définition fournit un format général des jetons permettant la mise en œuvre de la confidentialité et l'échange des données signées spécifiques aux applications.

Le caractère général de la solution a comme point principal la simplicité. L'avantage de cette approche est la fourniture d'une solution commune à de nombreuses applications, évitant ainsi que chaque application définisse ses propres solutions, qui sont normalement basées sur l'utilisation de mots de passe.

De plus, la présence d'un identificateur d'objet permet la définition explicite de sémantiques attachées à chaque mécanisme, permettant ainsi que la solution satisfasse aux spécificités de chaque application et service.

Il est clair que cet effort doit être mené en conjonction avec les groupes de normalisation, responsables du cadre de sécurité dans OSI, pour que cette solution puisse être adoptée et déployée de façon standard. Pour contourner le manque de mécanismes d'extensibilité pour les éléments de service commun de la couche application, d'autres solutions moins élégantes pour l'implémentation de telles enveloppes sécurisées peuvent être envisagées à travers la définition de nouveaux contextes de présentation.

PARTIE IV

EVOLUTION DU CONCEPT DU DAS

11. INTRODUCTION
12. REORGANISATION DU SYSTEME
13. DESCRIPTION DES POLITIQUES DE CERTIFICATION
14. VERIFICATION DE CLES PUBLIQUES
15. LE SERVEUR DE DONNEES POUR L'AUTHENTIFICATION

Les besoins et l'importance des mécanismes de sécurité dans le monde des télécommunications sont, aujourd'hui largement reconnus. Dans certaines circonstances les parties communicantes d'un système distribué (principaux[27]) peuvent avoir besoin de s'assurer de l'identité de leur paires, par exemple, pour des implications légales ou financières. Les principaux peuvent être tenus responsables des actions commises en leur "nom" ou peuvent être amenés à payer des services obtenus sur la fois de leur identité. Un service d'authentification est fondamental pour la prestation de services de sécurité car l'identification est à l'origine de tous les mécanismes de sécurité, mais aussi de l'administration de ces activités, par exemple, la comptabilité (accounting) et l'audit.

[27]Un principal est une entité dont l'identité peut être authentifié, selon Needham et Schroeder [NS78].

Le manque de méthodes sûres et viables d'authentification à grande échelle constitue l'un des obstacles majeurs au développement d'une économie "informatisée". L'authentification forte et les signatures numériques permettraient l'existence et l'échange de toutes sortes de document légal sous forme électronique. Dans un tel contexte, l'existence d'une *infrastructure globale d'authentification* est extrêmement importante. Si une telle infrastructure était viable, des documents signés pourraient être échanges entre différents pays et sous différentes plates-formes ; ce potentiel d'interoperabilité est nécessaire pour qu'une vraie économie informatisée puisse se développer. Autrement dit, l'implémentation de mécanismes de sécurité à échelon mondial requiert la création d'une infrastructure globale pour l'authentification.

Dans la partie précédante nous avons présenté notre expérimentation dans le domaine de la sécurité pour les systèmes ouverts. Notre participation au projet Password et la conception de l'architecture SecUcomx nous a permis d'expérimenter le cadre d'authentification X.509. En particulier, nous nous sommes intéressés aux problèmes d'obtention de données pour l'authentification. Comme réponse à ces problèmes, nous avons conçu un élément particulier pour l'architecture SecUcomx, le DAS. Celui-ci constitue un module global à l'architecture et spécifique à l'accomplissement des tâches liées à l'obtention de données pour l'authentification.

Cela nous a fait détecter des problèmes ouverts en ce qui concerne l'utilisation du cadre X.509 pour la mise en place d'une infrastructure commune pour les services d'authentification. Nous avons exposé ces problèmes dans la Partie III, Section 10.1 de ce travail. La question de l'obtention de données pour l'authentification dans un environnement ouvert et hétérogène nous a poussé à étudier l'évolution du concept du DAS vers la conception d'une approche permettant la mise en place d'une infrastructure globale pour les services de sécurité. Ainsi, dans cette partie nous proposons des solutions aux déficiences du cadre X.509.

Le principal avantage du cadre d'authentification défini par la Recommandation X.509 est de fournir une infrastructure permettant la construction d'architectures de sécurité homogènes du point de vue de l'utilisateur. En effet, si de telles architectures deviennent viables, celles-ci permettront l'établissement d'un canal sécurisé entre deux principaux quelconques, ainsi la quasi-majorité sinon tous les utilisateurs souhaitera profiter des services sécurisés.

Il se trouve que X.509 constitue un cadre très général, pouvant accommoder toute une gamme de modèles de certification. Cependant, si d'une part de nombreux modèles de certification ont été définis à partir du cadre X.509; d'autre part ceux-ci ne constituent pas moins de mondes isolés les uns des autres. De plus, des aspects importants sont considérés comme des "questions locales", des structures de données contiennent des défaillances et manquent de précision au niveau sémantique. Comme résultat, le cadre X.509, tel qu'il est défini actuellement, ne permet pas la réalisation d'une infrastructure telle que mentionnée ci-dessus, dans la mesure où il ne permet pas à l'utilisateur final d'identifier les diverses sémantiques d'authentification, pour être capable de prendre des décisions précises basés sur des certificats.

Pour être applicables dans un environnement ouvert et hétérogène, les procédures d'authentification doivent disposer d'une infrastructure permettant de rendre explicites les sémantiques liées à la certification. Si les politiques et sémantiques de certification sont exposées de façon claire et bien définie, l'utilisateur final sera capable de "filtrer" des certificats dont le rapport "certificateur-certifié" est douteux et de prendre des décisions précises basés sur des certificats. D'ailleurs, si les schémas de certification peuvent être harmonisés d'une façon sûre,

claire et bien définie, alors une infrastructure globale d'authentification pourra se développer sans le besoin d'un modèle de certification commun à tous les utilisateurs.

Cette partie se décompose en cinq chapitres. Le premier chapitre introduit les problèmes traités et donne un aperçu général de notre approche. Le deuxième chapitre propose une réorganisation modulaire du système. Le troisième chapitre présente une technique pour la description de politiques de certification. Le quatrième et cinquième chapitres traitent, respectivement, des problèmes liés à la vérification et à l'obtention de clés publiques dans un environnement ouvert et hétérogène.

CHAPITRE 11
INTRODUCTION

11.1. Motivations

Dans 10.1 de la Partie III, nous avons détaillé nombre de problèmes ouverts du cadre d'authentification X.509. Dans cette section nous mettons en evidence d'autres aspects négligés par ce cadre.

11.1.1. X.509 et les Modèles de Confiance

L'utilisation d'une autorité de certification comme service d'introduction constitue une méthode de distribution de clés plus efficace que celle où les parties communicantes obtiennent les clés publiques les unes des autres par des moyens hors bande. En contrepartie, cela crée un point central d'autorité et de confiance. Dans une grande organisation, il se peut qu'une seule entité ne soit pas suffisament de confiance pour enregistrer et de gérer les clés publiques de tous les individus. Lorsque l'authentification doit être appliquée dans un contexte plus ample, par exemple entre organisations, le concept d'une CA unique est impraticable.

Le déploiement de services d'authentification à grande échelle basés sur l'utilisation de crypto-systèmes asymétriques, exige une disponibilité répandue de clés publiques, et donc un mécanisme décentralisé pour la distribution des certificats. Ainsi, il est nécessaire d'établir un *modèle de confiance* (aussi appelé modèle de certification ou de distribution de CA), tout en préservant un niveau de sécurité adéquat et la conformité avec les standards. Cela correspond à la définition du rôle, de l'emplacement dans l'espace de noms et de la juridiction des autorités de certification dans le système.

Le cadre d'authentification X.509 ne spécifie pas un modèle de certification. En fait, la norme n'impose aucun critère de distribution de CA, mais suggère seulement une distribution *hiérarchique* des CA, isomorphe à la hiérarchie de l'arbre d'information de l'annuaire X.500 et *commune* à tous les utilisateurs. De plus, ce modèle est tel que toutes les CA sont égales, i.e.,

toute CA est autorisée à signer tout type de certificat et à certifier tout type d'entité. Bien que ce modèle soit intérieurement consistent, il ne represente pas les besoins des utilisateurs ni l'hétérogénéité du système : il ne serait realiste d'envisager que toutes les CA en vigueur auront le même niveau d'assurance et le même degré de responsabilité. Du point de vue de l'utilisateur final, certaines CA seront considerés plus dignes de confiance que d'autres.

Théoriquement, un sous-ensemble quelconque de CA connectées par des certificats ou certificats croisés peut être considéré comme un modèle de distribution de points de confiance. Par contre, trouver une distribution parfaite des CA est un problème difficile à résoudre. Quelle que soit l'approche, hiérarchique ou non-hiérarchique, la compromission d'une CA peut mettre plusieurs sites en danger. Une distribution optimale de CA serait celle où :

- la sécurité de chaque CA dépend seulement d'elle même,
- l'introduction d'une CA pirate dans le système est difficile, sinon détectable,
- une CA dont la compromission est détectée peut être facilement supprimée du système,
- le respect des politiques de sécurité peut être imposé et surveillé,
- l'architecture est raisonnablement proche des besoins réels de la majorité des utilisateurs,
- la gestion est simple,
- toute information nécessaire à sa composition/gestion/vérification peut être obtenue de manière sûre.

Il se trouve que les aspects ci-dessus sont faciles à établir mais difficiles à réaliser. La technologie disponible aujourd'hui ne permet pas l'implémentation simple d'une distribution optimale des Autorités de Certification. De plus, la mise en place d'une topologie unique des CA à grande échelle se heurte à des problèmes politiques et administratifs. Ainsi, il n'est pas raisonnable de considérer qu'une topologie universelle de CA sera adoptée. Toutefois, le besoin d'une infrastructure de certification, globale à plusieurs communautés différentes, regroupant des politiques de certification et semantiques d'authentification diverses a amené la définition de divers systèmes de certification comportant différents types de CA, dont chacun peut jouer un rôle différent. Nous en avons décrits diverses approches dans la Partie II (PEM, Chimaera et PGP) et le modèle Password dans la Partie III.

11.1.2. Manque de Precision du Cadre X.509

Lorsqu'une CA émet un certificat pour une certaine entité, ceci peut avoir une sémantique particulière selon le type de CA, le type d'entité et le système de certification en question. Par exemple, dans le modèle PEM lorsqu'une PCA attribue un certificat à une CA Organisationnelle cela exprime une délégation d'autorité. En retour, cette dernière doit obéir à un ensemble de prérogatives établies par la PCA, comme l'illustre la Figure 23.

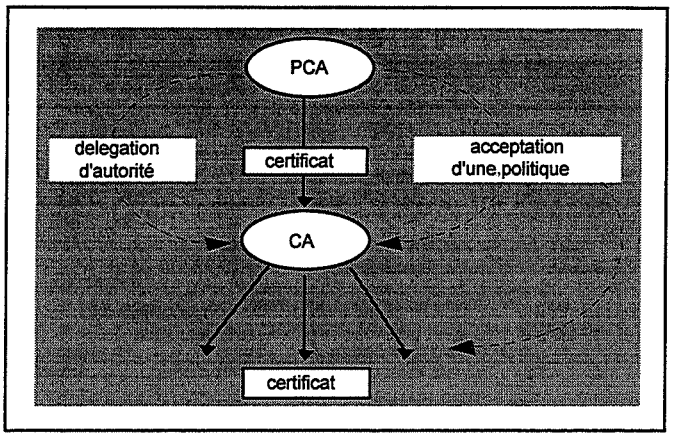

Figure 23. Rapport entre une PCA et une CA dans le modèle PEM

De même, dans le modèle Chimæra, lorsque deux CA de haut niveau appartenant à des domaines différents se certifient, ceci ne délègue pas d'autorité mais exprime la possibilité d'interfonctionnement entre ces domaines, à partir de la reconnaissance d'un certain niveau de confiance, comme le montre la Figure 24. Ces certificats n'autorisent pas la CA de Haut Niveau 1 à certifier des entités appartenant au Domaine 2 et vice-versa.

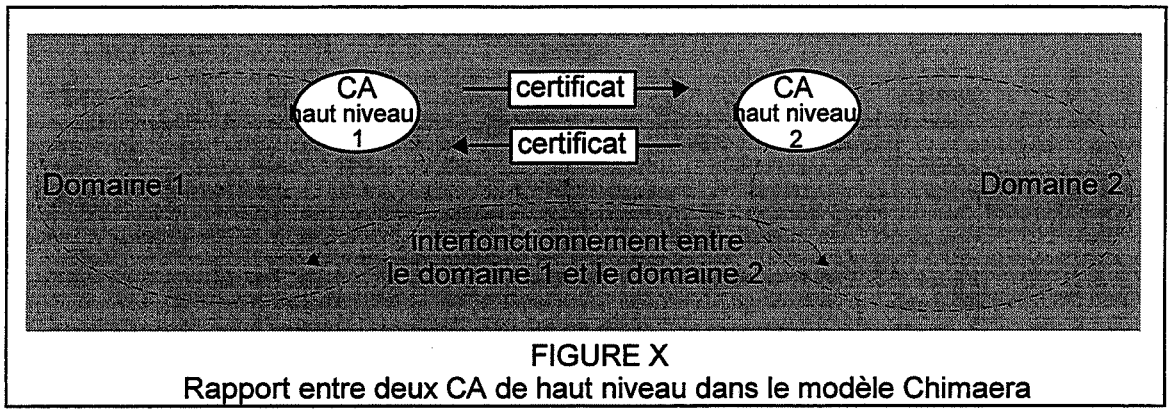

Figure 24. Rapport entre deux CA de Haut Niveau dans le Modèle Chimæra

Il se trouve que le cadre X.509, bien qu'il puisse servir comme cadre de base reste assez "pauvre" en ce qui concerne la certification dans un environnement ouvert et hétérogène. En plus des défaillances au niveau syntactique déjà publiées, d'autres défauts peuvent être identifiés, au niveau sémantique. Par exemple, le format de certificat X.509 (voir 5.1. et Annexe A) ne permet de distinguer de façon précise entre les certificats d'utilisateur, de CA et les certificats croisés ; i.e., il n'existe pas un moyen standard pour déterminer la *sémantique d'un certificat.*

Tel que ce cadre est défini actuellement, il ne permet pas de distinguer entre le certificat de la Figure 23 et ceux de la Figure 24, ni entre une PCA et une CA (dans la Figure 23), ni entre une CA de haut niveau et une de bas niveau (dans le modèle Chimæra). Il est encore plus difficile d'identifier la politique que la CA de la Figure 23 doit respecter lorsqu'elle certifie d'autres entités.

Par conséquent, les Figures 23 et 24 ci-dessus ne peuvent pas être interprétées telles quelles par l'utilisateur[28]. En effet, du point de vue de l'utilisateur final, la certification conforme au cadre X.509, est interprétée tel comme illustré par la Figure 25.

En particulier, en ce qui concerne la certification croisée, l'objectif est de rendre plus facile la transition et l'interfonctionnement entre deux domaines différents. Les sémantiques liées à la certification peuvent varier considérablement d'un domaine de gestion à l'autre. Toutefois, le cadre X.509 ne fournit pas les moyens de désigner un certificat comme inter-domaines ou intra-domaine, i.e., un certificat croisé ou un certificat ordinaire de CA. Du point de vue de l'utilisateur, il est quelque peu trompeur de franchir les frontières des domaines de sécurité sans être averti, puisque cela peut impliquer des changements de politiques et de sémantiques de certification. De plus, une telle défaillance peut faire qu'une procédure d'authentification réussisse tandis qu'elle devrait échouer. Ainsi, lors des procédures d'authentification, l'établissement de chemins de certification ne doit pas omettre l'éventuel interfonctionnement entre deux domaines.

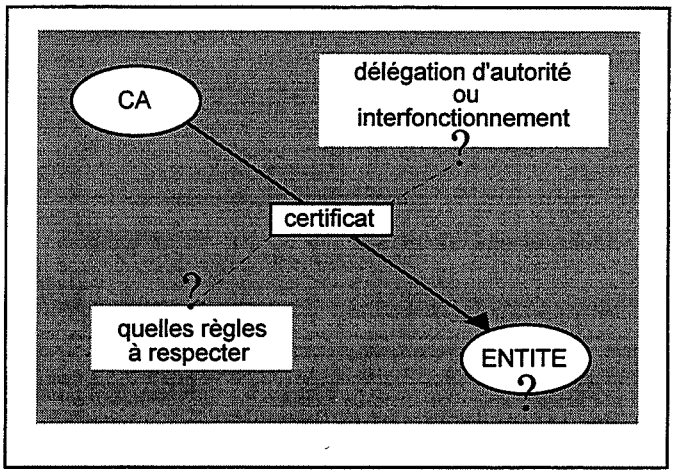

Figure 25. Interprétation d'un Certificat X.509

En l'absence de méthodes spéciaux et standard, il n'existe pas des moyens magiques pour la designation et la différentiation des particularités (portée, semantique, régles d'usage) d'un certificat. L'alternative, pour la distinction tant entre utilisateur et CA et entre catégories de CA, que de la juridiction d'une CA, est :

(i) du côté des modèles : l'imposition de conventions de nommage, ce qui peut affecter la liberté des organisations pour structurer l'espace de noms qui leur a été délégué,

(ii) du côté des applications : la maintenance d'une liste protégée de noms de CA associée au type des CA en question,

(iii) du côté des utilisateurs : l'exigence que ceux-ci examinent tous les certificats de chaque chemin de certification établi ou reçu.

En particulier une convention de nommage utilisée pour définir la juridiction des CA est connue comme "*subordination de noms*". Ceci signifie que sous de tels modèles, tous les utilisateurs sous

---

[28]Le terme désigne un utilisateur humain ou application cherchant à construire ou à valider un chemin de certification.

la juridiction d'une autorité de certification doivent avoir leur nom d'annuaire subordonné au nom de cette CA. Bien que cela constitue une méthode pratique pour la définition de la juridiction des CA, les conséquences sont moins pratiques, voire inappropriées, à savoir :

(1) cela mélange les concepts de hiérarchie de nommage et de certification,

(2) si un utilisateur possède plusieurs certificats émis par des CA différentes, alors cet usager aura également plusieurs noms d'annuaire différents,

(3) les organisation et les utilisateurs perdent leur liberté de choisir leur nom d'annuaire ; de plus les noms risquent d'être moins descriptifs, moins intuitifs et moins précis,

(4) cela ne permet pas une plus grande granularité, par exemple :

- une seule CA ne peut pas établir sa juridiction sur des parties disjointes de l'espace de noms,
- une CA ne peut pas être désignée comme responsable de la certification d'entités appartenant à une classe d'objet spécifique.

En particulier, les aspects (2) et (3) constituent une dégénération du rôle des noms d'annuaire dans la certification, à savoir la possibilité d'avoir des noms uniques, précis et descriptifs.

Bien que les systèmes de certification puissent imposer des conventions de nommage, pour pallier au problème de la distinction entre les types de CA, cela ne résout pas le problème d'identification de politiques de certification. Cela n'offre pas non plus les moyens permettant la distinction et l'identification de modèles de certification. Par conséquent, la fourniture de services de sécurité basés sur la distribution de certificats reste très dépendante des modèles de certification spécifiques.

De plus, les mesures (ii) et (iii) ci-dessus sont fort coûteuses et ne sont que des faux palliatifs puisque les utilisateurs ne sont toujours pas en mesure de prendre de décisions précises basées sur des certificats, surtout sur ceux signés par des CA d'autres domaines.

L'émergence de modèles de certification pouvant regrouper divers types d'entité, politiques et sémantiques d'authentification, requiert la définition d'un *format commun* pour la description de ces éléments. Un tel format commun doit permettre :

- aux certificateurs de rendre explicite les sémantiques liées à la certification et les règles d'usage des certificats émis,
- la mise en vigueur de différentes politiques de certification,
- l'identification sûre des modèles de certification, des classes d'utilisateur, des classes CA,
- la détermination automatique des politiques sous lesquelles les certificats sont émis,
- l'accommodation de différents modèles de certification, tout en permettant l'interopérabilité entre eux et en préservant la sécurité des mécanismes,
- son traitement automatique par les outils chargés de la manipulation de certificats, i.e., l'automatisation de la validation de certificats.

Cela constitue l'une des lignes principales de notre travail. Dans le Chapitre 13 nous presenterons une technique pour la description de politiques de certification. Dans le Chapitre 14 nous

proposerons une methode d'évaluation de certificats prennant en compte, entre autres, les descriptions de politique.

11.1.3. Diversité, Sécurité ou Interopérabilité ?

Bien que la réplication des CA resoude le problème de centralisation d'autorité et de la distribution de certificats, cela aggrave le problème de la confiance accordée aux CA et aux certificats.

L'une des questions liées au concept de confiance est qu'il n'y a pas de définition formelle à cet égard. D'autant plus que le cadre X.509 ne prend pas en compte l'hétérogénéité du système, i.e., il ne permet pas de distinguer entre ces divers types de principal et de CA, ni d'exprimer le montant de confiance et la semantique accordée aux certificats.

Par consequent, l'expansion des architectures de sécurité ainsi que la multiplicité de types de principaux et de CA mettent en cause la confiance accordée aux chemins de certification. Même si un certain chemin de certification est cryptographiquement valide, il peut être non-conforme au(x) modèles(s) de certification et à la politique de sécurité en vigueur.

En conséquence des problèmes décrits au cours de ce chapitre ainsi que de ceux décrits dans le Chapitre 10.1 de la Partie III, l'utilisateur final est confronté à tout un ensemble confus de graphes de certification, où il n'est pas capable de distinguer les types de principal, de certificat, de CA, et encore moins les modèles de certification.

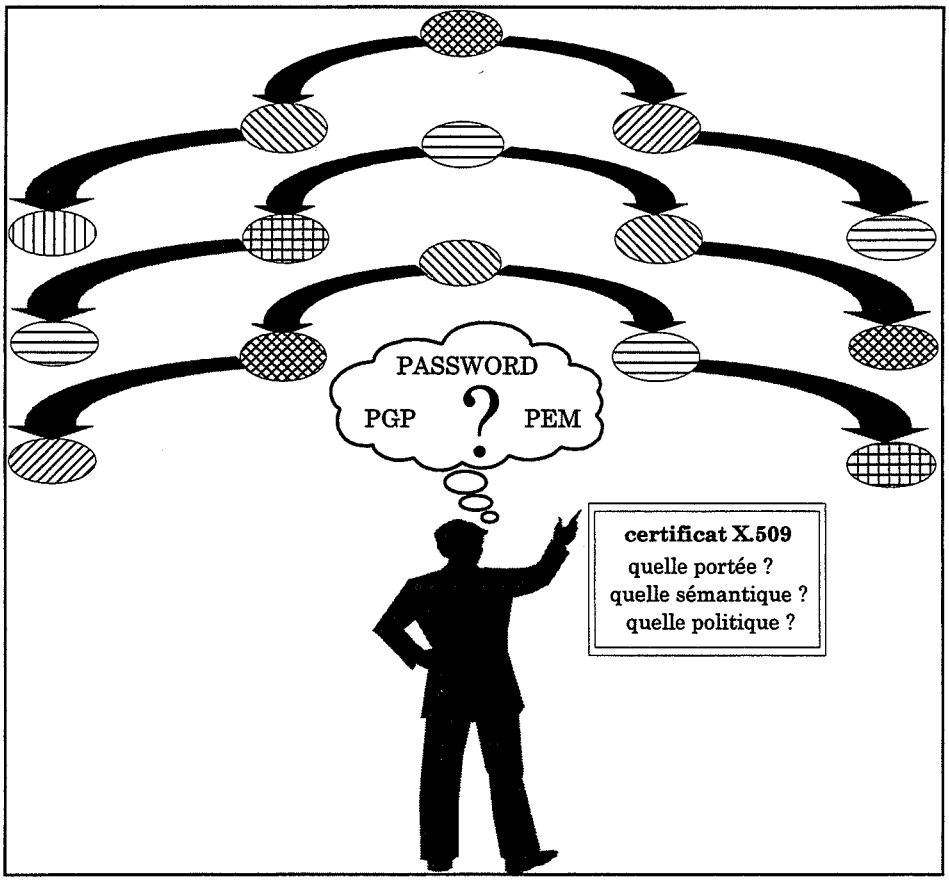

Figure 26. Utilisateur confronté aux Graphes de Certification

Un autre problème qui se pose concerne le choix du service de noms qui servirait de dépositaire pour les données publiques d'authentification, à savoir : les certificats d'utilisateurs, les certificats des CA, les certificats croisés et les listes de révocation. Le problème de la dépendance vis-à-vis de l'annuaire X.500 a été évoqué dans 10.1.4.

De plus, comme nous l'avons déjà décrit, il existe au moins trois types de structures des données actuellement définies pour représenter des chaînes de points de confiance :

- Certification Path,
- Certificates,
- PEM Forward Certification Path.

Nous avons décrit chacun de ces types dans 8.4.1. En fait, ces différentes structures constituent des variations, introduites par différentes applications, sur le même thème, à savoir la chaîne de *certificats*. En pratique, les applications sécurisées introduisent ces chemins comme un moyen de permettre à un utilisateur de fournir à d'autre(s) utilisateur(s) les données nécessaires à son authentification.

En plus des divers formats de chemin de certification, il existe plusieurs formats de certificat, par exemple PKCS #6 [PKCS #6] et ANSI X9.30 [ANSI93], bien que ceux-ci soient dérives du format X.509. De plus, il est possible que d'autres formats soient définis à l'avenir car le format de certificat X.509 contient actuellement certaines défaillances. Par ailleurs, la syntaxe de certificats X.509 est actuellement dans sa deuxième version, et des rapports d'anomalie à la version 1993 de la Recommandation X.509 réclament une nouvelle version de certificats contenant un champ d'extensibilité.

Nous avions déjà évoqué (dans 8.4) la complexité liée à l'obtention de l'information d'authentification. Toute cette diversité de modèles de certification et de formats de données, ainsi que leurs possibilités d'évolution dans un avenir proche, demande une approche modulaire et évolutive. Il existe un besoin croissant de mécanismes spécifiques à la construction et vérification de chemins de certification. De tels mécanismes doivent être capables d'isoler ces procédures de façon à alléger les applications et à faciliter l'évolution des architectures de gestion de clés et leur indépendance vis-à-vis des modèles de certification et des services de noms.

11.2. Une Nouvelle Approche

La mise en place de mécanismes de sécurité à grande échelle exige l'établissement d'une infrastructure commune pour l'authentification. En contrepartie, outre les problèmes déjà décrits, la réalisation d'une telle infrastructure se heurte à des problèmes politiques et administratifs, et par conséquent exige une organisation modulaire de tout le système pour que l'on puisse accommoder différentes solutions tout en conservant l'interopérabilité et la sécurité des mécanismes. Ce sujet a comme origine la question de la gestion de clés.

L'objectif de la gestion de clés est de fournir des procédures sûres pour la manipulation du matériel cryptographique utilisé pendant les mécanismes de sécurité. En contraste avec d'autres services de sécurité, plusieurs aspects de la gestion de clés sont traités plutôt par l'expérience

pratique que par la fondation d'une théorie. Les systèmes de gestion de clés dépendent, généralement, du type de clés qui seront utilisées, des propriétés de l'environnement de déploiement et de l'application de ces clés [FL93]. Les considérations les plus importantes dans ce contexte sont les menaces contre lesquelles le système doit être protégé et la structure architecturale et physique du système.

Un des problèmes centraux de gestion de clés est la distribution initiale de clés, i.e., la mise en place de matériel cryptographique dont l'origine, l'intégrité et probablement la confidentialité (dans le cas des crypto-systèmes symétriques) doivent être garanties. Après cette distribution initiale, les parties communicantes doivent être capables d'obtenir, en sécurité, les clés des unes et des autres pour l'établissement des communications sécurisées. Cela est considéré comme un pré-requis par les services de sécurité basés sur des mécanismes cryptographiques.

Dans un environnement ouvert et hétérogène, l'infrastructure de gestion de clés doit aussi permettre l'authentification des entités qui peuvent être des diverses catégories d'utilisateur et/ou des ressources telles que des serveurs et des applications. Pour être applicables dans un tel environnement, les procédures d'authentification doivent disposer d'une infrastructure permettant, à la fois :

- la mise en œuvre et le respect des politiques de securité,
- l'indépendance vis-à-vis des modèles de certification,
- l'harmonisation des modèles de certification,
- la différenciation entre les divers types de principal, de CA et de modèle de certification,
- l'indépendance vis-à-vis des formats de données,
- l'indépendance vis-à-vis des services de noms où les données publiques d'authentification sont conservées.

Ainsi, prenant en considération les problèmes exposés dans la Section 10.1 de la Partie III et dans la Section 11.1. de cette partie, ce travail vise à proposer un ensemble modulaire de mécanismes dans le but de permettre l'établissement d'une telle infrastructure. Cela est concrétisé à travers :

- la réorganisation du système global de façon modulaire,
- la définition d'une méthode de description de politiques de certification,
- la définition d'un mécanisme d'évaluation de chemins de certification basés sur la description de politiques de certification mais aussi sur des critères locaux et sur des métriques de validation,
- l'introduction d'un nouvel élément, complémentaire aux architectures de gestion de clés, qui permettra la réalisation de toute l'approche.

La chapitre suivant porte sur la réorganisation de l'espace de distribution de clés publiques. Dans le Chapitre 13 nous présenterons une technique pour la description de politiques de certification. Dans les Chapitres 14 et 15 nous traiterons les questions liées à la vérification et à l'obtention de clés publiques.

CHAPITRE 12
REORGANISATION DU SYSTEME

Typiquement, le regroupement d'organisations et de systèmes en domaines permet de simplifier la gestion et la prise de décisions. Ce type de structuration est courant dans plusieurs organisations et systèmes existants ; par exemple, gestion nationale, régionale ou départementale. Cela s'applique également dans le contexte de la sécurité. Par exemple, l'application de mesures générales de sécurité telle que la sécurité au niveau physique, exige l'identification de limites dans lesquelles ces mesures communes doivent être appliquées. Ce type de regroupement est normalement appelé *domaine de sécurité* [IS10181-1].

Dans notre travail, nous sommes intéressés au segment spécifique de la sécurité qui traite de la gestion de clés publiques. La prolifération et l'expansion de systèmes utilisant des mécanismes de sécurité à clé publique rend très élevé le nombre total de clés. Cela rend nécessaire la partition de cet espace de clés en *domaines de gestion de clés*, où le terme domaine décrit l'extension du contrôle de gestion [FL93].

Le contrôle des domaines de gestion de clés est garanti par des *politiques de gestion de clés*. Une politique de gestion de clés est un ensemble de règles et de critères qui déterminent, restreignent et contrôlent la manipulation du matériel cryptographique disponible aux principaux d'un domaine. Un domaine de gestion de clés constitue donc le champ de juridiction d'une politique de gestion de clés.

Nous visualiserons ainsi l'espace de distribution de clés ; bien que cette réorganisation ne soit pas nécessairement exigée pour la réalisation de chacun des mécanismes qui composent notre approche. Dans la suite de ce travail, nous utiliserons le terme **KMD** (**K**ey **M**anagement **D**omain) pour désigner les **Domaines de Gestion de Clés.** Chaque KMD est gouverné par une **Politique de Gestion de Clés** ou KMP (**K**ey **M**anagement **P**olicy) bien définie. Afin d'isoler des applications la complexité des procédures de manipulation de certificats et pour que chaque KMD puisse choisir les modèles de confiance et les services de noms les plus appropriés à ses besoins, un nouvel élément est proposé : le **Serveur de Données pour l'Authentification** ou DAS (**D**ata **A**uthentication **S**erver). L'exigence principale de notre approche est que celle-ci soit

conforme au cadre d'authentification X.509 et qu'elle permette la création d'une infrastructure commune pour les services d'authentification, car l'authentification est à la base de tous les services de sécurité.

Cette approche ne prescrit pas de restrictions en ce qui concerne la configuration d'un KMD. Les types et le nombre limite de modèles de confiance, de CA et d'utilisateur final, ainsi que les limites de contrôle de gestion d'un KMD ne sont pas imposés. Ces aspects dépendent des besoins et de la capacité de gestion de chaque domaine.

Cette façon d'aborder les structures de contrôle de gestion permet toute une gamme de rapports entre KMD. Les KMD peuvent être disjoints, se recouvrir ou être des sous-ensembles d'autres KMD. Le concept de sous-ensemble d'un KMD est utile pour la partition d'un grand domaine afin de partager les responsabilités de gestion. Pourtant, il doit être noté que ces sous-ensembles ne constituent pas forcement une organisation hiérarchique. Un sous-domaine doit être vu comme un domaine qui constitue un objet dans un autre domaine. Cela simplifie la vision et la gestion des domaines.

Le regroupement d'organisations et systèmes sous une même politique de gestion de clés permet la prise de décisions communes en ce qui concerne l'interopérabilité entre domaines. Ces décisions sont basées sur le niveau de sécurité commun qui doit être mis en vigueur. Ainsi, la politique de gestion de clés d'un domaine doit spécifier les règles d'interrelation avec d'autres domaines de gestion de clés. Les domaines de gestion de clés et les rapports entre ces domaines doivent refléter les politiques de sécurité des systèmes les constituant. En particulier, du point de vue de la certification, les KMD peuvent être interconnectés pour permettre l'authentification des principaux appartenant à des KMD différents. L'interconnexion entre KMD est réalisée à travers la certification entre CA appartenant à des différents KMD.

Fondamentalement, la KMP constitue la base pour l'évaluation de certificats dans un KMD. Celle-ci contient des informations concernant le stockage des données publiques d'authentification des entités qui constituent le KMD, ainsi que les critères de validation de certificats dans le KMD et des références concernant les KMD interconnectés. Il est important que la KMP et les informations de connaissance d'un KMD soient accessibles, notamment au DAS. Cela permet à celui-ci de détecter la violation des politiques de sécurité et de fournir ses services de façon correcte et fiable.

Dans le chapitre suivant, nous proposerons une méthode à travers laquelle les modèles de confiance pourront rendre explicite les règles et sémantiques de certification. A partir de celles-ci, chaque KMD pourra dériver leurs critères locaux de validation de chemins de certification, selon leur politique de sécurité. De même, chaque utilisateur pourra exprimer les métriques et le seuil souhaités pour l'obtention et validation de certificats. Notre approche vise ainsi à :

- répondre aux besoins des utilisateurs et de l'environnement ouvert et hétérogène où ces usages se situent,
- permettre que le montant d'information que chaque utilisateur doit maintenir localement protégé reste réduit,
- permettre que chaque KMD puisse adopter la structure de sécurité la plus appropriée à ses besoins, tout en permettant l'interopérabilité des mécanismes,

- permettre que les diverses politiques de certification et critères locaux d'évaluation puissent être décrits de façon à permettre leur interprétation et traitement automatiques par les outils de manipulation de certificats (que dans notre approche sont représentés par le DAS). La Figure 27 illustre un tel scénario.

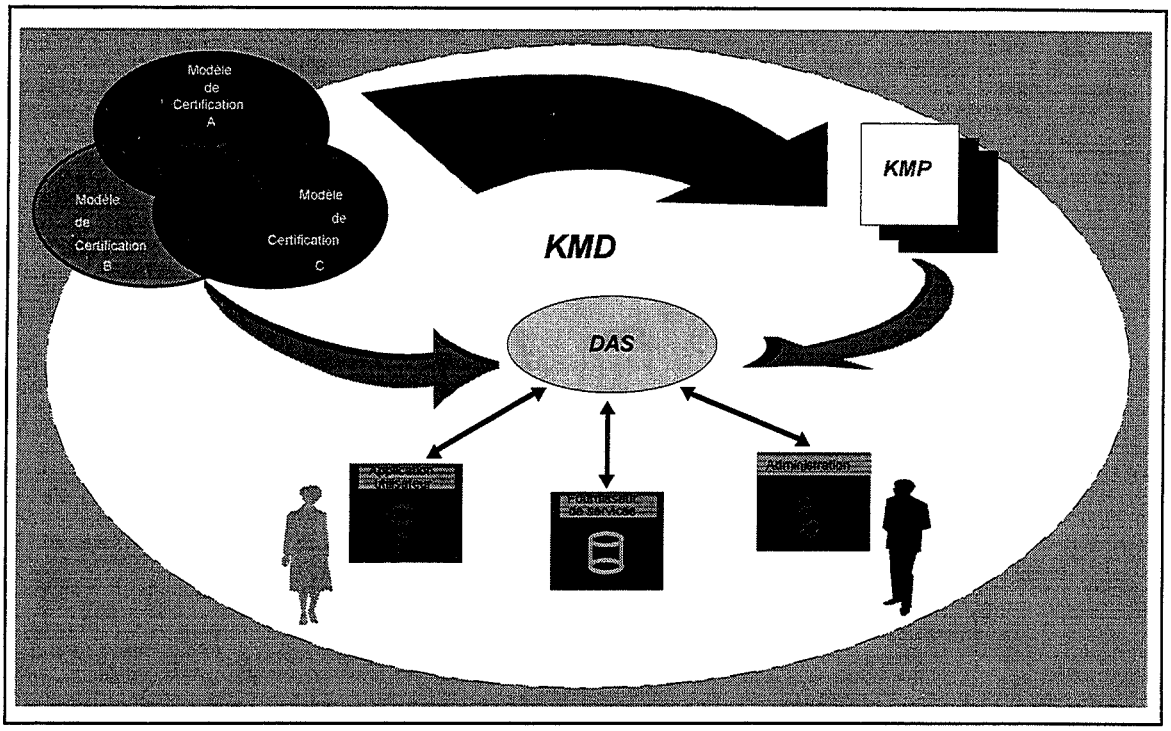

Figure 27. Structure d'un KMD

La suite de cette partie approfondit notre approche, à savoir l'obtention et la vérification de clés publiques dans un environnement ouvert à travers l'évolution du concept du DAS. Cette approche vise à proposer des solutions aux problèmes exposés dans 10.1 et 11.1., car ces problèmes constituent les obstacles à l'établissement d'une infrastructure commune pour l'authentification dans les systèmes ouverts.

CHAPITRE 13
DESCRIPTION DES POLITIQUES DE CERTIFICATION

Les politiques de sécurité consistent en un ensemble de règles qui gouvernent le comportement et l'accès à chaque composant d'un système. Les politiques de sécurité adoptées dans un système distribué doivent être déterminées par les besoins des entités et des organisations intégrantes du système. La portée de ces politiques dépend des fonctionnalités du système et de la granularité souhaitée.

Dans le contexte de la certification, l'objectif primaire de la politique de sécurité est d'assurer qu'un certificat ne sera émis qu'avec l'absolue évidence de l'identité du principal. D'autres objectifs sont de limiter les conséquences de l'action malicieuse des opérateurs de CA, et d'assurer que ces opérateurs peuvent être maintenus responsables de leurs actions et que l'on disposera des moyens nécessaires pour leur en faire rendre compte [Roe92].

Etant donné que les CA sont des entités de confiance, la politique de certification doit limiter la possibilité d'un abus de pouvoir. Cela se produit lorsqu'une entité se sert de son autorité pour réaliser une fonction non autorisée. Par exemple :

* une CA qui émet un certificat pour un membre d'une organisation hors de sa juridiction,
* une CA qui révoque un certificat émis par une autre autorité.

Un danger similaire est celui de l'abus de confiance. Ceci a lieu lorsqu'une entité se sert de la confiance qu'on lui a accordée pour réaliser une action malveillante. Par exemple :

* une CA qui utilise la clé privée d'un utilisateur pour contrefaire des signatures ou pour déchiffrer des informations confidentielles,
* une CA qui émet des certificats incorrects pour détourner un mécanisme de sécurité,
* une CA qui émet des listes de révocation incorrectes afin de provoquer un refus de service.

Le sous-ensemble de la politique de sécurité concernant les aspects mentionnés ci-dessus constitue la *politique de certification*, aussi connue comme *politique de distribution de clés publiques*. Elle doit préciser la topologie des CA et des utilisateurs, la définition de la juridiction des CA, les rapports entre CA et les rapports entre les CA et les utilisateurs, de manière à contrôler la distribution de ces ressources. En résumé, le rôle d'une politique de distribution de clés publiques est de :

- ◊ définir l'emplacement de clés publiques dans l'espace de noms,
- ◊ catégoriser les entités concernées dans l'espace de clés,
- ◊ préciser les rapports et les règles de certification entre ces entités.

L'implémentation de mécanismes de sécurité à grande échelle requiert une infrastructure d'authentification capable d'accommoder toute une gamme de politiques (pour utilisateurs et organisations) d'une façon précise et explicite. Une telle infrastructure doit permettre à chaque utilisateur d'accéder à ces politiques. Cela permet à chaque utilisateur (ou domaine) de définir les règles et les modalités à respecter lors de la vérification des certificats.

13.2. Objectifs

Dans un environnement hétérogene, l'infrastructure de certification doit aussi permettre l'authentification (et l'identification) d'entités qui peuvent être de diverses classes d'utilisateur final et de ressources tels que services et applications. Afin d'être applicables dans un tel environnement, les procédures d'authentification doivent disposer d'une infrastructure permettant aux systèmes de certification de publier leurs politiques de certification et de les rendre disponibles à grande echelle. Si les politiques et sémantiques de certification sont exposées de façon claire et bien définie, l'utilisateur final sera capable de prendre des décisions précises basés sur des certificats. De plus, comme nous l'avons déjà mentionné, si les schémas de certification peuvent être harmonisés d'une façon sûre, claire et bien définie, alors une infrastructure globale d'authentification pourra se développer sans le besoin d'un modèle de certification commun à tous les utilisateurs.

Notre expérience de X.509 nous a fait constater que la création de modèles de confiance pouvant regrouper des CA, des politiques et des sémantiques de certification hétérogènes exige la définition d'un *format commun* pour la description de telles politiques. A travers un tel format chaque modèle pourrait mettre à disposition des informations permettant d'identifier sans ambiguïté le rôle des entités et des certificats.

Or, il est important que ces politiques soient clairement établies et qu'elles soient visibles aux outils chargés de la récupération et de la vérification de clés, puisque ces procédures constituent le point de départ pour la prestation de services de sécurité. Cela devient judicieux lorsque des configurations hétérogènes sont mises en place. La possible harmonisation entre politiques doit aussi être décrite. Cela peut permettre la coexistence des différentes configurations et la transparence vis-à-vis des applications et des utilisateurs.

Puisque les certificats et les structures d'identification des entités sont définis par des structures de données, nous devons chercher une *représentation abstraite* pour la description de ces politiques car ce genre de représentation permet :

- que les types de données soient décrits indépendamment des structures et des restrictions de l'environnement d'implémentation,
- leur traitement automatique par des applications et la détection automatique de la violation de ces politiques.

Dans la suite de cette section nous allons donc developper une méthode de description de politiques de certification ayant pour but de remplir les besoins mentionés au cours de cette section.

13.3. Une Méthode de Description de Politiques de Certification

Après avoir identifié les déficiences du format de certificat X.509 et les besoins de description des politiques de certification, nous avons conçu une méthode permettant de rendre explicite les rapports de certification. Avant de décrire la méthode, nous présenterons les éléments du modèle de description et le langage choisi. Après la présentation de la méthode, nous étudierons son application vis-à-vis des modèles de certification existants. Dans le Chapitre 14, nous étudierons l'application de cette méthode pour l'évaluation de certificats et pour la définition de critères locaux d'évaluation de chemins de certification dans un domaine.

13.3.1. Eléments du Modèle de Description

La description d'un modèle de certification doit prendre en compte les types d'entités qui le constituent et la façon dont ce modèle est organisé. Ainsi, nous nous sommes intéressés à la description de trois types principaux d'objet : les **modèles** de certification, les **entités** qui constituent ces modèles et les **ressources** manipulées pour la formation de ces modèles.

Les Entités

Deux types principaux d'entité peuvent être reconnus dans le contexte de la certification : des entités dont la crédibilité est déterminée par des "autorités administratives", et donc, la crédibilité (ou la confiance) attribuée à des entités de ce type ont une extension qui couvre une ou plusieurs communautés. Ce type d'entités sera appelé, dans cette approche, *entités de confiance publique*. Les autorités de certification sont des exemples de ce type d'entité.

L'autre type d'entité est tel que la crédibilité n'est pas déterminée par des autorités administratives et donc, la crédibilité d'une entité de ce type ne peut être déterminée que pour elle-même ou pour d'autres entités qui le font de façon privée (ou personelle). Par conséquent, la crédibilité de ces entités n'est pas garantie par le modèle. Ce type d'entité sera appelé, dans cette approche, *entités de confiance privée*. Les utilisateurs finals sont des exemples de ce type d'entité.

Modèles Publics et Privés

En conséquence des types d'entités identifiés ci-dessus, d'autres termes peuvent être introduits: un modèle où la notion de confiance est déterminée (et/ou imposée) par des entités de confiance publique est appelé un *modèle public* ; tandis qu'un modèle où la notion de confiance n'est pas imposée par des moyens externes mais définie par chaque utilisateur est appelé un *modèle privé*.

Pour joindre un modèle public de confiance, une CA doit accepter certaines obligations [Roe92]. De façon générale, les obligations d'une CA entrent dans les catégories suivantes :

- l'obligation de ne pas faire d'affirmations fausses, i.e., une CA ne peut affirmer que des faits dignes de foi,
- l'obligation de contrôler ses affirmations avec soin, i.e., toute affirmation d'une CA doit être justifiée (ou justifiable),
- l'obligation de ne pas faire des affirmations ambiguës, i.e., le langage utilisé pour exprimer des affirmations (le format de certificat) doit associer une et seulement une signification à une affirmation faite. La sémantique associée à un certificat particulier est définie partiellement par le standard X.509 et partiellement par la politique de sécurité adoptée.

D'autres obligations varient selon le rôle et la position de la CA dans le modèle. En récompense de l'acceptation de ces obligations, une CA reçoit juridiction sur des affirmations faites sous une forme particulière. C'est-à-dire, le modèle public déclare que l'on doit croire aux affirmations faites par les CA sous la forme de certificats.

En ce qui concerne les modèles privés, dans de tels modèles la certification n'est pas réalisée par des entités globalement reconnues. Les structures de certification sont établies par la propre initiative de chaque individu. Chaque utilisateur détermine s'il doit accorder confiance à d'autres entités et le degré de confiance qu'il doit accorder à chacune des entités. Ainsi, les certificats émis ne sont pas endossés par une politique globalement reconnue. Comme exemples de ce type de modèles privés, nous pouvons citer RIPEM [Rio92], où les certificats sont signés par leurs propres propriétaires (*self-signed certificates*), et PGP [Zim93] où chaque utilisateur certifie les clés des utilisateurs auxquels il fait confiance.

Du point de vue de la fiabilité, les structures privées de certification auront une étendue limité, comparées aux structures publiques. Ces premières se présentent plutôt dans des environnements locaux et dans des scénarios où la confiance est issue d'un contact direct et interpersonnel. Cependant, ces modèles privés sont utiles au déploiement de services de sécurité lorsque les usagers ne nécessitent ou ne souhaitent pas avoir un endossement légal[29].

Du point de vue des politiques de certification, il doit être noté que les modèles de confiance peuvent définir divers types d'entité afin d'obtenir une granularité plus fine vis-à-vis des sémantiques d'authentification. Par conséquent, il est nécessaire de fournir des moyens pour différencier, de façon sûre, les plusieurs modèles de certification et types d'entité. Cela a l'avantage additionnel d'éviter des conflits entre des profils de certification et de permettre l'automatiser d'avantage les procédures de construction et de validation de chemins de certification.

---

[29]Lorsque les certificats servent à des fins de non-répudiation (démonstration à une tierce partie qu'une opération a été réalisée), le modèle de confiance utilisé par l'arbitre est le modèle public adopté comme règle de jugement, et non celui d'une des parties en litige. Si le modèle public déclare que l'on ne doit pas croire à un type particulier de certificat, l'arbitre ne l'acceptera pas. Par conséquent, les utilisateurs qui décident de faire confiance à un tel certificat, le font à leurs propres risques.

Les Ressources

Une politique de certification définit l'attribution et l'usage de certaines ressources entre les entités qui forment le modèle. En ce qui concerne les ressources traitées dans cette approche, nous identifions :

- des algorithmes et des clés de chiffrement,
- des certificats,
- des méthodes de révocation de certificats,
- des moyens d'identification des entités.

Ainsi, outre les modèles de confiance et les types d'entité qui les intègrent, il est nécessaire de pouvoir identifier sans ambiguïté :

- **Le niveau d'assurance et la juridiction accordés aux entités certificatrices**

  Partant du principe que l'attribution d'un certificat à une CA constitue une délégation de confiance, il semble approprié qu'une CA qui certifie une autre puisse aussi exprimer le degré de confiance accordé à la CA certifiée.

  Ainsi, pour chaque entité certificatrice, il doit être possible de déterminer le niveau d'assurance qui lui a été accordé, quels types d'entité elle est autorisée à certifier et les types de certificat qu'elle est autorisée à emettre.

- **La sémantique liée à chaque certificat**

  Il doit être possible de déterminer si un certain certificat permet de spécifier des politiques, s'il permet de déléguer autorité, s'il désigne la connectivité entre deux domaines de sécurité.

- **L'éventuel rapport entre des modèles de certification**

  Il s'agit des critères d'interconnexion avec d'autres modèles, à travers quels types d'entité certificatrice et quels types de certificat.

- **Des aspects de gestion de certificat**

  Lorsqu'une entité est autorisée à certifier, il doit être possible de déterminer :

  - l'algorithme et la taille de clés qu'une entité est autorisée à certifier,
  - la durée de validité qui doit être attribuée aux certificats émis,
  - la méthode et la fréquence qu'une entité doit respecter pour annoncer la révocation de certificats.

- **Des règles d'usage d'un certificat**

  Les aspects ci-dessus définissent implicitement les règles d'usage des certificats attribués à des entités certificatrices. Au niveau des utilisateurs finals, il est également nécessaire de pouvoir déterminer les règles d'usage associées aux certificats qui leur sont attribués.

Puisque l'attribution de ressources est strictement liée à l'attribution d'un certificat, la description de leur usage doit être associée au certificat correspondant. Cela permet la différence d'usage

d'une même ressource par diverses entités d'un même type, qu'une même entité puisse jouer des rôles différents et d'intégrer plusieurs modèles.

13.3.2. Choix du Langage

Dans le monde OSI, une méthode de représentation abstraite a été adoptée, à savoir, l'**ASN.1** (**A**bstract **S**yntax **N**otation One). Cette notation constitue un langage formel pour la description de types de données permettant un degré de complexité arbitraire de types décrits. Ce langage est indépendant des mécanismes utilisés pour la production de représentations concrètes. En particulier, un ensemble de règles de codage de base, le BER (Basic Encoding Rules) [X.209], est défini pour l'ASN.1 de façon que les données puissent être transmises sans ambiguïté. Néanmoins, d'autres règles de codage peuvent être définies pour la représentation des instances concrètes de types de données définis en ASN.1 [X.208].

L'ASN.1 définit un ensemble de types primitifs et fournit des mécanismes qui permettent la construction des nouveaux éléments. Cela permet la définition de nouveaux types de données tels que ces types soient reconnaissables de manière unique dans une application. Les *macros* ASN.1 constituent des mécanismes qui permettent non seulement l'expansion de la grammaire de la notation ASN.1, mais aussi la formulation de nouvelles sémantiques. Ce mécanisme réécrit les règles de grammaire du langage ASN.1.

Dans le contexte des politiques de certification, nous n'avons pas besoin d'une base mathématique, mais d'un **moyen de description** de règles de certification qui définissent les limites d'action de chaque principal du modèle. De plus, les données de base sur lesquelles les procédures de certification sont basées ont déjà leur représentation en ASN.1. Il semble donc logique de se servir de la puissance et du formalisme compris dans cette notation pour décrire les politiques de distribution de clés publiques.

13.3.3. Méthode de Description

De manière générale, l'attribution d'un certificat implique l'acceptation de la politique définie par l'entité certificatrice. Par conséquent, le modèle de description est tel que chaque entité certificatrice d'un modèle de certification spécifie les règles de certification applicables aux entités certifiées. Ainsi, au moment de l'émission d'un certificat, l'entité certificatrice émet également la politique de certification qui doit être respectée par l'entité certifiée. La Figure 28 illustre le modèle d'attribution de politiques.

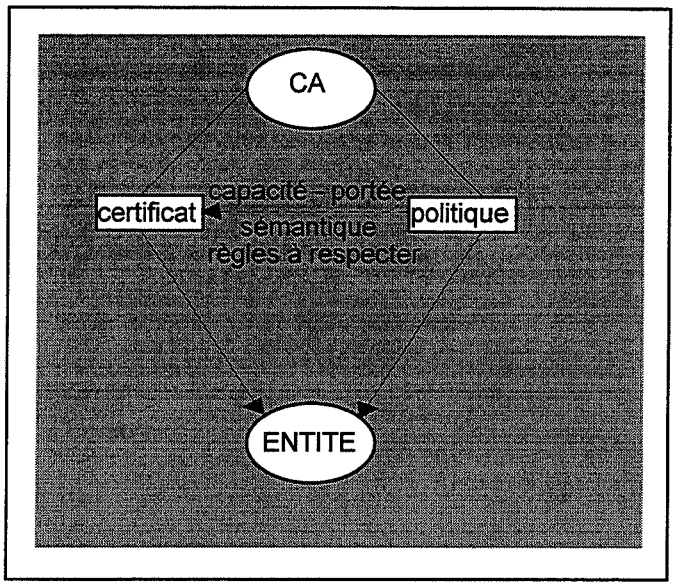

Figure 28. Attribution de Politique de Certification

A partir d'une telle description de politiques, chaque domaine de sécurité pourra définir ses propres modèles privés et/ou critères locaux d'évaluation, tels qu'on les décrira dans la Section 13.4.1. La suite de cette section présente les composants de notre méthode de description de politiques de certification et son emplacement vis-à-vis du cadre X.509.

Identification de modèles de certification

L'identification de modèles de certification peut être réalisée à travers l'attribution des *identificateurs d'objet*. Cela fournit un moyen formel et normalisé pour la distinction entre modèles de certification.

```
CertificationModel ::= OBJECT IDENTIFIER
```

Identification de types d'entité

De même, la distinction entre catégories de CA ainsi que l'identification de types d'utilisateur peut être réalisée à travers l'attribution des identificateurs d'objet *subordonnés* à celui attribué au modèle. Ainsi, chaque classe d'entité est associée à un identificateur d'objet unique. Cela permet d'identifier de façon globale chaque catégorie de CA et d'utilisateur et de determiner le modèle auquel une entité appartient.

```
EntityType ::= OBJECT IDENTIFIER
```

Sémantique des certificats

Lorsqu'une entité certifie une autre, elle doit spécifier la sémantique associée au certificat émis. La structure suivante permet d'identifier la sémantique associée à un certificat :

```
CertificateSemantic ::= ENUMERATED {
        indicates_a_policy_object (1),
        allows_authority_delegation (2),
        indicates_authority_delegation (3),
```

```
                indicates_a_reverse_certificate (4),
                indicates_a_cross_certificate (5),
                certification_disallowed (6)
        }
```

Chaque valeur a la signification suivante :

(1) *indicates_a_policy_object* indique que le certificat émis autorise le sujet à établir des politiques (la délégation d'autorité est implicite),

(2) *allows_authority_delegation* indique que le certificat émis autorise le sujet à déléguer autorité,

(3) *indicates_authority_delegation* indique que le certificat émis exprime une délégation d'autorité, mais n'autorise pas à déléguer autorité,

(4) *indicates_a_reverse_certificate* indique un certificat inverse, i.e., émis pour faciliter la validation de chemins de certification,

(5) *is_a_cross_certificate* indique que le certificat émis exprime l'interfonctionnement entre deux domaines de gestion ou de politiques différentes,

(6) *certification_disallowed* indique que le certificat émis n'autorise pas le sujet à émettre des certificats. Cela élimine le risque d'attaque au modèle par l'action malveillante d'un utilisateur qui essaie de se faire passer pour une CA, par le biais du format unique des certificats.

Niveau d'assurance

Lorsqu'un certificat exprime délégation d'autorité, l'entité certificatrice doit expliciter le niveau d'assurance associé à l'entité certifiée. La structure suivante permet d'identifier le niveau d'assurance associé à l'entité :

```
AssuranceLevel ::= ENUMERATED {
                        high (0), medium (1), low (2), unknown (3) }
```

En particulier, la valeur *unknown* permet de designer l'interfonctionnement avec des domaines dont le niveau d'assurance n'est pas garanti, ainsi que l'harmonisation entre deux modèles de certification sans que cela implique une convergence au niveau des politiques de sécurité et sans engagement quant à la notion de confiance.

Règles de certification

Pour chaque entité autorisée à certifier, la politique de certification doit spécifier les règles que celle-ci doit respecter pour la certification. Ces règles peuvent être décrites à travers la structure ASN.1 suivante :

```
CertificationRules ::= SEQUENCE {
    jurisdiction                [0] SET OF {
                                    entities    NamingRules,
                                    semantic    CertificateSemantic},
    revocation_frequency        [1] INTEGER,
                                    -- in days
    crl_responder_address       [2] PresentationAddress OPTIONAL,
```

```
                                        – absent if not used
    certificate_validity                [3] INTEGER,
                                        – in days
    key_restriction                     [4] SEQUENCE OF AlgorithmIdentifier,
                                        – parameters may be used to
                                        – specify the minimum key size
    distinct_keys_for_different_issuers [5] BOOLEAN OPTIONAL,
                                        – absent if the certified key is
                                        – one of the master keys
    how_many_levels_up_to_the_leaf      [6] INTEGER  OPTIONAL,
                                        – absent if undefined
}
```

Chaque élément a la signification suivante :

- *jurisdiction* : détermine la juridiction accordée à l'entité certifiée et la sémantique qui doit être associée aux certificats émis pour chaque type d'entité ou portion de l'espace de noms déléguée.
- *revocation_frequency* : indique la fréquence avec laquelle l'entité doit mettre à jour sa liste de révocation,
- *crl_responder_address* : indique l'adresse du répondeur de liste de révocation de l'entité certifiée. Cet élément est absent si l'entité ne possède pas un répondeur de liste de révocation,
- *certificate_validity* : indique la validité que l'entité doit attribuer aux certificats émis,
- *key_restriction* : indique les algorithmes et la taille minimale de clés que l'entité est autorisée à certifier,
- *distinct_keys_for_different_issuers* : à travers cet élément, l'entité certificatrice indique si l'entité certifiée est autorisée à avoir des certificats émis par d'autres autorités, pour la clé publique certifiée pour cette première. Cet élément est absent s'il s'agit d'une clé maître du modèle,
- *how_many_levels_up_to_the_leaf* : à travers cet élément, l'entité certificatrice peut indiquer le nombre maximum de niveaux d'entités certificatrices de l'entité certifiée jusqu'à la feuille de l'arbre de certification. Cet élément est absent si cet aspect n'est pas défini par la politique de l'entité certificatrice.

Identification de juridiction

La juridiction d'une entité certificatrice d'un modèle est identifiée en termes de règles de nommage permettant de spécifier les entités que celle-ci est autorisée à certifier soit en termes de portions de l'espace de noms, soit en termes d'identification de types d'entité. Afin de permettre toutes sortes de définitions, ces règles de nommage peuvent être décrites à travers la structure ASN.1 suivante :

```
NamingRules ::= CHOICE {
    item [0]  NamingRulesItem,
    and  [1]  SET OF NamingRules,
    or   [2]  SET OF NamingRules,
```

```
    not  [3]   NamingRules
}

NamingRulesItem ::= CHOICE {
    subordination   [0] NULL,
    present         [1] AttributeType,
    equality        [2] AttributeValueAssertion,
    substrings      [3] SEQUENCE {
                            type        AttributeType,
                            strings     SEQUENCE OF CHOICE {
                                            initial     [0] AttributeValue,
                                            any         [1] AttributeValue,
                                            final       [2] AttributeValue }},
    objectClass     [4] OBJECT IDENTIFIER
    entityType      [5] OBJECT IDENTIFIER
}
```

Nous avons défini l'élément *NamingRules* sous forme de filtre pour admettre la combinaison de diverses règles de nommage au moyen des opérateurs logiques. Cela permet l'élaboration de définitions de complexité arbitraire.

Finalement, la description de la politique de certification de chaque entité d'un modèle de confiance prend la forme suivante :

```
PolicyDescription ::= SEQUENCE {
        has_trusted_key             [0] BOOLEAN DEFAULT FALSE,
        entityType                  [1] EntityType,
        certificateSemantic         [2] CertificateSemantic,
        assuranceLevel              [3] AssuranceLevel OPTIONAL,
        rulesforCertification       [4] CertificationRules OPTIONAL
                        -- absent if the subject is not allowed to
                        -- certify or if it is a cross or reverse certificate
}
```

Chaque élément a la signification suivante :

- *has_trusted_key* : indique si l'entité certifiée possède la clé maître du modèle de certification en question,
- *entityType* : identifie l'entité certifiée vis-à-vis du modèle de certification,
- *certificateSemantic* : identifie la sémantique liée au certificat émis,
- *assuranceLevel* : indique le niveau d'assurance accordé à l'entité certifiée. Cet élément est absent si l'entité certifiée n'est pas autorisée à certifier ou s'il s'agit d'un certificat inverse.
- *rulesforCertification* : détermine les règles que l'entité certifiée doit respecter lorsqu'elle certifie d'autres entités. Cet élément est absent si l'entité certifiée n'est pas autorisée à certifier ou s'il s'agit d'un certificat croisé ou inverse.

13.3.4. Emplacement

Pour que la description de la politique soit opérationnelle du point de vue de l'authentification, en plus d'être liée au certificat, elle doit être largement disponible. La description de politique doit donc être placée dans l'entrée des entités certifiées dans le service de noms. Il paraît ainsi naturel de la placer dans le certificat. Cela permettrait :

- que l'intégrité de cette valeur soit cryptographiquement protégée,
- qu'une partie de la procédure d'évaluation des chemins de certification soit effectuée directement à partir du chemin lui-même,
- que l'altération de ces valeurs soit immédiatement reflétée,
- que la CA certificatrice puisse changer cette valeur lorsque nécessaire. Par exemple, lorsque l'évolution de la technologie exige une augmentation de la taille de clés, les CA ne supportant pas la génération de clés conforme aux exigences pourront avoir leur certificat substitué par un autre spécifiant un degré d'assurance réduit.

En contrepartie, l'augmentation de la taille de certificats peut nuire à l'utilisation et à l'administration du système. Les utilisateurs (et les applications) seraient amenés à récupérer la description de politique même lorsqu'ils s'intéressent seulement au certificat. Ainsi, il peut être préférable de définir un nouvel attribut "*policy*", spécifié comme obligatoire dans les entrées de classe "*certificationAuthority*" et "*strongAuthenticationUser*". Pour assurer l'intégrité de cet ensemble d'informations, il doit être signé par l'entité émettrice.

Accessoirement, on peut envisager d'y inclure des *attributs authentifiés*, par exemple des adresses de messagerie ou des identificateurs de contrôle d'accès. Ainsi, cet attribut aurait la définition suivante :

```
Policy ::= SIGNED SEQUENCE {
    version              [0] INTEGER { 1994(0) } DEFAULT 1994,
    policyDescription    [1] PolicyDescription,
    signedBy             [2] DistinguishedName,
    subject              [3] DistinguishedName,
    othersAttributes     [4] SET OF AttributeValueAssertion OPTIONAL,
    relates_to           [5] SEQUENCE OF SerialNumber}
```

La signification de chaque élément est montrée ci-dessus :

- *version* : indique la version du format de description de politiques,
- *policyDescription* : décrit la politique de certification spécifiée pour l'entité certificatrice,
- *signedBy* : indique le nom d'annuaire de l'entité responsable de l'émission de la politique et doit, évidemment, être le même que celui spécifié dans le champ "*issuer*" du certificat correspondant,
- *subject* : correspond au nom d'annuaire de l'entité certifiée et donc soumise à la politique en question ; le nom doit être le même que celui spécifié dans le champ "*subject*" du certificat correspondant,

- *othersAttributes* : sert à désigner d'autres identificateurs associés à l'entité certifiée vis-à-vis des services sécurisés, par exemple, l'adresse présentation d'un serveur, l'adresse postale ou de messagerie d'un utilisateur, des identificateurs de contrôle d'accès.

  Cet élément peut également servir à délimiter le champ d'action du certificat émis, i.e., champ de validité des signatures produites au moyen du composant privé correspondant. Ceci peut être réalisé en utilisant l'attribut "*Description*". Par exemple,

  Description = "Ce certificat n'est valide qu'en territoire national."

  ou,

  Description = "Ce certificat n'autorise pas à signer quoi que ce soit au nom de la Société W."

- *relates_to* : indique les certificats auxquels cette politique s'applique.

La mise en place de cette méthode demande la redéfinition des classes d'objet *strongAuthenticationUser* et *certificationAuthority*, définies dans la Recommandation X.521 [X.521], de façon à comporter ce nouvel attribut. La nouvelle définition de ces classes d'objet serait la suivante :

```
strongAuthenticationUser   OBJECT-CLASS
        SUBCLASS OF     top
        MUST CONTAIN { userCertificate, policy }
::= new-oid-to-be-defined

certificationAuthority   OBJECT-CLASS
        SUBCLASS OF     top
        MUST CONTAIN  { cACertificate, certificateRevocationList, policy }
        MAY CONTAIN   { crossCertificatePair }
::= new-oid-to-be-defined

policy        ATTRIBUTE
              WITH ATTRIBUTE-SYNTAX Policy
::= oid-to-be-defined
```

Alternativement, la description de la politique de certification peut être placée dans un certificat du type PKCS #6, sous forme d'un attribut authentifié. Cela permet d'extraire le certificat X.509 sans intervention cryptographique.

Les applications chargées de la construction et de la validation de chemins de certification ne nécessitent pas de récupérer cet attribut à chaque fois qu'un certificat doit être vérifié, car sa validité est strictement liée à celle du certificat en question. Ainsi, cet attribut peut être conservé localement jusqu'à ce que sa date d'expiration soit atteinte.

13.4. Etude de cas et Exemples

La réalisation de cette méthode demande la définition d'identificateurs d'objet pour les modèles existants. Nous pourrions définir un sous-arbre d'identificateurs d'objet au-dessous de l'identificateur défini pour le cadre d'authentification X509. Ce sous-arbre s'appellerait

certificationModel et aurait deux branches, à savoir *public* et *private*, permettant de distinguer entre les modèles publics et privés.

Cela nous offrirait deux sous-arbres au-dessous desquels nous pourrions enregistrer des modèles selon leur type. Au-dessous de chaque modèle nous enregistrerions les types d'entité. En prenant comme exemple les modèles déjà décrits dans ce travail, nous aurions le scénario suivant :

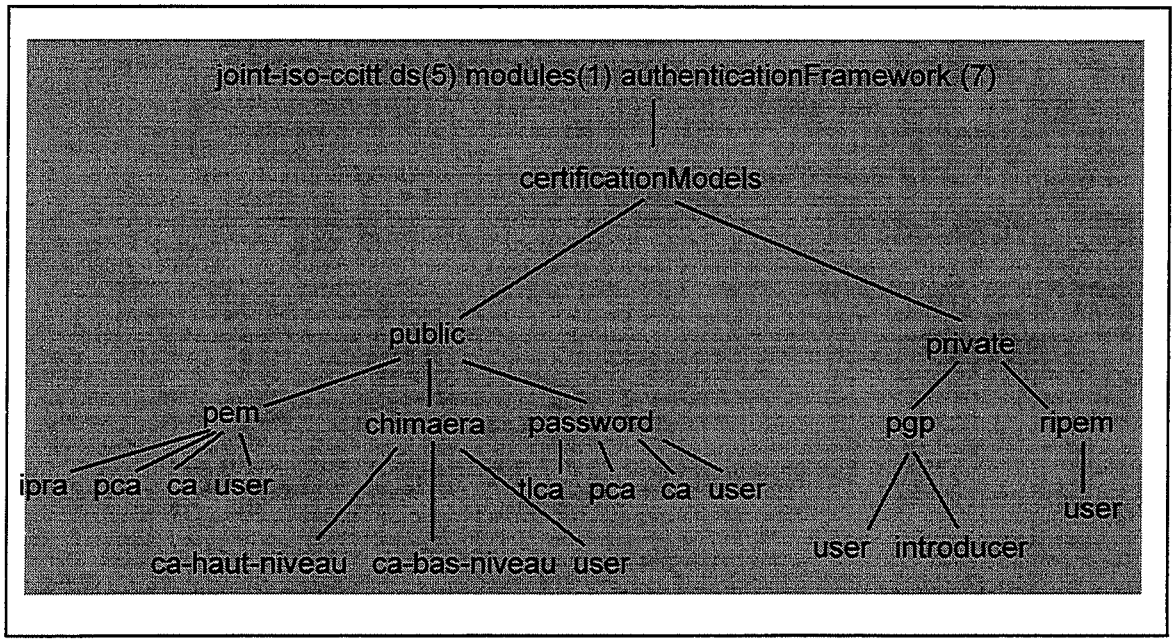

Figure 29. Attribution d'Identificateurs d'Objet aux Modèles de Certification

Si une plus grande granularité est souhaitée, chaque modèle peut créer des identificateurs pour d'autres types d'entités. Par exemple, dans PEM d'autres identificateurs peuvent être désignés pour faire la distinction entre les CA Organisationnelles, les CA Résidentielles et les CA Persona. De même, il peut être souhaitable de désigner un type spécifique de CA pour la certification de processus d'application, serveurs ou services.

Cette technique permet la création d'identificateurs pour les modèles à l'échelon international. Elle permet aussi que chaque type d'entité identifie le modèle auquel il appartient. D'autres identificateurs peuvent être définis par des organismes nationaux ou des organisations privées. Cela implique que certains organismes soit chargés de désigner des identificateurs de telle sorte que chacun d'eux soit distinct de tous les autres identificateurs affectés.

13.4.1. Modèles Publics

Pour illustrer l'application de la méthode dans le cas des modèles publics, nous prenons comme exemple la partie du modèle Password concernant le consortium français (voir 7.2.1.1 dans la Partie III). Dans un tel contexte, lors de la mise en place de la structure du projet, chaque TLCA génère sa paire de clés, son certificat et sa politique qui doit être approuvée par l'ensemble du projet. La TLCA française aurait une description de politique suivante :

Exemple 1

```
policy :
    policyDescription :
      has_trusted_key :              TRUE
      entityType :                   oid-model-public-password-tlca
      certificateSemantic :          indicates_a_policy_object
      assuranceLevel :               high
      rulesForCertification :
                  juridisdiction :
                        namingRules :      entityType :    oid-model-public-password-pca
                                           and
                                           equality : C=FR
                        semantic :         indicates_a_policy_object
                        namingRules :      entityType :    oid-model-public-password-pca
                                           and
                                           equality : C=DE ou equality : C=GB
                        semantic :         indicates_a_cross_certificate
                  revocation_frequency :   30
                  certificate_validity :   365
                  key_restriction :        algorithm       rsa
                                           parameters      1024
    signedBy :    <C=FR; O=TLCA-FR>
    subject :     <C=FR; O=TLCA-FR>
    relates_to :  1
```

L'élément *has_trusted_key* indique que le certificat de < C=FR; O=TLCA-FR > contient la clé maître de l'arbre de certification du consortium français. La description de sa juridiction montre qu'elle est autorisée à certifier des PCA du modèle Password avec les restrictions suivantes :

- la certification d'une PCA française indique que cette dernière peut établir des politiques,
- la certification d'une PCA anglaise ou allemande indique l'interfonctionnement avec ces deux pays,
- la certification de toute autre entité n'est pas autorisée.

Par conséquent, lorsque la TLCA française certifie une PCA française d'assurance moyenne (disons <C=FR; O=INRIA; OU=PCA>), elle doit définir la politique que cette dernière doit respecter :

Exemple 2

```
policy :
    policyDescription :
      entityType :                   oid-model-public-password-pca
      certificateSemantic :          indicates_a_policy_object
      assuranceLevel :               medium
      rulesForCertification :
                  juridisdiction :
                        namingRules :      entityType : oid-model-public-password-ca
                                           and          equality : C=FR
                                           and          present : O
                                           not          present : OU
                        semantic :         allows_authority_delegation
                  revocation_frequency :   30
                  certificate_validity :   365
                  key_restriction :        algorithm       rsa
```

```
                                        parameters    512
  signedBy :    <C=FR; O=TLCA-FR>
  subject :     <C=FR; O=INRIA; OU=PCA>
  relates_to :  2
```

De même, lorsque la TLCA française certifie une PCA allemande de faible assurance (disons < C=DE; O=GMD >, certificat numéro 3), elle doit définir la sémantique liée à ce certificat :

Exemple 3

```
policy :
   policyDescription :
     entityType :                 oid-model-public-password-pca
     certificateSemantic :        indicates_a_cross_certificate
     assuranceLevel :             low
   signedBy :    <C=FR; O=TLCA-FR>
   subject :     <C=DE; O=GMD>
   relates_to :  3
```

L'exemple 3 montre que le certificat numéro 3 émis pour < C=DE; O=GMD > ne lui concède aucune juridiction mais seulement exprime l'interfonctionnement entre deux communautés.

Maintenant, supposons que la PCA française certifie une CA opérée par la Societé Z, ayant le nom <C=FR; O=Societé Z>. Cette CA doit être autorisée à certifier d'autres CA subordonnées qui ne sont pas autorisées à certifier d'autres CA, et des utilisateurs appartenant à la Société X ou à la Société Y. La politique correspondante aurait le format suivant :

Exemple 4

```
policy :
   policyDescription :
     entityType :                 oid-model-public-password-ca
     certificateSemantic :        allows_authority_delegation
     assuranceLevel :             medium
     rulesForCertification :
            juridisdiction :
                  namingRules :      entityType : oid-model-public-password-ca
                                     and        equality : C=FR
                                     and        equality : O=Z
                                     and        present : OU
                  semantic :         indicates_authority_delegation
                  namingRules :      entityType : oid-model-public-password-user
                                     and        equality : C=FR
                                     and        equality : O=X
                  semantic :         certification_disallowed
                  namingRules :      entityType : oid-model-public-password-user
                                     and        equality : C=FR
                                     and        equality : O=Y
                  semantic :         certification_disallowed
            revocation_frequency :   30
            certificate_validity :   365
            key_restriction :        algorithm     rsa
                                     parameters    512
   signedBy :    <C=FR; O=INRIA; OU=PCA>
   subject :     <C=FR; O=Societé Z>
   relates_to :  4
```

Finalement, imaginons que la CA de la Société Z certifie un utilisateur de la Société X qui possède deux paires de clés dont une plus longue et donc de plus forte assurance. Cet utilisateur souhaite pouvoir s'authentifier non seulement utilisant son nom d'annuaire < C=FR;O=Societé Z; CN="José da Silva" >, mais aussi ses adresses de messagerie (Jose.Silva@sophia.z.fr et Jose.Silva@bahia.z.br). La politique que la CA émettrait pour cet utilisateur aurait donc la forme suivante :

Exemple 5

```
policy :
    policyDescription :
      entityType :                  oid-model-public-password-user
      certificateSemantic :         certification_disallowed
    signedBy :                      <C=FR; O=Societé Z>
    subject :                       <C=FR; O=Societé Z; CN=José da Silva>
    authentified_attributs :
      rfc822 :          Jose.Silva@sophia.Z.fr, Jose.Silva@bahia.Z.br
      description :     Please use certificat 6 for top secret messages
    relates_to :        5, 6
```

13.4.2. Modèles Privés

Dans des modèles privés, les structures de certification ne dépendent pas d'un endossement externe ou institutionnel mais sont créées par chaque utilisateur de sa propre initiative. Cependant, les utilisateurs peuvent s'échanger des certificats pour augmenter le nombre de clés dans leurs environnements locaux de manière à étendre la population d'utilisateurs avec lesquels un canal sécurisé peut être établi.

La méthode peut alors servir comme format commun pour l'échange de certificats entre utilisateurs. L'intention est de permettre le transfert d'informations additionnelles indiquant le jugement de l'expéditeur vis-à-vis d'un certificat ou de son propriétaire ainsi que d'autres identificateurs associés à ce dernier.

En analysant les éléments de la méthode de description dans le contexte des modèles privés, nous identifions les semantiques et differences suivantes :

- *entityType* - identifie le modèle privé et le type d'entité en question,
- *assuranceLevel* - identifie le degré de confiance accordé au certificat en question,
- *certificateSemantic* et *certificationRules* - ces éléments peuvent ne pas avoir de sens pour des modèles privés "plats", où aucune hiérarchie de certification (ou délégation d'autorité) n'est établie. Par exemple, dans l'approche RIPEM chaque certificat est signé par son propriétaire (*self-signed certificate*).

  En contrepartie, dans des modèles privés où la notion de délégation d'autorité existe (PGP, par exemple), ces éléments peuvent servir à expliciter le rôle accordé à chaque introducteur.

  Dans les deux cas nous pouvons étendre l'élément *certificateSemantic* avec d'autres valeurs appropriées aux modèles privés.
- La signature de *policyDescription* peut être calculée sur les éléments de description *plus le certificat* envoyé. Cela offre au destinataire la possibilité de vérifier l'authenticité et l'intégrité de tout l'ensemble, en supossant que celui-ci ait obtenu, au préalable, la clé publique de l'expéditeur.

A titre d'exemple, dans le cas du modèle RIPEM, supposons qu'un utilisateur A envoie à B deux certficats : le certificat de C que A a obtenu directement de C, et le certificat de D que A a obtenu d'une source inconnue. La description associée au certificat de C aurait la forme suivante :

Exemple 6

```
policy :
    policyDescription :
      entityType :                  oid-model-private-ripem-user
      assuranceLevel :              high
    signedBy :                      <C=FR; O=Organisation X; CN=A>
    subject :                       <C=FR; O=Organisation Y; CN=C>
    authentified_attributs :
      rfc822 :                      C@domain1.fr, C@domain2.com
      advices :                     " Envoie-lui ton CV. Il est sympa et sa boite paye bien. "
    relates_to :                    7
```

La description associée au certificat de D aurait la forme suivante :

Exemple 7

```
policy :
    policyDescription :
      entityType :                  oid-model-private-ripem-user
      assuranceLevel :              unknown
    signedBy :                      <C=FR; O=Organisation X; CN=A>
    subject :                       <C=FR; O=Organisation Z; CN=D>
    authentified_attributs :
      postalAddress :               "45, Rue Bellevue - Nice - 06123"
      telephoneNumber :             "33 12 34 56 78"
      rfc822 :                      D@domain1.gov, D@domain2.edu
    relates_to :                    8
```

Dans le cas du modèle PGP, un utilisateur W certifie les clés d'autres utilisateurs, obtenues par des moyens sûrs et les stocke dans des BBS. Ultérieurement, d'autres utilisateurs qui considèrent W comme introducteur récupèrent ces clés publiques et les conservent dans leur environnement local.

Chaque utilisateur peut utiliser notre méthode pour définir le rôle de ses introducteurs. Imaginons qu'un utilisateur X considère W comme introducteur mais avec des restrictions: il est intéressé par des clés certifiées par des introducteurs de W et par des clés certifiées directement par W. De plus ces clés doivent appartenir à des utilisateurs aux Etats Unis et être des clés du type RSA ou ElGamal. L'exemple 8 montre la description de la politique que l'utilisateur X associe à l'utilisateur W.

Exemple 8

```
policy :
    policyDescription :
      entityType :                  oid-model-private-pgp-introducer
      assuranceLevel :              medium
      rulesForCertification :
                juridisdiction :
                        namingRules :       entityType : oid-model-private-pgp-introducer
                                            and          equality :  C=US
                        semantic :          indicates_authority_delegation
                        namingRules :       entityType : oid-model-private-pgp-user
                                            and          equality : C=US
                        semantic :          certification_disallowed
                        key_restriction :   algorithm :    rsa
                                            algorithm :    elGamal
    signedBy :      <C=FR; O=Organisation; CN=X>
    subject :       <C=US; O=Societé Z; CN=W>
    relates_to :    9
```

Au moment où l'utilisateur récupère des clés publiques à partir des BBS ou d'autres moyens, son logiciel PGP peut utiliser ces descriptions pour sélectionner les clés conformes au souhait de l'utilisateur.

13.4.3. Interopérabilité entre Modèles Hétérogènes

Du fait que dans des modèles privés les critères de confiance et de certification sont déterminés par chaque utilisateur, ceci peut certifier la clé de n'importe quelle entité pour être capable d'établir un canal sécurisé avec cette dernière.

Par exemple, un utilisateur PGP peut intégrer totalement ou partiellement la hiérarchie de certification PEM dans son modèle privé simplement en déclarant l'une des CA PEM comme son introducteur. Cela permet que cet utilisateur puisse authentifier tout utilisateur PEM et même lui envoyer des données secrètes sans besoin d'intégrer ce modèle public.

A l'inverse, avec les moyens disponibles actuellement, il n'est pas possible pour un utilisateur d'un modèle public d'établir un canal sécurisé avec un utilisateur qui n'intègre pas le même modèle.

Puisque notre méthode rend explicite la sémantique des certificats et identifie globalement les types d'entités de chaque modèle, elle offre la possibilité d'interopérabilité entre modèles hétérogènes. Supposons, par exemple, que la communauté Internet souhaite offrir à ses utilisateurs la possibilité de créer des canaux sécurisés avec des utilisateurs intégrant le modèle PGP. Pour cela, il suffirait que l'IPRA désigne une PCA (par exemple, < C=US; O=PCA-PGP >) ayant un niveau d'assurance *inconnu*, dont le rôle serait de certifier des utilisateurs PGP considérés comme introducteurs.

Par l'attribution d'un niveau d'assurance inconnu, le modèle PEM déclare n'accorder aucun capital de confiance à cette PCA et n'être pas responsable des actes commis dans ce sous-arbre. La politique que l'IPRA émettrait pour la PCA en question aurait la forme suivante :

Exemple 9

```
policy :
    policyDescription :
      entityType :                  oid-model-public-pem-pca
      certificateSemantic :         indicates_a_policy_object
      assuranceLevel :              unknown
      rulesForCertification :
                  juridisdiction :
                        namingRules :       entityType : oid-model-private-pgp-introducer
                        semantic :          allows_authority_delegation
                  revocation_frequency :    30
                  certificate_validity :    365
                  key_restriction :         algorithm       rsa
                                            parameters      512
    signedBy :    <L=Internet; O=IPRA>
    subject :     <C=US; O=PCA-PGP>
    relates_to :  10
```

Nous pouvons aussi envisager que la communauté PEM souhaite interopérer avec la communauté Password. Dans ce cas, l'IPRA pourrait émettre un certificat pour une ou plusieurs PCA Password ayant la sémantique d'un certificat croisé. Ce certificat contiendrait, en fait, une clé déjà certifiée par la TLCA Password responsable de la PCA en question et n'accorderait aucune juridiction à cette PCA mais indiquerait seulement la possibilité d'interopérabilité entre ces deux communautés. Le niveau d'assurance accordé à la PCA dépend de son évaluation par l'IPRA et du

degré d'harmonisation souhaité entre les deux communautés. L'exemple 10 illustre un tel scénario.

Exemple 10

```
policy :
    policyDescription :
      entityType :                  oid-model-public-password-pca
      certificateSemantic :         indicates_a_cross_certificate
      assuranceLevel :              medium
    signedBy :    <Loc=Internet; CN=IPRA>
    subject :     <C=FR; O=INRIA; OU=PCA>
    relates_to :  11
```

13.5. Considérations

Outre la résolution des problèmes ouverts du cadre X.509 décrits dans cette Partie et dans la Partie précedante, il est nécessaire d'étendre l'infrastructure de validation de clés publiques pour accommoder plus qu'un modèle de certification, voire permettre à des individus ou à des groupes d'usagers d'établir un canal sécurisé sans l'exigence d'attachement à un modèle de confiance spécifique. De plus, cette infrastructure doit permettre la dissociation entre les concepts de hiérarchie de nommage et de hiérarchie de certification.

Notre étude de cas a montré que notre approche est capable de répondre aux besoins ci-dessus sans beaucoup d'impact ou d'ajout de complexité aux systèmes de certification. En plus d'un certificat, chaque CA n'a qu'à émettre la politique correspondante. De plus, une même politique peut s'appliquer à plus d'un certificat pour une même entité certifiée. Du côté des applications, celles-ci n'ont pas besoin de récupérer les mêmes politiques à chaque fois qu'un chemin de certification est vérifié. Ayant la même durée de validité que le certificat correspondant, ces politiques peuvent être conservées localement jusqu'au moment où ce certificat n'est plus valide.

Notre méthode rend viable la mise en place d'une infrastructure commune pour les services d'authentification dans la mesure où elle rend explicite les sémantiques liées à la certification et offre un format commun pour la publication de politiques, permettant ainsi la coexistence et l'harmonisation entre configurations hétérogènes. L'ajout d'un nouveau modèle à cette infrastructure commune exige seulement l'attribution de nouveaux identifiants d'objet.

Un autre avantage de notre méthode est l'élimination du besoin d'imposition de règles de nommage restrictives telles que la subordination de noms. La méthode permet également d'associer à chaque utilisateur d'autres identificateurs que le nom d'annuaire. Cela permet que les utilisateurs puissent être identifiés par un nom d'annuaire unique, indépendant du (des) système(s) de certification qu'il intègre et des services sécurisés auxquels l'utilisateur accède.

Finalement, notre méthode permet à l'utilisateur final de déterminer la portée et la sémantique liées à chaque certificat, ainsi que d'identifier chaque modèle de certification et l'interfonctionnement entre politiques différentes. Cela permet aux utilisateurs de prendre des décisions précises basées sur des certificats, tout en tenant compte de l'hétérogénéité du système global.

CHAPITRE 14
VERIFICATION DE CLES PUBLIQUES

Comme nous l'avons déjà mentionné, la mise en place d'un cadre commun d'authentification et l'expansion des architectures de sécurité regroupant diverses politiques et CA de différents niveaux d'assurance mettent en cause la confiance accordée aux chemins de certification. Dans des tels environnements, la preuve de l'identité d'une entité exige plus que la simple vérification de la signature de son certificat. En effet, cela demande la définition d'un mécanisme d'évaluation capable de prendre en compte les besoins en sécurité du vérificateur.

Dans ce chapitre nous cherchons à discuter et à proposer des solutions aux questions liées à la vérification de clés publiques dans un environnement ouvert et hétérogène. Le chapitre se décompose en cinq sections. Dans la Section 14.1 nous abordons la question de l'évaluation de chemins de certification. La Section 14.2 s'attaque aux problèmes liés à la récupération de listes de révocation. La Section 14.3 présente la façon dont les descriptions de politiques (voir 13) peuvent être utilisées pour la validation de certificats. Dans la Section 14.4 nous développerons un mécanisme permettant aux domaines de sécurité de dériver des critères locaux en utilisant les éléments des politiques de certification émises par les autorités des modèles de certification. Finalement, dans la Section 14.5 nous présenterons une méthode d'expression de métriques d'évaluation et une fonction d'évaluation qui compléteront notre mécanisme d'évaluation de chemins de certification.

14.1. L'Evaluation de Chemins de Certification

La Recommandation X.509 propose un cadre d'authentification où les CA sont implicitement homogènes. Cependant, le besoin d'une infrastructure de certification, globale à plusieurs communautés différentes, avec des politiques et sémantiques de certification diverses a amené la définition de systèmes de certification comportant des CA de différents niveaux d'assurance. Cela rend nécessaire l'introduction de mécanismes d'évaluation de chemins de certification.

La question de l'évaluation de chemins de certification a été déjà soulevée par Altarah à travers le modèle Chimæra [Alt92]. Ce modèle propose un mécanisme d'évaluation des certificats basée sur l'attribution de valeurs de confiance à toutes les CA et à tous les certificats. Il ne nous semble pas réaliste de garantir que toutes les CA au niveau mondial pourront attribuer des valeurs à toutes les autres CA et à tous les certificats. Cela paraît difficilement "gérable".

Si une telle attribution de valeurs est restreinte aux CA d'un même domaine, alors pour l'authentification entre deux principaux appartenant à des domaines distincts, il serait nécessaire que les CA d'un domaine fassent confiance aux CA de l'autre domaine quant à l'attribution de valeurs de confiance. De plus, l'avance de la technologie impliquerait l'augmentation ou la diminution de ces valeurs en fonction de la capacité de chaque CA à suivre cette évolution.

Puisque l'objectif d'un tel système est de protéger l'utilisateur final, les besoins de ces utilisateurs doivent être pris en compte lors de l'élaboration de tels mécanismes. Lorsqu'un principal (utilisateur ou application) souhaite obtenir la clé publique d'un autre principal à travers l'établissement d'un chemin de certification, le premier souhaite que ce chemin soit le plus fiable possible et compatible avec l'objectif du processus d'authentification qu'il désire initier. Par exemple, si un utilisateur établit un chemin de certification pour envoyer un message PEM, son intérêt est que ce chemin respecte la politique de certification définie pour les procédures de sécurité PEM.

En outre, on ne peut pas prendre comme hypothèse que tous les KMD adopteront le même modèle de distribution de clés, ni que toutes les CA auront la même capacité ou le même niveau de protection. En fait, une même CA peut certifier d'autres CA de différents niveaux d'assurance. Il faut donc considérer l'hétérogénéité du système d'un point de vue macroscopique.

Cette réflexion nous amène à proposer un mécanisme modulaire d'évaluation des chemins de certification. Quatre niveaux d'évaluation peuvent être envisagés :

(1) Vérification Minimale

Ce niveau d'évaluation est implicite au processus de vérification d'un chemin de certification. Dans cette étape, il est vérifié si le chemin constitue une chaîne de certificats (signatures) vérifiable récursivement, si chaque certificat dans la chaîne n'est pas périmé ni ne figure dans la dernière liste de révocation de la CA émettrice.

(2) Vérification du Modèle

Ce niveau d'évaluation vérifie le respect du modèle de distribution de CA exigé par la procédure d'authentification définie pour le service que l'utilisateur souhaite initier. Cette partie du mécanisme d'évaluation est basée

sur les *politiques de certification* sous lequelles les certificats (dans le chemin) sont émis.

L'obtention et la vérification de chemins de certification doit prendre en compte le respect de ces politiques, permettant ainsi que et la violation de ces dernières soit immédiatement détectée.

(3) Vérification des Règles Globales au KMD

La validation de chemins de certification doit prendre en compte le respect non seulement des politiques de certification, mais aussi la politique de sécurité adoptée dans un domaine pour l'évaluation de chemins de certification.

Ce niveau d'évaluation vérifie donc le respect des règles globales au KMD et/ou des règles définies particulièrement pour le site.

(4) Vérification des Règles d'Evaluation Personelles

Cela implique la définition de métriques d'évaluation basées sur le choix personnel de l'utilisateur.

La mise en œuvre de l'étape (1) demande la disponibilité d'un mécanisme fiable pour la récupération de listes de révocation de certificats. L'étape (2) nécessite la disponibilité d'une représentation interprétable des politiques de certification. Nous avons proposé (dans 13) une technique pour la description de ces politiques. Les étapes (3) et (4) impliquent la définition d'un moyen d'expression de critères et de métriques d'évaluation de chemins de certification.

Le restant de ce chapitre proposera donc des moyens permettant de réaliser chacune des étapes de ce mécanisme d'évaluation. Il doit être noté que les étapes (3) et (4) fournissent des moyens complémentaires d'évaluation des chemins de certification sans imposer de fonctionnalité additionnelle aux entités certificatrices des modèles de certification. L'intégration de tels mécanismes dans notre approche serait comme illustrée par la Figure 30.

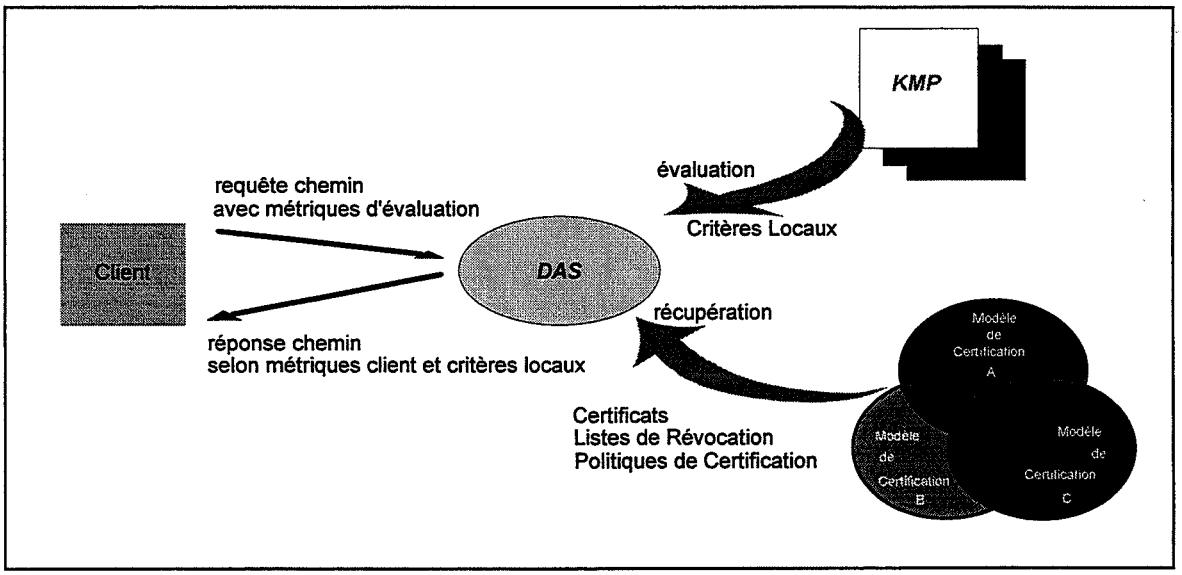

Figure 30. Rapport entre les Politiques de Certification, les Critères Locaux et le Fonctionnement du DAS

14.2. Récupération de Listes de Révocation

Un des sujets très abordés aujourd'hui est celui concernant la récupération des listes de révocation. En principe, toute CA est censée publier une liste de révocation de certificats ou CRL (**C**ertificate **R**evocation **L**ist) à intervalles définis par sa politique de sécurité. Cette liste doit être publiée même si elle est vide, pour indiquer qu'aucun certificat n'a été annulé. La CRL indique la date prévue pour la prochaine émission de la liste, mais si entre-temps un certificat est révoqué, par exemple, en cas de compromission de la clé privée correspondante, la CA doit remplacer cette liste pour une autre mentionnant l'annulation de ce certificat.

Selon les mécanismes existants, un utilisateur peut demander la révocation de son certificat à n'importe quel moment. Une CA peut aussi révoquer un certificat unilatéralement si la liaison exprimée par le certificat n'est plus valide.

Les CA sont supposées publier leur CRL dès qu'un nouvel élément doit être ajouté. En contrepartie, il n'existe pas de règles concernant la fréquence de publication des CRL, ni sur le degré de fiabilité, de disponibilité et de temps de réponse qui doit être fourni. Manifestement, ces aspects doivent être décidés par l'utilisateur, ou au moins chaque CA, lors du choix de la politique de sécurité adoptée, ce qui dans les normes est cité comme des "questions locales".

Les CRLs contiennent une estampille de temps, affectée par la CA émettrice, au moment de la génération de la liste. Par contre, la date de publication, i.e., de mise à disposition de la liste n'est pas indiquée. De plus, puisqu'une liste peut être remplacée à n'importe quel moment, i.e., dés qu'un certificat doit être annulé, on ne peut pas être sûr qu'une certaine liste est la plus récente.

Par conséquent, en utilisant les moyens disponibles aujourd'hui, si le destinataire d'un document signé désire en prouver la légitimité, il doit le soumettre à un serveur d'estampillage de temps (*trusted time server*) afin d'établir exactement la date de réception du document. Ensuite, il doit attendre la prochaine émission prévue de CRL par la CA émettrice du certificat de l'expéditeur afin d'établir le fait que ce certificat n'a pas été annulé.

Cela signifie qu'en utilisant les mécanismes existants, lorsqu'on envoie un message chiffré à quelqu'un d'autre, la confidentialité du message n'est pas complètement assurée car on n'est pas sûr que la clé du destinataire n'a pas fait l'objet de compromission. Il est clair qu'un tel mécanisme n'est pas approprié à des transactions du type commercial, par exemple, ou dans n'importe quel contexte où la révocation de droits est importante.

Il s'ensuit que si un certificat représente une liaison entre l'identité d'un principal et sa clé publique et si cette liaison doit être légalement valable, alors le mécanisme de distribution périodique de listes de révocation est déficient et ne répond pas complètement aux besoins d'authentification. Des mécanismes de requête/réponse seront éventuellement exigés.

Il faut noter que la disponibilité et l'ubiquité des systèmes d'annuaire X.500 ne résolvent pas ce problème. Outre le fait que ce système d'annuaire n'est pas conçu pour être globalement synchronisé, cela demandera des efforts de gestion des caches en ce qui concerne l'élimination des entrées contenant des listes de révocation obsolètes. Cela, entre autres, justifie l'indépendance de notre approche par rapport au service de noms.

Nous pouvons envisager plusieurs solutions au problème de récupération de listes de révocation :

(1) Inclusion d'une fonctionnalité additionnelle aux CA permettant le support d'un mécanisme requête/réponse pour la récupération des listes de révocation. La requête pourrait être de trois types :

(1.1) requête d'envoi de la dernière liste,

(1.2) requête d'envoi d'une liste avec une estampille de temps signée indiquant la date et heure courantes,

(1.3) requête de confirmation de la validité d'un certificat en particulier, i.e., ayant comme entrée l'identification d'un utilisateur X, certificat numéro N, la CA confirme si ce certificat est toujours valide.

(2) Restructuration des CA permettant l'inclusion d'un service du type *répondeur* spécifique à la récupération des listes de révocation. Cela correspond à l'ajout d'un module à la CA qui servirait comme dépositaire des listes de révocation de certificats. Ce module serait attaché à la CA mais de façon complètement indépendante de la fonctionnalité normale de cette entité.

L'implémentation de cette solution serait telle que chaque CA serait chargée d'envoyer sa dernière CRL à son répondeur, qui serait chargé de répondre aux requêtes (1.1) à (1.3) ci-dessus.

La solution (1) exige que la CA opère comme une application *on-line* et pourtant ne peut être adoptée par des architectures dont la politique de sécurité ne le permet pas. Cette solution fournit une grande fonctionnalité au niveau de la révocation de clés mais contredit la tendance actuelle d'établissement de CA dans des dispositifs du type "*key vault*".

La solution (2) ne présente pas de restriction au niveau de la protection de la clé privée de la CA ni n'impose que la CA soit *on-line*. En fait, ce répondeur peut être vu comme un dispositif *on-line* de la CA qui est complètement indépendant des zones protégées de la CA et qui n'a pas d'accès à ces zones. Ces deux parties ne se communiquent que lors de la mise à disposition d'une nouvelle CRL. Ce mécanisme suppose que le répondeur dispose d'une paire de clés dont le composant public est certifié par la CA à laquelle ce répondeur est attaché.

Il doit être noté que si la confirmation (1.3) ci-dessus n'est pas signée par la CA, les utilisateurs sont obligés de faire confiance à la confirmation générée par le répondeur. De plus, cela augmente les chances d'une attaque réussie où un adversaire essaie de détourner la fonctionnalité du répondeur pour qu'il retourne des confirmations fausses.

Il est donc souhaitable que la CA soit capable de produire des accusés de révocation individuels pour chaque certificat révoqué. Cela a l'avantage additionnel de permettre aux utilisateurs de conserver ces accusés à des fins de non-répudiation de révocation. Chaque accusé individuel pourrait avoir la définition suivante en ASN.1 :

```
revokedCertificate      SIGNED SEQUENCE {
    issuer                  Name,
    subject                 Name,
    certificate             CertificateSerialNumber,
    revocationDate          UTCTime }
```

L'ensemble de nouveaux accusés individuels, que la CA envoie à son répondeur, aurait donc la définition suivante en ASN.1 :

revokedCertificates ::= SEQUENCE OF revokedCertificate

Nous décrivons ci-dessous, les protocoles qui composent ce nouveau mécanisme. La Figure 31 illustre les rapports entre la CA et son répondeur et entre celui-ci et les utilisateurs.

Notation :

A → B : represente "A envoie à B"

{data}A : indique les données *data* signées au moyen de la clé privée de A.

Protocoles :

(1) Le protocole de mise à jour de CRL est constitué d'un échange unique, initié par la CA :

CA → répondeur : CRL, revokedCertificates

Comme cet échange est toujours initié par la CA, ce mécanisme n'ajoute pas de vulnérabilité à la CA.

(2) Le répondeur procède selon les requêtes d'utilisateur, de la sorte :

(2.1) si l'utilisateur souhaite seulement avoir la dernière liste émise par la CA, l'échange prend la forme ci-dessous.

utilisateur → répondeur : requete_CRL
répondeur → utilisateur : {CRL}CA

(2.2) si l'utilisateur souhaite avoir la dernière liste accompagnée d'une estampillle de temps, l'échange prend la forme ci-dessous. La reponse est signée avec la clé privée du répondeur.

utilisateur → répondeur : requete_CRL_estampille
répondeur → CA : {CRL, date}répondeur

(2.3) si l'utilisateur souhaite disposer de la confirmation de la validite du certificat numéro N d'un utilisateur X, l'échange prend la forme ci-dessous. La réponse est signée avec la clé privée du répondeur.

utilisateur → répondeur : requete_confirmation_certificat_NX
répondeur → CA : {N, X, confirmation, date}répondeur

La confirmation dans (2.3) ci-dessus est du type *revokedCertificate*, si le certificat en question a été revoqué. Sinon, la confirmation est vide.

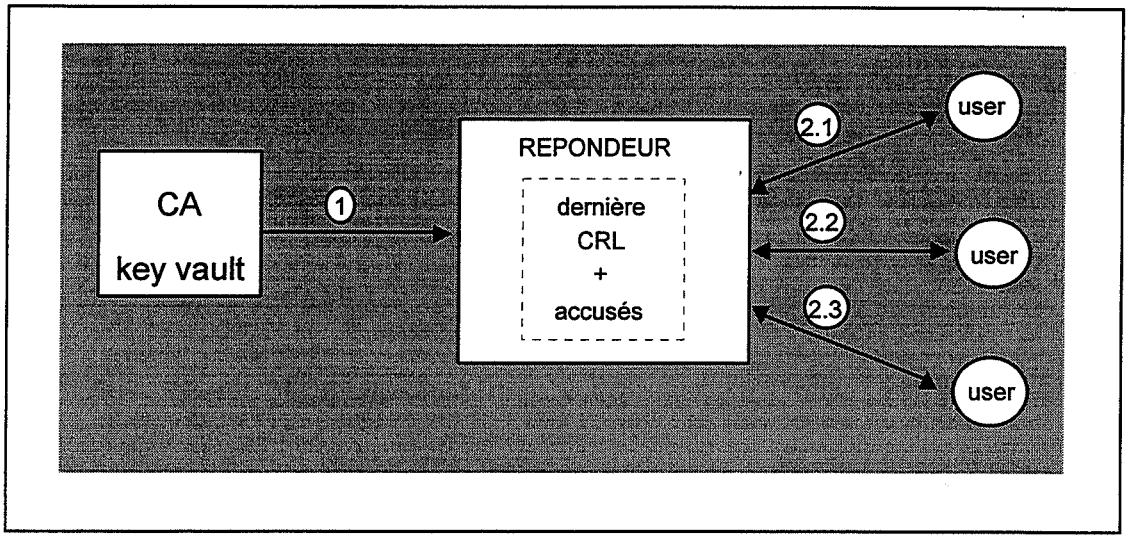

Figure 31. Rapport entre une CA, son répondeur et les utilisateurs

Cette solution peut être intégrée dans le modèle présenté dans ce travail par deux différentes approches à travers le DAS :

(i) Le DAS communique avec les répondeurs des CA internes à son KMD. La Figure 32 illustre une telle configuration.

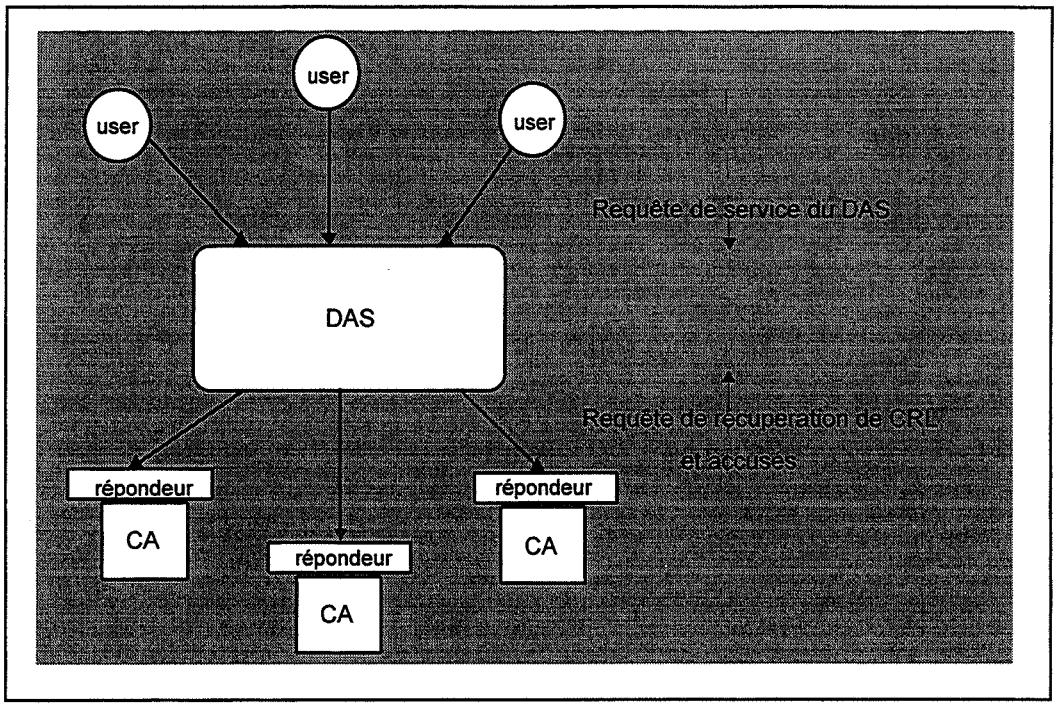

Figure 32. Rapport entre le DAS et les répondeurs de CA

(ii) Le DAS constitue lui-même le répondeur des CA internes à son KMD. Dans ce CA, le DAS joue le rôle de répondeur pour les CA, i.e., c'est avec le DAS que les CA échangeront leur CRLs. Cette approche améliore la performance de la fourniture des services du DAS, mais, en contrepartie, elle peut ajouter un délai entre la date de génération d'une liste et sa date de mise à disposition.

Ce délai constitue, en fait, une vulnérabilité du service car pendant ce temps-là, des procédures d'authentification qui devraient échouer peuvent erronément réussir. Cette vulnérabilité peut être éliminée par l'ajout d'une étape dans le protocole de mise à jour de CRL : cette nouvelle étape est constituée d'une "alarme" à travers laquelle une CA prévient au DAS qu'elle initie une procédure de révocation.

Bien que cette nouvelle étape permette d'éliminer la vulnérabilité décrite ci-dessus, elle alourdit la fonctionnalité des CA et du DAS : lorsqu'une CA doit initier la révocation d'un certificat, elle envoie une alarme au DAS, qui à son tour met en attente les requêtes dont l'exécution comprend des certificats émis par la CA en question. Après la génération de la nouvelle liste, la CA donnera suite au protocole de mise à jour de CRL avec le DAS, qui alors pourra procéder à son opération normalement. Bien entendu, cette alarme doit être signée par la CA émettrice de façon à prouver son origine.

La Figure 33 illustre la configuration (ii).

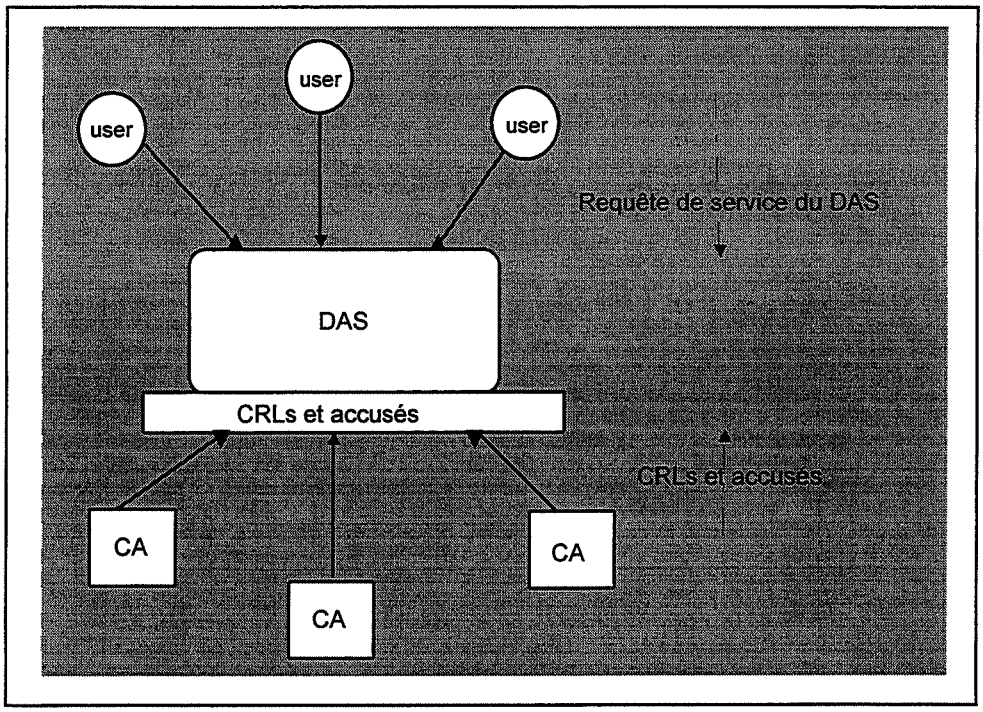

Figure 33. Intégration des Répondeurs de CA dans le DAS

Il doit être noté qu'une combinaison des approches (i) et (ii) ci-dessus peut être aussi adoptée. Cela offre une certaine flexibilité entre le mode d'opération des CA et leur intégration au modèle.

14.3. Utilisation des Description de Politiques de Certification

De la même façon qu'un certificat, dans un chemin de certification, vérifie la signature du certificat suivant dans la chaîne, l'attribut de description de politique peut être utilisé pour vérifier si le certificat suivant dans la chaîne est conforme à la politique que son émetteur doit respecter. Cet attribut sert également à déterminer d'autres aspects liés au certificat correspondant, comme nous le verrons par la suite. Prenant l'attribut de description de politique en considération, nous pouvons représenter un chemin de certification de la façon suivante :

$$(C_i, P_i) \rightarrow (C_{i+1}, P_{i+1}) \rightarrow \ldots \rightarrow (C_n, P_n);$$

où, C_k est le $k^{ième}$ certificat dans la chaîne, P_k est la politique correspondante et :

$$C_k ::= (I_k, S_k, PK_k)$$

$$P_k ::= (TK_k, ET_k, CS_k, AL_k, RfC_k);$$

où, I_k, S_k, PK_k representent, respectivement, le nom de l'emetteur (*issuer*), le nom du propriétaire (*subject*) et la clé publique contenus dans le certificat C_k. De même, TK_k, ET_k, CS_k, AL_k, RfC_k representent les éléments de l'attribut *policy*, i.e., has_trusted_key, entityType, certificateSemantic, assuranceLevel et rulesForCertification. Nous allons utiliser cette notation pour décrire la façon dont les descriptions de politique peuvent être utilisées dans la vérification de chemins de certification.

Tout d'abord, la clé publique PK_i contenue dans le certificat C_i, est verifie la signature de C_{i+1} et de P_{i+1}. Ensuite, P_i peut être utilisé pour identifier :

- le modèle de confiance sous lequel C_i a été emis, utilisant ET_i,
- le rôle de S_i dans le modèle de confiance, utilisant ET_i,
- le niveau d'assurance accordé à S_i, utilisant AL_i.

En ce qui concerne les certificats suivants dans le chaîne, P_i peut être utilisé pour vérifier :

- si S_i est autorisé à emettre des certificats, utilisant CS_i,
- si S_{i+1} est sous la juridiction de S_i, utilisant RfC_i.jurisdiction.entities,
- si la validité deC_{i+1} est conforme, utilisant RfC_i.certificate_validity,
- si $S_i$est autorisé à emettre des certificats ayant la semantique de C_{i+1}, en comparant RfC_i.jurisdiction.semantic avec CS_{i+1},
- l'interfonctionnement entre politiques differentes, utilisant CS_i ; en particulier, l'interfonctionnement entre modèles de confiance peut être identifié en comparant ET_i avec ET_{i+1},
- la longueur du chemin, utilisant RfC_i.how_many_levels_up_to_the_leaf.

En ce qui concerne les listes de révocation, l'élément RfC.revocation_frequency peut être utilisé pour estimer la "fraicheur" d'une CRL, si le format des listes X.509 est utilisé. Cet élément peut être aussi utilisé pour vérifier si la CA respecte la frequence établie pour la mise à jour de CRL.

L'élément RfC.crl_responder_address peut être utilisé pour contacter le répondeur de la CA correspondante, si applicable.

Finalement, la description de politique correspondante au dernier certificat dans le chemin (le certificat de l'utilisateur authentifié) peut être utilisée de deux manières différentes :

- si l'utilisateur en question est le signataire d'une certaine portion de données, la description de politique sert à vérifier si les données signées sont conformes à la portée et aux règles d'usage associées au certificat de l'utilisateur,
- la description de politique sert également à obtenir, de façon sûre, d'autres attributs et formes d'identification associés à l'utilisateur.

Bien entendu, l'usage de l'attribut de description de politique peut être fait avec plus ou moins de rigueur selon le niveau de sécurité souhaité. Il doit être également noté que les outils chargés de la obtention/vérification de certificats n'ont pas besoin de récuperer cet attribut à chaque fois qu'un certificat (ou chemin de certification) doit être validé. Puisque la validité de l'attribut *policy* est strictement liée à celle du certificat correspondant, cet attribut peut être conservé localement jusqu'à ce que la date d'expiration de ce certificat soit atteinte.

14.4. Critères Locaux d'un Domaine

Indépendamment des rôles et degrés de confiance accordés aux CA par les modèles publics, en pratique l'utilisateur final peut avoir son propre aperçu du système. En fait, chaque utilisateur peut accorder un degré de confiance différent à chaque CA ou placer les CA différemment dans sa hiérarchie privée. Ces divers degrés se justifient puisque le rapport entre l'utilisateur et les CA peut varier. Des CA géographiquement ou politiquement distantes n'auront pas le même degré de confiance car leur rapport avec l'utilisateur sera moins proche.

Par exemple, normalement un utilisateur accordera plus de confiance à sa CA locale car il connaît son opérateur et peut se fier à lui pour ne pas agir avec malveillance. Individuellement, un utilisateur peut accorder une juridiction différente à la CA, ou bien, il peut refuser de croire à des affirmations sur lesquelles la CA a juridiction, ou même croire à des affirmations faites par la CA hors de sa juridiction.

De même, il se peut que la politique de sécurité d'un domaine ou communauté d'utilisateurs exige l'imposition de contraintes à l'ensemble des modèles existants. Ceci constitue les critères locaux d'un domaine. Lors des procédures d'authentification, ces critères seront consultés pour autoriser ou interdire l'établissement des canaux sécurisés avec les entités du domaine en question.

De la sorte, en rapport avec l'ensemble des modèles existants, la politique de sécurité d'un domaine peut :

- sélectionner un sous-ensemble de modèles,
- établir des restrictions aux modèles sélectionnés,
- définir des rôles différents pour certaines entités ou certains types d'entités d'un modèle,
- attribuer des degrés de confiance différents à certaines entités d'un modèle,
- accorder une juridiction différente à certaines entités d'un modèle,

- refuser des certificats émis par un type donné de CA ou par une CA spécifique,
- sélectionner un sous-ensemble d'algorithmes admis dans le domaine,
- définir la fréquence de récupération des listes de révocation.

Par exemple, un domaine peut interdire l'interfonctionnement avec le modèle PEM, ou refuser des certificats émis par des CA du type Persona ou par des CA situées dans des régions d'une certaine tendance politique.

Il doit être noté que les mécanismes de sécurité sont déjà, par leur nature, coûteux au niveau du calcul. Pour éviter d'augmenter le coût de ces mécanismes, on doit chercher à minimiser le trafic généré pour l'obtention de certificats et de listes de révocation nécessaires à l'authentification. En ce qui concerne les certificats et politiques, il est convenable d'utiliser un système de caches (voir 14.3). Quant aux listes de révocation, en plus des caches, il est convenable d'établir, au niveau de chaque domaine, la fréquence de récupération des listes de révocation. Selon les besoins en sécurité d'un domaine, son administration peut établir, par exemple, qu'une liste est valide 24 heures après sa récupération en toute circonstance.

En utilisant les éléments de notre méthode de description de politiques de certification, nous avons défini un mécanisme pour la spécification des critères locaux d'un domaine :

```
DomainePolicyDescription ::= SIGNED SEQUENCE {
        accept          [0] SET OF DomainePolicy OPTIONAL,
        refuse          [1] SET OF DomainePolicy OPTIONAL,
        strict_policy   [2] BOOLEAN OPTIONAL,
        crl_retrieval   [3] INTEGER – frequency in hours
        signedBy        [4] DistinguishedName,
        signedAt        [5] UTCTime}
```

Les critères spécifiés par *DomainePolicyDescription* sont signés afin de garantir leur authenticité et leur intégrité. Les paramètres de *DomainePolicyDescription* ont la signification suivante :

- *accept* : spécifie les éléments autorisés *ou* les éventuelles restrictions à des éléments des modèles publics,
- *refuse* : spécifie les éléments interdits dans le domaine,
- *strict_policy* : indique si tout autre élément non référencé par le champ *accept* doit être refusé (strict_policy = TRUE) ou si le champ *accept* indique seulement des restrictions à certains éléments des modèles publics[30] (strict_policy = FALSE) ; cet élément doit être present si le champ *accept* est specifié,
- *crl_retrieval* : indique la fréquence (en heures) de récupération des listes de révocation,
- *signedBy* : indique le nom d'annuaire du responsable de la définition des critères,
- *signedAt* : indique la date de la génération de la signature.

Les éléments de certification autorisés ou interdits dans un domaine sont spécifiés à travers le paramètre *DomainePolicy*, ayant la définition suivante :

---

[30]Cela sert à éviter que les domaines aient à décrire les modèles publics dans leurs politiques privées.

```
DomainePolicy ::= SEQUENCE {
    model           [0] CertificationModel      OPTIONAL,
    certificate_id  [1] SEQUENCE {
                            number    SerialNumber,
                            issuer    DistinguishedName } OPTIONAL,
    subject         [2] DistinguishedName       OPTIONAL,
    policy          [3] PolicyDescription       OPTIONAL
          – at least one of them must be present – }
```

Les champs de *DomainePolicy* ont la signification suivante :

- *model* : identifie tout un modèle, ainsi lorsqu'il est présent tous les autres champs n'ont pas de signification et doivent donc être absents,
- *certificate_id* : permet, avec le champ *subject*, d'identifier un certificat en spécifiant son numéro de série et le nom d'annuaire de l'entité émettrice,
- *subject* : indique le nom d'annuaire d'une entité,
- *policy* : permet de prescrire des restrictions et de redéfinir des paramètres établis par un modèle public selon la politique de sécurité d'un domaine. En supposant l'utilisation de ce même type (*PolicyDescription*) pour la description des politiques de certification (tel que décrit dans 13.3), la reattribution des valeurs est faite dans les mêmes termes.

  Pour simplifier la description des critères locaux nous avons adopté l'exception suivante : tous les champs de *PolicyDescription* sont considérés comme *optionnels*.

Dans l'exemple 11 ci-dessus, un domaine établit que le seul modèle public qu'il accepte est le modèle Password. Par conséquent, tout certificat émis sous une autre politique sera refusé.

Exemple 11

```
DomainePolicyDescription :
   accept :
     model :          oid-model-public-password
   strict_policy :    TRUE
   signedBy :         < C=Pays; O=Organisation; CN= L'Administrateur Sécurité >
   signedAt :         Mercredi 12 May 1994 19:32
```

L'exemple 12 décrit des critères où le domaine déclare n'accepter que des certificats émis sous le modèle PEM et établit que la CA de plus haut niveau n'est pas l'IPRA mais la PCA de nom < C=Pays; O=Autorité de Certification X >. Comme résultat, les seuls certificats admis dans le domaine sont ceux sous l'arborescence ayant cette CA comme racine.

Exemple 12

```
DomainePolicyDescription :
   accept :
     model :          oid-model-public-pem
   accept :
     certificate_id
          number :    5
          issuer :    < L=Internet; O=IPRA >
     subject :        < C=Pays; O=Autorité de Certification X >
     policy :
                      has_trusted_key : TRUE
```

```
                    entityType : oid-model-public-pem-pca
strict_policy :     TRUE
signedBy :          < C=Pays; O=Organisation; CN= L'Administrateur Sécurité >
signedAt :          Mercredi 12 May 1994 19:54
```

Dans l'exemple 13, un domaine accepte tous les modèles existants mais attribue un niveau d'assurance moyen à la CA < C=Pays; O=Autorité de Certification Y >, indépendamment de celui déclaré par le modèle public. Quant aux certificats émis par la CA < C=Pays; O=Autorité de Certification Z >, le domaine n'accepte que ceux émis pour des CA Organisationnelles du modèle PEM appartenant aux organisations W et K dans le pays "Pays". De plus, le domaine refuse tout certificat du type Persona du modèle PEM et tous les certificats des utilisateurs de "l'organisation Z" dans le pays "Pays".

Exemple 13

```
DomainePolicyDescription :
   accept :
     subject :     < C=Pays; O=Autorité de Certification Y >
     policy :
                   entityType : oid-model-public-password-pca
                   assuranceLevel : medium
   accept :
     subject :     < C=Pays; O=Autorité de Certification Z >
     policy :
          entityType :    oid-model-public-pem-pca
          rulesforCertification :
               juridisdiction :
                    namingRules :
                                   entityType : oid-pem-ca
                                   and equality : C=Pays and equality : O=Organisation W
                                   and equality : C=Pays and equality : O=Organisation K
                    key_restriction :
                         algorithm :      rsa
                         parameters : 1024
   refuse :
          policy : entityType : oid-model-public-pem-persona
   refuse :
     policy :
          rulesforCertification :
               juridisdiction :
                    namingRules :        and equality : C=Pays
                                         and equality : O=Organisation  Z
   strict_policy :     FALSE
   signedBy :          < C=Pays; O=Organisation; CN= L'Administrateur Sécurité >
   signedAt :          Mercredi 12 May 1994 20:33
```

14.5. Métriques d'Evaluation

Outre l'expression des critères d'un domaine, le mécanisme d'évaluation doit permettre à un utilisateur de définir la métrique et le seuil permis dans la construction et vérification d'un chemin de certification. Les métriques d'évaluation constituent les paramètres de mesure de fiabilité d'un chemin de certification. De plus, cela permet de restreindre les informations échangées sur le réseau à celles satisfaisant les conditions définies par la métrique et le seuil [Alt92].

En contrepartie, il doit être noté que la définition et l'interprétation tant des métriques que de la fonction d'évaluation constituent des questions de domaine plutôt local. Celles-ci dépendront et varieront selon le contexte dans lequel la communication a lieu, par exemple le modèle de certification utilisé, l'objectif des procédures d'authentification et le niveau de sécurité souhaité. Ainsi, dans la suite de cette section, nous allons donner un exemple simple d'application de notre approche pour la description de politiques de certification dans le cadre de l'évaluation de chemins de certification.

Au premier abord, nous pouvons envisager trois types de métriques :

(1) *la longueur d'un chemin*

La sémantique d'évaluation est définie en fonction du nombre de certificats de CA qui composent un chemin.

(2) *le niveau d'assurance d'un chemin*

La sémantique d'évaluation est définie en fonction du niveau d'assurance attribué aux certificats de CA qui composent un chemin.

(3) *une combinaison de (1) et (2)* ci-dessus.

Pour une métrique du type (1) ci-dessus, la valeur ou résultat d'évaluation est directe. Il suffit de compter le nombre de certificats de CA dans un chemin. En contrepartie pour une métrique du type (2), la fonction d'évaluation doit prendre en compte le niveau d'assurance accordé à chaque certificat de CA constituant un chemin.

Dans notre méthode de description de politiques de certification (voir 13.3), une CA attribue un niveau d'assurance à chaque certificat de CA émis. Ce niveau d'assurance correspond au montant de confiance accordé à cette dernière vis-à-vis de la politique de sécurité de la CA émettrice. A l'intérieur d'un KMD, la politique de sécurité peut attribuer d'autres valeurs à ces mêmes certificats à travers le champ *accept* de la description des critères locaux, conforme décrit dans la Section 13.4.1.

Nous avions défini quatre niveaux d'assurance : inconnu (*unknown*), faible (*low*), moyen (*medium*) et haut (*high*). Pour définir la fonction d'évaluation, nous proposons une attribution de valeurs de méfiance à ces niveaux d'assurance, comme montré dans le tableau ci-dessous.

niveau d'assurance	valeur de méfiance (dv)
high	0
medium	1

low	2
unknown	-1

Puisque notre approche doit prendre en considération l'hétérogénéité au niveau des catégories de CA, nous proposons également une attribution de coefficients de pondération (w) aux divers types de CA existants. En particulier, ces coefficients de pondération doivent être des valeurs supérieures à un (1) ; le coefficient 1 étant associé, par défaut, aux types de CA pour lesquels aucun coefficient n'a été attribué. La fonction d'évaluation est alors la suivante :

$$evaluation = \frac{\sum_{i=1}^{i=n} dv_i \times w_i}{\sum_{i=1}^{i=n} w_i}$$

Où,

- n est le nombre de certificats de CA dans un chemin,
- dv_i est la valeur de méfiance associée au certificat i dans un chemin ; $1 \leq i \leq n$,
- w_i est le coefficient de pondération associé au type de CA propriétaire du certificat i dans un chemin ; $1 \leq i \leq n$.

La fonction d'évaluation est telle que le résultat définit la valeur de méfiance d'un chemin. Cette valeur correspond à un niveau d'assurance vis-à-vis de la correspondance de valeurs montrée dans le tableau ci-dessus.

Lorsque toutes les CA qui constituent un chemin sont d'un même niveau d'assurance, la fonction d'évaluation retournera une valeur correspondante à ce niveau. Lorsque le résultat d'évaluation est un nombre non entier, celui-ci doit être interprété localement, vis-à-vis des valeurs définies pour les niveaux d'assurance, au choix de l'utilisateur.

A titre d'exemple, considérons un chemin contenant les certificats suivants :

- C_1 d'un niveau d'assurance *high*, le type de la CA émettrice étant associé au coefficient $w_1 = 4$,
- C_2 d'un niveau d'assurance *medium*, le type de la CA émettrice étant associé au coefficient $w_2 = 3$,
- C_3 d'un niveau d'assurance *medium*, le type de la CA émettrice étant associé au coefficient $w_3 = 3$,
- C_4 d'un niveau d'assurance *low*, le type de la CA émettrice étant associé au coefficient $w_4 = 2$,

Un tel chemin aurait une valeur de méfiance 0,83 comme résultat d'évaluation, ce qui correspond à un niveau d'assurance approximativement moyen (*medium*). Si le certificat C_1 avait un niveau d'assurance faible (*low*), le chemin aurait une valeur de méfiance 1,5 comme résultat d'évaluation, ce qui correspond à un niveau d'assurance entre moyen (*medium*) et faible (*low*).

Dans le contexte de notre approche, la fonction d'évaluation doit être calculée par rapport aux critères locaux au KMD. Bien entendu, cette fonction pourrait prendre aussi en compte des critères personnels de chaque utilisateur. Cependant, ceci augmenterait la complexité de l'accès au DAS. Par conséquent, les utilisateurs finals doivent gérer cela localement.

Les métriques d'évaluation peuvent être spécifiées par un utilisateur à travers la structure ASN.1 suivante :

```
metrics ::= SEQUENCE {
        path-length     [0] INTEGER OPTIONAL,
        cp-assurance    [1] CpAssuranceIndication OPTIONAL,
        both            [2] BOOLEAN OPTIONAL }
```

Le composant *path-length* permet à un utilisateur d'indiquer au DAS la taille maximale du chemin, i.e., le nombre maximum de certificats de CA dans le chemin.

Le composant *cp-assurance* permet à un utilisateur d'indiquer au DAS le niveau minimum d'assurance du chemin de certification. Ce composant a la définition suivante :

```
CpAssuranceIndication ::= SEQUENCE {
    globalCpAssurance       [0] AssuranceLevel OPTIONAL,
    specificsCaAssurance    [1] SEQUENCE OF {
                                    caClass     OBJECT IDENTIFIER,
                                    value       AssuranceLevel
                                } OPTIONAL,
    minimumCAAssurance      [2] AssuranceLevel OPTIONAL }
```

L'élément *globalCpAssurance* spécifie le niveau minimum d'assurance d'un chemin. L'élément *specificsCaAssurance* permet de spécifier un niveau d'assurance spécifique pour des catégories particulières de CA dans le chemin. L'élément *minimumCAAssurance* permet de spécifier un niveau minimum d'assurance pour toutes les CA dans le chemin. Le moyen d'identification des catégories de CA et du niveau d'assurance a été décrit dans la Section 13.3.

Le résultat d'évaluation peut être rendu sous la forme suivante :

```
EvaluationResult ::= REAL
```

Les coefficients de pondération attribués aux catégories de CA font partie de la KMP d'un domaine. Ceux-ci peuvent être décrits sous la forme suivante :

```
CaClassesWeighting ::= SEQUENCE OF {
        caClass     OBJECT IDENTIFIER,
        weight      INTEGER
}
```

Comme nous l'avons déjà mentionné dans cette section, les coefficients de pondération doivent être des valeurs supérieures à 1. La valeur 1 est réservée pour être associée par défaut aux classes de CA auxquelles aucun coefficient n'a été attribué.

Il doit être noté que ce mécanisme d'évaluation peut également être appliqué à des modèles privés où le concept de CA n'existe pas. Il suffit de substituer le terme "CA" par "entité quelconque à laquelle l'on délègue la tâche de certification".

CHAPITRE 15
LE SERVEUR DE DONNEES POUR L'AUTHENTIFICATION

Il existe, dès maintenant, un besoin croissant des mécanismes pour l'obtention et la vérification des données publiques d'authentification, surtout entre configurations hétérogènes. Tous les mécanismes qui utilisent des systèmes cryptographiques asymétriques sont basés sur des mécanismes d'authentification [Men93] qui, à leur tour, sont basés sur l'obtention et la vérification correcte et fiable de clés publiques.

Il est clair que de tels mécanismes de sécurité ont besoin de mécanismes spéciaux qui permettent, à la fois, la mise en vigueur et le respect des règles définies dans un KMD, l'indépendance de chaque KMD au niveau du choix de la distribution des CA et du service de noms les plus appropriés à ses besoins.

De plus, comme déjà mentionné dans le Chapitre 8, nous disposons actuellement d'au moins trois types de structures des données définies pour représenter des chaînes de points de confiance, et qui constituent des variations sur le thème "chaîne de certificats".

Actuellement, les procédures d'obtention et de vérification de ces données publiques nécessaires à l'authentification sont encapsulées dans les propres applications. Il devient donc judicieux de disposer des mécanismes globaux capables de construire et de vérifier ces structures à partir de la structure de base certificat. Une approche modulaire doit isoler ces procédures de façon à alléger les applications et à permettre l'indépendance au niveau des architectures de gestion de clés. Les conséquences d'une telle approche sont :

- la facilité de migration et d'évolution des architectures de gestion de clés,
- l'indépendance des applications par rapport aux différents modèles de certification ; par exemple, la possibilité d'envoyer un message sécurisé X.400 en utilisant le modèle de certification PEM ou PGP,
- la possibilité de mise en vigueur des politiques de certification et des critères locaux à travers le mécanisme d'évaluation des chemins de certification.

Les aspects mentionnés ci-dessus constituent les points fondamentaux pour l'implantation d'une infrastructure globale de clés publiques pour les mécanismes de sécurité. Pour la réalisation d'une telle approche, ce travail propose l'ajout d'un nouvel élément dans l'architecture de gestion de clés : le Serveur de Données pour l'Authentification ou **DAS** (**D**ata **A**uthentication **S**erver). Le rôle de ce nouvel élément est précisé dans la suite de ce chapitre.

15.1. Rôle du DAS dans l'Architecture de Gestion de Clés

La fonction principale du DAS est d'obtenir et de vérifier, de façon correcte et fiable, les structures de données nécessaires aux mécanismes d'authentification dans le cadre des applications sécurisées. Il faut donc définir l'emplacement, les services et le fonctionnement de cet élément dans l'architecture de gestion de clés.

15.1.1. Emplacement du DAS

Lorsqu'un KMD est défini, sa KMP doit être spécifiée. Celle-ci comprend les critères locaux du KMD et la composition du KMD en ce qui concerne les autorités de certification. Le DAS doit être capable d'accéder à ces informations. La définition de la KMP sera décrite dans 14.2.3.1.

Il faut noter que cette approche n'impose pas de restrictions en ce qui concerne la configuration d'un KMD. Un KMD peut admettre un nombre quelconque de CA et des modèles quelconques de distribution de CA. En supposant l'utilisation de la méthode de description de politiques de certification proposée dans le Chapitre 13, le DAS peut identifier le modèle qu'une CA intègre. De plus, une même CA peut intégrer plusieurs modèles différents.

Les KMD peuvent être interconnectés pour permettre l'authentification de principaux appartenant à des KMD différents. L'interconnexion entre KMD est réalisée à travers la certification croisée entre CA appartenant à des KMD différents.

Le DAS est donc placé au niveau de chaque KMD. Le fonctionnement de ce serveur est basé sur la KMP, sur la connaissance du domaine qu'il sert et sur des références vers d'autres domaines auxquels ce domaine est connecté. Cela permet au DAS de détecter la violation de la politique de sécurité du KMD, et de fournir ses services de façon correcte et fiable.

15.1.2. Analyse des Services du DAS

Nous avons décrit plus haut l'existence des besoins réels d'un tel service et les types de données que le DAS doit manipuler. De façon à pouvoir structurer la description de la fonctionnalité fournie par le DAS, nous appelons le service fourni par des DAS comme *Service DAS*, et les utilisateurs de ce service seront appelés *Clients DAS*.

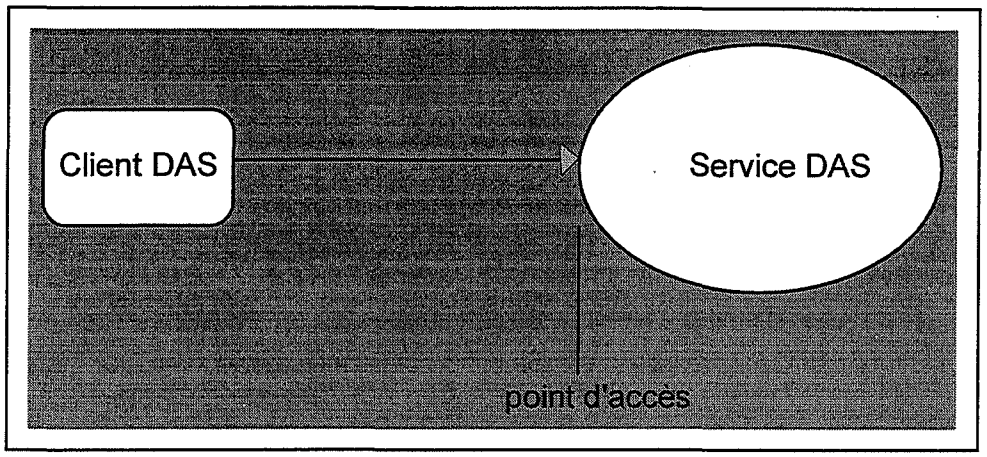

Figure 34. Modèle Fonctionnel du Service DAS

Nous pouvons considérer le service DAS comme un *objet* sur lequel se trouvent des points d'accès disponibles aux clients. Cela traduit une architecture classique du type client-serveur. Nous devons maintenant définir les classes d'interaction disponibles aux clients DAS.

Une exigence de base de ce service est qu'il soit proche des besoins des utilisateurs et qu'il réponde aux besoins des mécanismes d'authentification existants. Le DAS a été initialement conçu comme un élément de l'architecture SecUcomx. La définition initiale des services du DAS a été présentée dans la Partie III (voir 8.4.1). Nous avions défini une classe interaction pour chaque type de chemin disponible.

Cette approche a bien répondu à nos besoins durant le projet Password. Toutefois lors de l'évolution du concept du DAS, nous avons constaté qu'en pratique une implémentation basée sur une telle spécification ne serait pas convenable pour les raisons suivantes :

- toutes les structures du type "chemin de certification" sont des variations sur un même thème : chaîne de certificats. En particulier, du point de vue sémantique, nous remarquons deux types de chemin : le chemin complet et le chemin incomplet.

  Le chemin *complet* est tel qu'il contient toutes les clés nécessaires à l'authentification d'un utilisateur. Un chemin du type *incomplet* est tel qu'il ne contient pas toutes les clés nécessaires à l'authentification d'un utilisateur. Ainsi, chaque récepteur d'un chemin incomplet doit le compléter avec le mi-chemin qui correspond au graphe de certification entre le point final du chemin reçu et sa propre autorité de certification.

- nous pouvons clairement distinguer deux groupes de services : un premier groupe de services de récupération et un deuxième groupe de services de vérification de données.

Par conséquent, l'implémentation d'opérations différentes pour chaque type de structure implique une complexité superflue au niveau de l'accès au DAS. En effet, un tel service doit être tel que sa complexité soit la plus réduite possible, de façon à minimiser les efforts de codage, de gestion et par conséquent le temps de réponse.

En fait, nous avons deux problèmes principaux :

(1) la construction d'une chaîne de certificats,

(2) l'évaluation d'une chaîne de certificats.

Pour la construction d'un chemin de certification, le client doit spécifier l'identification des deux principaux (A et B) entre lesquels le chemin doit être établi et le type de chemin souhaité (complet ou incomplet). Lorsqu'un chemin du type complet est demandé, le Service DAS doit récupérer deux séries ordonnées de clés publiques nécessaires à l'authentification des deux principaux spécifieé, i.e., les chemins A→B et B→A.

Lorsqu'un chemin du type incomplet est demandé, l'un des principaux spécifiés constitue une CA commune à un modèle de confiance. Alors, le Service DAS doit récupérer la série ordonnée de clés publiques qui forme le chemin CA→A. Ultérieurement si le destinataire B d'un tel chemin fait confiance à cette CA, celui-ci le vérifie en utilisant la clé publique de cette dernière. Sinon, ce chemin doit être complété avec le chemin B→CA pour l'authentification du principal A.

Pour l'évaluation d'un chemin de certification, le client doit spécifier l'identification du principal auquel le chemin est adressé. Si le chemin est du type complet, le Service DAS doit récupérer les listes de révocation et les politiques de certification des autorités émettrices des certificats qui forment le chemin, et ensuite procéder au mécanisme d'évaluation de chemins de certification.

Si le client demande l'évaluation d'un chemin du type incomplet, la procédure décrite ci-dessus est précédée par la récupération de la série de certificats permettant de compléter le chemin en question. Dans les deux cas, si le DAS dispose d'un meilleur chemin vis-à-vis des critères d'évaluation et des données accumulées pendant son opération (par exemple, l'utilisation de caches), il retourne ce chemin alternatif au client.

Les services du DAS sont réalisés à travers l'interaction avec le service de noms et/ou l'accès à la base de données du DAS de façon à pouvoir récupérer des certificats, des listes de révocation, vérifier les signatures des certificats, leur validité individuelle et, finalement, vérifier que la séquence de certificats obtenue n'est pas en contradiction avec la politique de sécurité du KMD. Ce service doit prendre en compte le mécanisme d'évaluation de chemins de certification décrit dans le Chapitre 13.

Nous concluons que les problèmes (1) et (2) ci-dessus sont résolus au moyen de deux fonctionnalités de base :

- la recherche de certificats et de listes de révocation de certificats,
- la consultation des politiques de certification, des critères locaux d'évaluation et des besoins de l'utilisateur en ce qui concerne les métriques d'évaluation, le modèle de certification, l'algorithme et la taille des clés publiques.

Dans la section suivante nous analyserons la façon dont le DAS réalise ses services vis-à-vis de l'environnement dans lequel il s'insère.

15.1.3. Analyse du Fonctionnement du DAS

Le DAS est censé offrir ses services aux utilisateurs du KMD qu'il sert. En contrepartie, il doit être noté que les certificats sont potentiellement répartis entre plusieurs KMD dont chacun peut être servi par un DAS. L'exécution d'une opération peut donc exiger des informations non disponibles dans le KMD du DAS en question.

Pour que le Service DAS soit indépendant des services de noms et pour garantir la complétude de notre travail, nous devons donner lieu à la conception d'une opération distribuée entre DAS. Un DAS doit alors réaliser ses services de la façon suivante :

- Si toute l'information nécessaire à l'exécution d'une opération est localisée dans le KMD servi par le DAS en question, ce dernier donnera suite à l'exécution de l'opération suivant la politique de distribution de clés et la topologie définie pour son domaine.
- Si l'information nécessaire à l'exécution d'une opération est répartie, l'opération demandera la coopération d'autre(s) DAS. Tel est le cas si les deux principaux qui désirent s'authentifier appartiennent à deux KMD différents.
- Chaque DAS doit effectuer les actions nécessaires pour compléter l'opération dans son domaine, et lorsque l'exécution coopérative de l'opération se fait nécessaire, le DAS doit solliciter d'autre(s) DAS pour la continuation de l'opération.

Un DAS doit donc être capable d'établir et de vérifier des chemins de certification intra-domaines et de communiquer avec d'autres DAS afin d'établir et de vérifier des chemins de certification inter-domaines. Pour que ce service puisse être réalisé, nous devons définir deux aspects :

- ♦ le modèle distribué des DAS.
- ♦ les protocoles d'accès et d'interaction entre DAS,

Ensuite, pour que les services du DAS soient fiables, non seulement une analyse des possibles menaces à ces services doit être faite, mais aussi des mécanismes de protection au service doivent être identifiés.

NOTA : Il doit être tout de même noté que notre travail ne dépend pas de l'opération distribuée entre DAS. Celle-ci peut être implémentée ou non, selon les besoins de l'infrastructure d'authentification. En contrepartie, l'opération répartie permet que chaque KMD ait la liberté de choisir le(s) service(s) de noms le plus appropriés à ses exigences, sans besoin de connectivité ou de convergence entre tous ces services.

15.2. Modèle de Fonctionnement du DAS

Le Service DAS, tel qu'il a été défini dans la Section 4.1.2, est modélisé comme un objet offrant un ensemble de services d'obtention de données pour l'authentification à ses utilisateurs, les clients DAS. Les clients DAS accèdent à ces services par des points d'accès. Les clients DAS peuvent être dotés d'un agent utilisateur ou d'une API (Application Programming Interface) qui facilitera leur interaction avec le service DAS.

Les certificats sont potentiellement distribués entre plusieurs KMD dont chacun est géré par un DAS. Toutefois, chaque DAS est chargé d'offrir ses services avec les données contenues dans son KMD. Dans le cas où toute l'information nécessaire à l'exécution d'une opération est localisée dans le KMD géré par le DAS en question, ce dernier donnera suite à l'exécution de l'opération suivant la politique de distribution de clés et la topologie définie pour son domaine.

Les principes qui régissent la répartition des clés publiques sont définis par la Politique de Gestion de Clés de chaque KMD. Les KMD peuvent être interconnectés pour permettre l'authentification des principaux appartenant à des KMD différents. Cela implique le besoin d'une architecture distribuée où les DAS coopèrent pour l'exécution des opérations mais chacun fonctionne de façon autonome.

Avant de définir l'opération distribuée des DAS, il semble intéressant de visualiser l'environnement distribué où les DAS sont placés. Cela nous permettra d'établir une correspondance entre les opérations de base d'un DAS et les opérations distribuées qu'elles peuvent déclencher.

15.2.1. Environnement Distribué des DAS

Comme mentionné ci-dessus, l'espace de clés publiques est distribué entre les KMD. La figure suivante illustre l'environnement distribué qui constitue le Service DAS. La modularité de notre approche permet l'indépendance et l'autonomie des domaines au niveau de la gestion de clés. Par conséquent, un DAS n'est pas supposé connaître la topologie globale de l'espace de clés mais seulement la configuration de son KMD. La liaison entre domaines est définie au moyen de la certification inter-domaines, i.e., l'existence de certificats et/ou certificats croisés entre CA appartenant à deux domaines distincts.

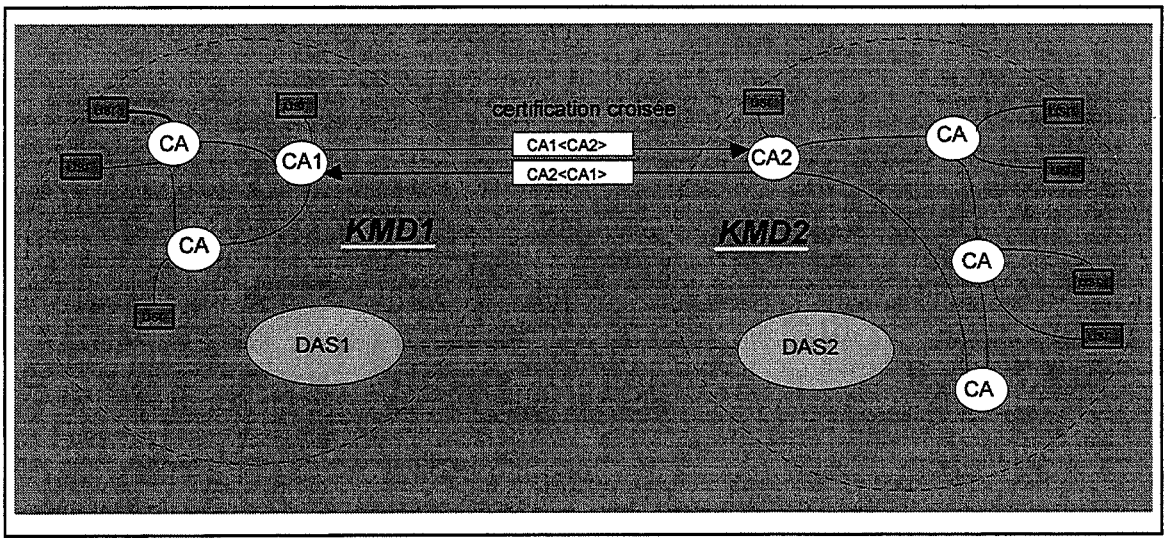

Figure 35. Environnement Distribué des DAS

La figure précedante illustre l'environnement reparti des DAS qui sera utilisé comme base de spécification des aspects repartis du Service DAS.

Afin d'assurer la clarté de notre exposition, nous introduisons les termes suivants :

- *CA d'interconnexion*

  Une autorité de certification, CA_1, est considérée une CA d'interconnexion d'un domaine de gestion de clés KMD_1 si $CA_1 \in KMD_1$ et il existe $CA_2 \in KMD_2$ ($KMD_1 \neq KMD_2$ et $CA_1 \neq CA_2$) tel que les certificats $CA_1<CA_2>$ et $CA_1<CA_2>$ existent.

- *KMD interconnectés*

  Deux KMD, KMD_1 et KMD_2, sont interconnectés s'il existe au moins une CA dans chaque KMD, $CA_1 \in KMD_1$ et $CA_2 \in KMD_2$, tel que les certificats $CA_1<CA_2>$ et $CA_2<CA_1>$ existent.

- *DAS adjacents*

  Si deux domaines KMD_1 et KMD_2 sont interconnectés et il existe deux DAS, DAS_1 et DAS_2, tels que le DAS_1 sert le KMD_1 et le DAS_2 sert le KMD_2, alors les DAS_1 et DAS_2 sont dits DAS adjacents.

Dans la section suivante nous établirons la correspondance entre les opérations de base d'un DAS et les opérations distribuées qu'en peuvent être déclenchées.

15.2.2. Opération Distribuée des DAS

Nous avions déjà identifié les deux problèmes principaux qu'un DAS doit résoudre, à savoir, la construction et l'évaluation d'un chemin. De plus, nous avons identifié deux catégories de chemin, à savoir, le chemin complet et le chemin incomplet. Dans le cas où toute l'information nécessaire à l'exécution d'une opération est localisée dans le KMD servi par le DAS en question, ce dernier donnera suite à l'exécution de l'opération suivant la politique de distribution de clés et la topologie définie pour son domaine. Nous devons alors analyser le comportement d'un DAS, lorsque l'information nécessaire à l'exécution d'une opération n'est pas localisée dans son KMD. Dans un tel contexte :

- La construction d'un chemin (complet ou incomplet) doit être acheminée vers le(s) DAS adjacent(s) pour la continuation de l'opération à partir de la CA d'interconnexion entre les domaines.

- La vérification d'un chemin complet doit déclencher, vers le(s) DAS adjacent(s), des requêtes de récupération de listes de révocation des CA appartenant à d'autres domaines.

- La vérification d'un chemin incomplet doit engendrer, vers le(s) DAS adjacent(s), des requêtes pour que ce chemin soit complété et des requêtes de récupération de listes de révocation des CA appartenant à d'autres domaines.

Il doit être noté que l'évaluation de chemins de certification ne peut pas faire partie de l'opération distribuée des DAS. Une telle approche impliquerait une attribution de confiance à tous les DAS qui coopèrent pour l'exécution d'une opération.

De plus, pour que la sécurité du mécanisme soit effective, toute information utilisée pour l'évaluation, à savoir, les listes de révocation, les politiques de certification et les critères locaux, doit être disponible à l'utilisateur final, i.e., le Client DAS.

Pour assurer les caractéristiques opérationnelles des DAS, il est nécessaire que chaque DAS gérant un KMD soit capable d'identifier et éventuellement d'interagir avec d'autres DAS gérant d'autres KMD et avec des services de noms. Cela exige l'analyse de la navigation entre l'ensemble des serveurs qui forment le service DAS et la définition d'un protocole d'interaction entre ces serveurs. Ces deux aspects seront abordés dans les sections suivantes.

La figure précédante représente le modèle reparti des DAS caractérisé par des KMD dont la liaison est définie en fonction de la certification inter-domaines. La figure suivante est une simplification de la figure précédante afin de représenter l'objet Service DAS comme constitué d'un ensemble d'un ou de plusieurs objets DAS.

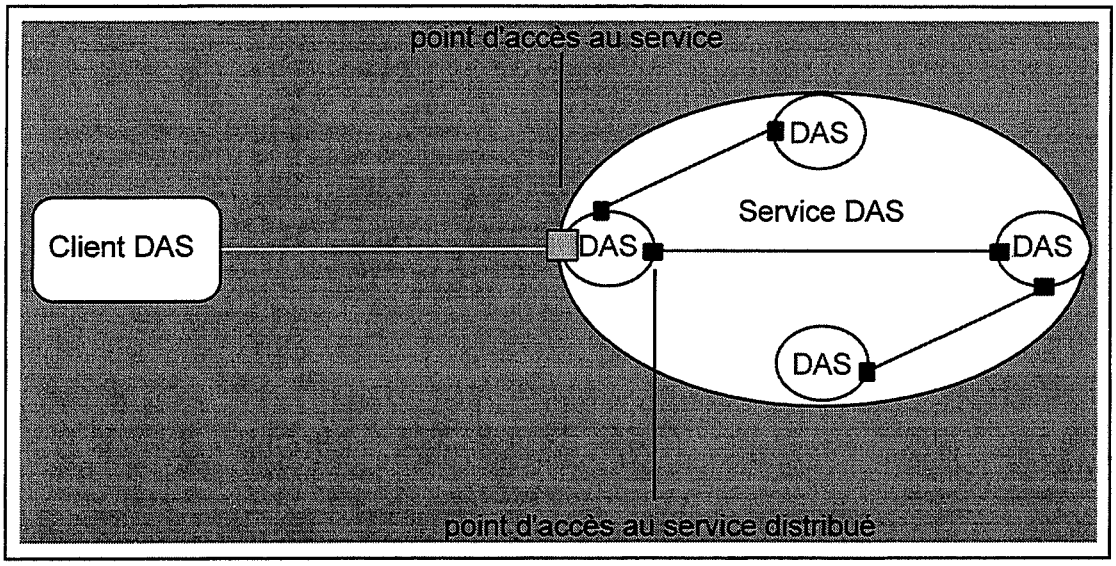

Figure 36. Modèle Distribué du Service DAS

Les objets DAS sont, de même que l'objet Service DAS, accessibles par leurs points d'accès visibles de l'extérieur. Conceptuellement, les points d'accès associés à un objet DAS sont de deux types : les points d'accès au service et les points d'accès au service distribué. Ces derniers permettent la communication entre DAS afin de pouvoir réaliser le Service DAS dans un environnement réparti.

15.2.3. Navigation dans l'Espace de Clés Publiques

Pour garantir le fonctionnement de chaque DAS dans son domaine et le comportement distribué de ces serveurs, il est nécessaire d'aborder deux aspects importants :

- **gestion de connaissance**

  Il s'agit de l'information nécessaire au fonctionnement d'un DAS dans son KMD et à l'interaction avec d'autres DAS.

- **navigation**

  Il s'agit des procédures qui définissent comment les DAS acheminent des opérations à travers le système distribué.

15.2.3.1. La Connaissance du DAS

La structure de connaissance d'un DAS doit permettre à celui-ci de déterminer si une certaine opération peut être effectuée en utilisant seulement les informations contenues dans son KMD, ou bien si une partie de cette opération doit être relayée vers un autre DAS. De plus, cette structure de connaissance doit permettre au DAS de déterminer le DAS vers lequel une certaine opération doit être relayée.

L'information nécessaire au fonctionnement d'un DAS dans son KMD sera nommée dans ce travail comme **connaissance interne**. L'information nécessaire à l'interaction avec d'autres DAS sera nommée comme **connaissance externe**. Les procédures d'évaluation de chemins de certification et de résolution de noms nécessaires au fonctionnement du DAS seront fondés sur ces informations de connaissance.

Le seul besoin de sécurité imposé à ces informations est l'intégrité de ces dernières car ces connaissances ne contiennent pas d'informations confidentielles. La Section 14.5.3 propose un mécanisme pour la maintenance de l'intégrité de la connaissance du DAS.

15.2.3.1.1. Connaissance interne

La structure de la connaissance interne du DAS doit garantir la réalisation des opérations dans son KMD et déterminer si l'exécution d'une opération nécessite la coopération d'autres DAS. Cela est assuré si les conditions suivantes sont satisfaites :

- le DAS est capable de déterminer :
  - si une CA donnée appartient à son KMD,
  - si un principal appartient à son KMD.
- le DAS conserve des informations spécifiant :
  - si un ou plusieurs serveurs de noms sont associés au KMD pour la stockage des données d'authentification appartenant aux entités du KMD,
  - les critères locaux et les métriques d'évaluation de certificats du KMD.

Par conséquent, conceptuellement la connaissance d'un DAS doit contenir les types d'informations suivantes :

- la KMP du KMD, qu'il s'agit de :
  - l'identification des CA qui appartiennent au KMD,
  - le(s) modèle(s) de confiance du KMD,
  - les critères locaux et les métriques du KMD pour l'évaluation des chemins de certification.
- Le choix du KMD concernant le stockage des données publiques pour l'authentification. Il s'agit des informations permettant au DAS de récupérer les données nécessaires à la prestation de son service, à savoir :
  - l'emplacement des certificats d'utilisateurs,

- l'emplacement des certificats, des politiques et de listes de révocation des CA.

Il est possible que certains KMD soient constitués de plusieurs portions disjointes de l'espace de noms. Dans un tel cas, une instance de ce type de référence doit comprendre :

- l'identification d'une portion de l'espace de noms,
- le point d'accès du serveur qui gère cette portion de l'espace de noms.

Quant à la détermination de l'emplacement d'un principal, le DAS peut déterminer si un principal appartient à son KMD en utilisant la méthode suivante : un principal appartient à un certain KMD si et seulement si un de ces certificats est émis par une CA appartenant à ce KMD.

Ayant comme principe que les entités sont identifiées par son nom d'annuaire, et que les certificats contiennent l'identification de leur émetteur, il s'avère qu'un accès au service de noms ou à la base du DAS est nécessaire pour déterminer si un principal appartient à son KMD.

Etant donné que notre méthode de description de politiques de certification permet d'identifier le modèle de confiance auquel une CA appartient, la connaissance interne d'un DAS peut être spécifiée au moyen de la structure ASN.1 suivante :

```
InternalKnowledge ::= SEQUENCE {
        kmp         KmpDescription,
        kmdData     KmdDataRetrieval
}
```

L'élément *kmp* décrit la politique de gestion des clés publiques du KMD et se définit de la façon suivante :

```
KmpDescription::= SEQUENCE {
        responsability          SET OF DistinguishedName,
        localCriteriaDomainPolicyDescription,
        weighting               CaClassesWeighting
}
```

Chaque champ de *KmpDescription* a la signification suivante :

- *responsability* : spécifie les noms d'annuaire des CA internes au KMD,
- *localCriteria* : spécifie les critères locaux du KMD vis-à-vis des modèles de confiance existants. La sémantique et la syntaxe de description de ces critères ont été présentées dans le Chapitre 13 (voir 13.4.1),
- *weighting* : spécifie les coefficients de pondération attribués aux classes de CA selon la politique de sécurité du KMD. Ces coefficients serviront au mécanisme d'évaluation de chemins de certification défini dans le Chapitre 13, où la sémantique et la syntaxe de ce champ ont été présentées (voir 13.4.2).

L'élément *kmdData* de la connaissance interne d'un DAS décrit l'emplacement des données publiques d'authentification des entités du KMD et se définit de la façon suivante :

```
KmdDataRetrieval ::= SEQUENCE {
        userDataRetrieval       UserDataRetrieval,
        caDataRetrieval         CaDataRetrieval
}
```

Chaque champ de *KmdDataRetrieval* a la signification suivante :

- *userDataRetrieval* : spécifie l'emplacement des données des utilisateurs finals du KMD,

```
UserDataRetrieval ::= UniqueChoice
```

- *caDataRetrieval* : spécifie l'emplacement des données des autorités de certification du KMD.

```
CaDataRetrieval ::= CHOICE {
        allCA  [0] UniqueChoice,
        eachCA     [1] SET OF EachCAChoice
}
```

La structure de connaissance proposée permet une flexibilité au niveau du choix de chaque CA. Il est possible de spécifier un choix différent d'emplacement de données pour chaque CA. Selon le choix du KMD, le DAS peut être le responsable du stockage de ces données. L'alternative *UniqueChoice* spécifie un seul emplacement de stockage de données.

```
UniqueChoice ::= SEQUENCE {
    type        [0] INTEGER { das(0), x500(1), solo(2), gopher(3), whois(4) },
    reference   [1] AccessPoint OPTIONAL
                            -- absent if "das" is chosen
    }
```

L'alternative *EachCAChoice* spécifie un emplacement de stockage de données différent pour chaque autorité de certification du KMD.

```
EachCAChoice ::= SEQUENCE {
        ca          SET OF DistinguishedName,
        choice      UniqueChoice
}
```

La section suivante décrit la procédure de routage des opérations dans l'environnement réparti des DAS.

15.2.3.2. La Procédure de Navigation des DAS

La procédure de navigation correspond aux étapes que les DAS doivent suivre pour relayer une opération vers le(s) DAS capable(s) de compléter cette opération.

La structure de connaissance d'un DAS, analysée jusqu'à maintenant, garantit qu'il peut déterminer si une certaine opération peut être complètement réalisée avec les données d'authentification internes à son KMD ou si une partie de cette opération doit être relayée vers un autre DAS. La question qui se pose est de savoir vers quel autre DAS une opération doit être

relayée pour qu'elle puisse être complétée. Pour répondre à cette question, les aspects suivants doivent être pris en compte :

(1) Les certificats sont potentiellement distribués parmi les KMD.

(2) Il n'existe pas d'imposition concernant la connectivité entre KMD, donc la connectivité globale entre tous les KMD n'est pas supposée. La connectivité entre KMD est définie par la certification inter-domaine, ce qui implique une attribution de confiance et/ou un accord bilatéral entre KMD.

(3) Un KMD n'est pas forcement défini par une ou plusieurs portions de l'espace de noms, mais par un ensemble de CA et les principaux certifiés par ces autorités. De plus, le modèle n'impose pas des contraintes de nommage aux composants des KMD. Celles-ci sont définies par les modèles de certification.

(4) Un DAS n'a pas la connaissance de la configuration globale des CA. Telle configuration est potentiellement mutable, surtout en ce qui concerne l'interconnexion entre KMD. Ainsi, un DAS ne connaît que son propre KMD.

(5) En utilisant la notation introduite dans la Section 14.2.1, la construction d'un chemin de certification entre deux utilisateurs appartenant à deux KMD distincts, KMD_a et KMD_b, est possible s'il existe une séquence de KMD adjacents

$$S = KMD_i\ KMD_{i+1}\ ...\ KMD_j,$$

telle que, KMD_a est adjacent à KMD_i et KMD_b est adjacent à KMD_j, ou vice-versa. Il s'ensuit que :

- la construction de chemins de certification doit suivre la certification inter-domaine et la procédure de navigation doit transférer une partie de la responsabilité au(x) DAS adjacent(s) pour que l'opération puisse être complétée.
- il est possible qu'il existe plusieurs chemins de certification entre deux utilisateurs.

Par conséquent, lorsqu'une opération ne peut pas être complétée dans le propre KMD, le DAS doit, *en principe*, relayer cette opération vers *tous* les DAS adjacents concernant l'opération. Le DAS initiateur chaîne la requête vers chaque DAS adjacent en ayant comme point de départ la CA d'interconnexion correspondante, i.e., la partie du chemin qui reste à résoudre. Cela constitue le *pas de navigation* du Service DAS.

Chaque DAS adjacent effectuera cette même procédure. Il donnera suite à l'exécution de l'opération dans son KMD. Si un DAS ne peut pas satisfaire entièrement la demande, i.e., si l'opération peut être complétée avec les informations contenues dans son KMD, il doit exécuter le pas de navigation vers ses DAS adjacents.

Cette procédure de navigation se répète jusqu'à ce que l'opération soit complétée. A ce moment, chaque DAS envoie son résultat au DAS qui lui a envoyé la requête, ce dernier fusionne ces résultats à ceux qu'il a pu obtenir dans son KMD pour l'envoyer au DAS expéditeur, et ainsi de suite jusqu'à ce que le résultat arrive au DAS de départ. Ce dernier doit, à son tour, livrer à son client tous les résultats obtenus. La figure suivante illustre un tel scénario.

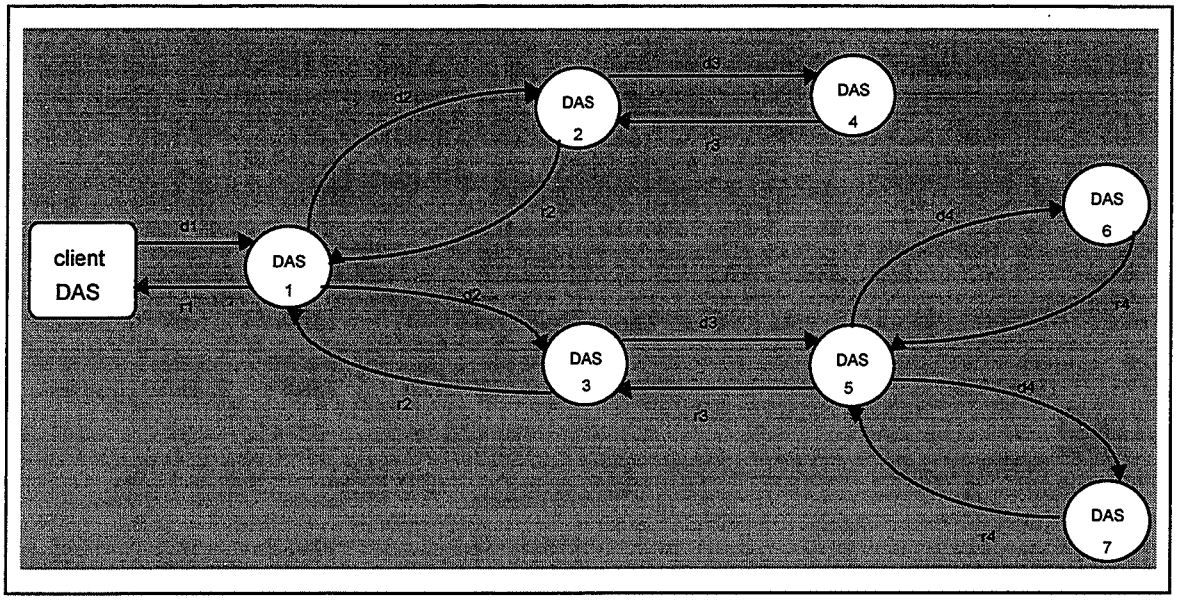

Figure 37. Navigation d'une Opération entre DAS

La section suivante identifiera les informations qu'un DAS doit retenir afin d'être capable de réaliser l'opération distribuée.

15.2.3.2.1. Connaissance externe

La connaissance externe d'un DAS doit garantir sa compétence de relayer la partie des opérations qui ne peut pas être effectuée dans son KMD. Cela assure que le DAS pourra toujours obtenir les données nécessaires à l'authentification entre deux utilisateurs appartenant à des KMD différents. Nous pouvons définir deux types de connaissance externe :

- **Connaissance externe minimale**

  Il s'agit de l'information minimale nécessaire à l'opération distribuée d'un DAS.

- **Connaissance externe ajoutée**

  Il s'agit des informations obtenues à travers l'opération distribuée des DAS et qui permet l'optimisation du service distribué.

La connaissance externe minimale consiste en un ensemble de références croisées permettant au DAS d'identifier un autre DAS vers lequel une certaine opération doit être relayée. Puisque l'interconnexion entre domaines est définie par la certification entre CA d'interconnexion, chaque instance de cette connaissance doit associer l'identification d'une CA d'interconnexion au DAS responsable du KMD auquel cette CA appartient. Ainsi, conceptuellement, la connaissance externe minimale d'un DAS doit contenir les types d'informations suivants :

- l'identification des CA d'interconnexion externes au KMD,
- le point d'accès du DAS qui sert au KMD auquel les CA ci-dessus appartiennent.

La connaissance externe minimale peut donc être spécifiée en ASN.1 de la façon suivante :

```
MinimalExternalKnowledge ::= SET OF ExternalReference
```

```
ExternalReference ::= SEQUENCE {
    dasAcess            AcessPoint,
    interconnexionCA    SET OF DistinguishedName }
```

15.2.3.2.2. Contrôle de la Navigation

Comme dans tout système distribué, la navigation entre DAS nécessite des mécanismes de contrôle de routage. Lorsque la procédure de navigation passe l'exécution d'une opération entre DAS adjacents, il est nécessaire de tenir compte de l'état d'exécution de l'opération pour éviter le bouclage et/ou le mauvais fonctionnement du système à cause des incohérences de la connaissance. L'incohérence des connaissances du DAS sera traitée dans la section suivante. L'état courant d'une opération correspond à la partie du chemin qui reste à construire.

Outre l'état courant d'une opération, un DAS a besoin de connaître tous les états antérieurs de cette opération. Il est donc nécessaire que les informations concernant la progression d'une opération soient propagées entre les DAS participants de l'exécution de cette opération pour qu'ils puissent prendre les décisions de navigation de façon correcte.

Ces informations doivent être enregistrées dans un élément du protocole d'interaction entre DAS. Pour cela, nous définirons l'élément de protocole **DasTrace**, qui servira à contrôler la navigation dans l'espace de clés publiques. Chaque DAS qui participe à l'exécution d'une opération doit ajouter à cet élément, son nom et la partie de opération qu'il a effectuée. L'élément DasTrace peut avoir la définition suivante :

```
DasTrace ::= SEQUENCE OF SEQUENCE {
                dasName         DistinguishedName,
                fromCA          DistinguishedName,
                toCA            DistinguishedName }
```

Où,

- *dasName* : indique le nom du DAS participant à l'opération,
- *fromCA* et *toCA* : indiquent respectivement, le point initial et le point final de l'opération effectuée par ce DAS.

En ce qui concerne le bouclage, il se produit si à un moment donné, l'opération revient à un état antérieur. Ceci ne veut pas dire qu'une opération ne peut pas être traitée plus d'une fois par un même DAS, mais signifie que ce DAS ne doit pas traiter plusieurs fois la même opération dans le même état. Deux stratégies doivent être définies pour parer aux bouclages :

- la prévention de bouclage,
- la détection de bouclage.

La prévention des bouclages nécessite qu'un DAS, immédiatement avant de transmettre une opération à un autre DAS, détermine si l'état de l'opération résultant des traitements du DAS apparaît dans la séquence d'états antérieurs enregistrés dans l'élément *DasTrace* de l'opération entrante initiale. La partie de l'opération effectuée par le DAS doit être ajoutée à l'élément *DasTrace*.

La détection de bouclage nécessite qu'un DAS détermine, lorsqu'il reçoit une opération entrante, si l'état courant de l'opération figure dans la séquence d'états enregistrés dans l'élément *DasTrace* de cette opération. Si c'est le cas, l'opération est en train de boucler et une indication d'erreur doit être renvoyée. Sinon, le DAS continue le traitement de l'opération selon les procédures normales.

15.2.3.2.3. Connaissance externe ajoutée

La connaissance externe minimale d'un DAS spécifie le minimum d'information qu'un DAS doit conserver de façon à réussir la procédure de navigation. Toutefois, cela ne spécifie pas la connaissance nécessaire pour une navigation optimale. La navigation utilisant la connaissance externe minimale comprend tous les DAS adjacents même dans les cas où ils ne sont pas tous nécessaires à l'exécution d'une opération distribuée.

En général, la navigation serait plus efficace si un DAS pouvait retenir des références concernant seulement les DAS adjacents nécessaires pour achever un certain KMD. Ces références constituent la *connaissance externe ajoutée* car elles ne sont pas strictement exigées à la navigation. En effet, la procédure de navigation opère comme décrite auparavant à l'exception des DAS adjacents qui ne sont pas nécessaires à l'exécution de l'opération.

L'utilisation de la connaissance externe ajoutée est particulièrement avantageuse si un KMD est interconnecté à un grand nombre de KMD. Cette connaissance peut être obtenue à partir des résultats des opérations distribuées des DAS, à travers le retour de l'élément *DasTrace*, défini dans le paragraphe précédant.

15.3. Utilisation de Caches

Comme nous l'avons déjà mentioné, les mécanismes de sécurité sont déjà, par leur nature, coûteux au niveau du calcul. Pour éviter d'augmenter le coût de ces mécanismes, on doit chercher à minimiser le trafic généré pour l'obtention des données nécessaires à l'authentification.

Afin d'améliorer la performance des opérations du DAS, de minimiser le nombre d'accès au service de noms et le traitement redondant d'information, l'utilisation de caches doit être envisagée. En fait, deux types de caches peuvent être considérés :

- les caches des services de noms,
- un système de caches propre au DAS.

15.3.1. Les Caches du Service de Noms

Au premier abord, ce type de cache pourrait être considéré utile car il ne demande aucun effort de gestion de la part du DAS. Par contre, le DAS n'est pas assuré que les caches du service de noms ne deviennent pas arbitrairement obsolètes car il n'y a pas de garantie sur la mise à jour de ces caches.

En ce qui concerne les certificats, l'utilisation de ce type de cache ne constituerait pas une atteinte à la fiabilité du Service DAS car ces éléments contiennent en eux leur durée de vie. Dans le cas

des listes de révocation, l'utilisation d'information obsolète peut entraîner des conséquences très graves car il se peut faire qu'une entité dont la clé privée a fait l'objet de compromission réussisse à s'authentifier alors que cette procédure devrait échouer.

15.3.2. Les Caches du DAS

La maintenance d'un cache propre au DAS permettrait de minimiser les problèmes de performance du DAS inhérents à sa fonctionnalité, en réduisant la dépendance vis-à-vis d'une source externe d'information, sans pour autant nuire à la fiabilité des services de sécurité.

15.3.2.1. Principes

L'utilisation de caches peut être implémentée sans que cela n'altère la sémantique des opérations du DAS. Par exemple, ces caches pourraient "rappeler" au DAS la clé publique qui a vérifié certaines signatures auparavant. Cela est très utile dans le cadre de la vérification de certificats qui sont normalement des structures à longue durée de validité.

En contrepartie, il est inacceptable que des procédures d'authentification échouent ou réussissent de façon erronée à cause de l'utilisation d'information obsolète. Lorsqu'une procédure d'authentification échoue de façon erronée, ceci amène un refus de service. En cas extrême, un utilisateur pourra toujours réitérer cette procédure en espérant que les informations concernées soient mises à jour. Par contre, la réussite d'une procédure d'authentification qui aurait dû échouer peut entraîner des conséquences beaucoup plus graves.

Par conséquent, un principe fondamental de l'utilisation de caches pour l'authentification est qu'elle ne doit jamais éliminer la consultation des sources originelles des données. La gestion de caches doit être soigneusement élaborée de façon à ce que des processus d'authentification ne réussissent ni n'échouent à tort.

De plus, l'utilisation de caches ne doit pas être transparente aux utilisateurs. A chaque accès au DAS, l'utilisateur doit spécifier s'il permet l'utilisation de caches. Cette décision doit dépendre de la politique de sécurité en vigueur et de l'objectif de la procédure d'authentification.

Une utilisation constante des services du DAS interdisant l'emploi de caches peut probablement amener à des conséquences pénalisantes au niveau de la performance mais cela n'affectera pas l'exactitude et la sécurité des opérations.

15.3.2.2. Besoins

L'objectif de l'utilisation de caches dans les opérations du DAS est de minimiser le traitement redondant d'information, le nombre d'accès à des serveurs de noms et la dépendance vis-à-vis de la disponibilité de ces serveurs, afin d'améliorer la performance du Service DAS. En particulier, notre expérience durant le projet Password nous a fait constater qu'il est assez décevant pour un utilisateur, de ne pas pouvoir donner suite à une procédure d'authentification à cause d'une indisponibilité du service de noms.

Avant de définir la structure des caches du DAS, nous allons analyser les besoins de tels caches vis-à-vis des données manipulées par le DAS. L'ensemble de données manipulées dans les opérations du DAS est constitué des éléments suivants :

- certificats,
- politiques associées aux certificats,
- listes de révocation de certificats (CRL),
- chemins de certification.

Il est important que les mêmes certificats ne soient pas récupérés à chaque opération réalisée par un DAS. Il en est de même pour les politiques de certification (voir 13.3), car celles-ci sont strictement liées aux certificats et ont une durée de vie identique aux certificats correspondants.

En ce qui concerne les listes de révocation, le cache de ces informations est intéressant dans la mesure où il permet de réduire la dépendance et le nombre d'accès au service de noms. Toutefois, il est important que cela ne permette pas que des procédures d'authentification réussissent par erreur à cause de l'utilisation d'une liste obsolète.

Comme la construction et l'évaluation de chemins de certification sont des opérations coûteuses, il serait intéressant de maintenir un cache de chemins de certification. Cela éviterait que le DAS ait à établir et à évaluer le même chemin plusieurs fois. Cependant, nous remarquons les aspects suivants :

(1) Les chemins entre des entités de deux domaines de certification différents peuvent être identiques (selon les métriques d'évaluation), hormis les extrémités. La figure suivante illustre un tel scénario.

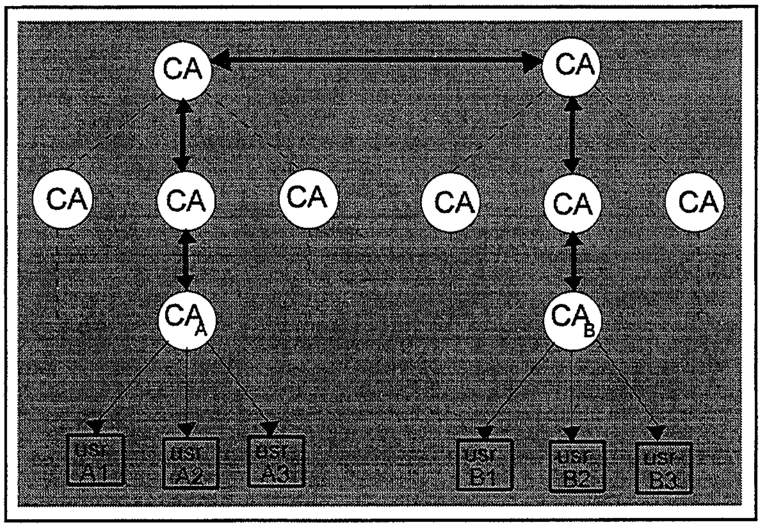

Figure 38. Exemples de Chemins de Certification

(2) Les durées de validité des certificats qui composent un chemin de certification sont indépendantes les unes des autres.

En conséquence de (1), l'adoption d'un tel cache comme cache de base du DAS impliquerait le stockage d'information redondante. En conséquence de (2), l'adoption d'un tel cache comme cache de base exigerait un effort de gestion non négligeable. De la sorte, nous concluons que :

- un tel cache ne doit pas constituer le cache de base du DAS mais un *cache auxiliaire*,
- ce cache ne doit pas contenir des chemins de certification entre utilisateurs finals mais entre les CA d'extrémité (par exemple, les CA_A et CA_B de la figure précédante),
- ce cache ne doit pas contenir des certificats mais des références à des certificats présents (ou non) dans le cache principal du DAS.

Les références mentionnées ci-dessus doivent identifier un certificat dans un chemin de certification. Ces références peuvent être telles que chacune de ces instances contient le numéro de série du certificat (SN), le nom du propriétaire (S) et en option, le nom de l'émetteur (I) d'un certificat, formant un "identificateur de certificat" (*cert-id*) ayant le format suivant :

$$\text{cert-id} ::= < SN, S [, I] >$$

Une séquence de ces identificateurs de certificats forme un "indicateur de chemin de certification" (*cp-indic*) ayant le format suivant :

$$\text{cp-indic} ::= < SN_1, S_1, I_1 > < SN_2, S_2 > < SN_3, S_3 > \ldots < SN_n, S_n >$$

Dans l'indicateur de chemin de certification, le nom de l'émetteur du certificat n'apparaît que dans le premier élément de la séquence car le propriétaire d'un certificat de la séquence est forcément l'émetteur du certificat suivant.

Puisque le DAS évalue chaque chemin de certification construit, chaque *cp-indic* peut être associé à la métrique, résultat et date d'évaluation du chemin. Dans le cas où un utilisateur demande l'évaluation d'un chemin de certification, le DAS peut aussi stocker dans ce cache les indicateurs de chemins de certification valides transmis par cet utilisateur.

En supposant que la révocation d'un certificat de CA est un événement rare, les entrées du cache auxiliaire du DAS auront une longue durée de vie. Seulement les numéros de série et les données d'évaluation doivent changer.

La propriété de "localisation de référence" (*locality of reference*) établit que le cache de données dans un environnement distribué augmentera la performance de la procédure de résolution seulement si les requêtes correspondantes présentent un haut "degré de localisation" [Smid81]. Cette notion de localisation peut être temporelle ou spatiale.

La localisation temporelle (1) se présente s'il existe une haute probabilité de réutilisation des données. La localisation spatiale (2) se présente s'il existe une haute probabilité de référence à des données de localisation proche (*neighboring data*).

En analysant ces principes dans le contexte de l'obtention de données pour l'authentification, nous constatons que (1) s'applique fortement à la construction de chemins de certification. Nous avons vu ci-dessus que les chemins de certification entre entités de deux domaines de certification sont identiques à l'exception des extrémités.

Il en est de même pour la propriété (2) puisque la construction de chemins de certification consiste à trouver un trajet orienté (une séquence de noeuds successifs) dans un graphe de

certification. Il doit être noté que ce raisonnement se réfère à la distribution de CA dans l'espace de clés et non dans l'espace de noms.

15.3.2.3. Gestion des Caches du DAS

Nous avons identifié le besoin de deux types de caches : des caches principaux contenant les données proprement dites et un cache auxiliaire contenant des indicateurs de chemin de certification. A ce stade nous devons analyser la gestion et le comportement d'un DAS vis-à-vis des caches afin d'être capables de définir la structure de ces caches.

Lorsque le DAS reçoit une requête, celui-ci vérifie d'abord si les données concernées sont présentes dans ses caches. Un "succès" (*hit*) se produit lorsqu'un élément nécessaire se trouve déjà dans les caches. Un "défaut" (*miss*) se produit lorsqu'un élément nécessaire ne se trouve pas dans les caches. Lors d'un défaut, le DAS doit procéder à son opération normalement et ensuite mettre les caches à jour avec les informations manquantes.

En particulier, s'il s'agit d'une demande de construction d'un chemin de certification et que le certificat du point initial du chemin n'est ni dans le cache ni fourni par le client, le DAS n'est pas capable de déterminer le nom de l'émetteur de ce certificat. Ainsi, la requête doit poursuivre la procédure normale.

S'il s'agit d'une demande d'évaluation d'un chemin, le DAS vérifie s'il dispose des données nécessaires à cette évaluation (listes récentes et politiques). Le DAS vérifie également s'il dispose d'un meilleur chemin vis-à-vis des métriques d'évaluation du client. La requête poursuit la procédure normale pour les informations manquantes dans les caches.

Si l'on définit le "taux de défaut" (*miss ratio*) comme la probabilité d'un élément nécessaire n'être pas présent dans les caches, alors plus la taille du cache augmente, moins élevé sera le taux de défaut. Ainsi, le DAS doit toujours mettre les caches à jour avec des informations valides récupérées lors de l'exécution d'une requête. De même, le DAS doit supprimer des caches les informations qui ne peuvent plus être considérées comme valides.

Normalement, les entrées d'un cache sont associées à un paramètre indiquant le temps de validité de cette entrée (*time-to-live* ou TTL). Dans le cas des données pour l'authentification cela n'est pas toujours nécessaire :

- Dans le cas des certificats, ceux-ci contiennent leur date d'expiration. A partir de cette date un certificat n'est plus valide et doit donc être supprimé du cache. La durée de validité des politiques est la même que celle du certificat correspondant.

  Un certificat (et sa politique) doit aussi être supprimé lorsqu'il est révoqué. Ainsi, lorsque le DAS récupère une nouvelle liste de révocation, celui-ci doit rechercher les certificats mentionnés et les politiques correspondantes afin de les supprimer du cache.

- Dans le cas particulier du cache d'indicateur de chemins de certification, une référence n'est remplacée que lorsque le DAS obtient un nouveau chemin (entre les mêmes points initial et final) ayant un meilleur résultat d'évaluation. Par conséquent, ce cache n'exige pas d'effort de gestion.

Les entrées de ce cache peuvent toujours donner une idée des entités composantes d'un chemin.

Dans le cas des CRL, bien que ces structures indiquent la date de la prochaine mise à jour, une CRL peut être remplacée avant cette date si des certificats sont révoqués entre-temps. Ainsi, les entrées du cache contenant des CRL doivent indiquer le TTL prévu pour ces structures.

L'estimation du TTL des listes de révocation doit prendre en compte le niveau de sécurité souhaité dans le domaine. Ces TTL peuvent être différentes selon le type de CA en question. Par exemple, un domaine peut attribuer un TTL plus court pour des CRL de CA désignées à la certification d'utilisateurs finals, puisque la révocation des certificats d'utilisateur doit être un événement assez fréquent. En contrepartie, la révocation d'un certificat de CA doit être un événement plus rare.

Il doit être noté que la taille des caches peut être en pratique limitée car cela exige une disponibilité de mémoire/espace-disque. Par conséquent, la gestion des caches doit prévoir une politique de remplacement des entrées. La littérature décrit plusieurs algorithmes de remplacement dont nous pouvons considérer :

- LRU (*least recenlty used*) - supprime l'entrée moins récemment utilisée,
- RAND (*random*) - supprime une entrée choisie aléatoirement,
- FREQ (*frequency count*) - supprime l'entrée moins utilisée,
- OPT (*theoretically optimal*) - supprime l'entrée qui sera référencée le plus tard.

Dans notre approche, le meilleur algorithme de remplacement est celui qui procure le taux de défaut le plus réduit avec un moindre effort de gestion, tout en respectant les besoins des utilisateurs.

L'algorithme OPT exige une prévision du futur et donc ne peut pas être implementé dans un système réel. Cependant, cet algorithme peut être considéré comme base d'évaluation d'autres algorithmes. L'algorithme RAND n'exige pas d'effort de gestion mais ne garantit pas un taux de défaut réduit par rapport aux autres algorithmes car il ne reflète pas les besoins des utilisateurs du DAS.

L'algorithme LRU a montré qu'il pouvait procurer le taux de défaut le plus réduit dans d'autres contextes [BA93, RD90] mais cet algorithme ne reflète pas forcément les besoins des utilisateurs du DAS. En contrepartie, l'algorithme FREQ paraît répondre mieux aux besoins des utilisateurs du DAS.

Des études d'évaluation d'algorithmes de remplacement dans le cadre de l'annuaire [BA93], ont montré que l'algorithme FREQ présente un taux de défaut comparable à celui obtenu avec LRU pour un cache de petite taille (une différence de 7.4 % pour un cache de 20 entrées) et des résultats très approximatifs au fur et à mesure que la taille du cache augmente (une différence de 0.5 % pour un cache de 200 entrées). Au niveau de la gestion, cet algorithme n'exige que la présence d'un compteur d'utilisations pour chaque élément du cache. Par conséquent, nous proposons l'utilisation de l'algorithme FREQ pour le remplacement des éléments des caches du DAS. Bien entendu, chaque domaine peut choisir la politique la plus adaptée à ses besoins car cela n'affecte pas l'opération distribuée des DAS.

Quel que soit l'algorithme de remplacement choisi, la gestion des caches du DAS doit prendre en compte le coût d'acquisition des données afin d'améliorer d'avantage la performance de l'opération du DAS. Ce coût d'acquisition peut être mesuré en termes de temps et du nombre de DAS nécessaires à l'exécution d'une opération.

15.3.2.4. Structure des Caches du DAS

Nous devons maintenant définir la structure des caches d'un DAS. Les caches principaux doivent contenir des certificats, des politiques, des listes de révocation et des données de contrôle qui permettront la gestion de ces caches. Ces données de contrôle doivent déterminer :

- le nombre d'utilisations d'un élément,
- le coût d'acquisition d'un élément,
- la date de la dernière utilisation d'un élément.

A cause des problèmes de sécurité mentionnés dans 14.3.2.1, la gestion des caches de listes de révocation ne peut pas être la même que celle des certificats et des politiques. D'autres études [BA93] montrent que le découpage du cache en un nombre restreint de caches indépendants, chacun étant associé à un type différent de gestion, réduit d'avantage le temps de traitement des requêtes[31]. Par conséquent, nous proposons la définition de deux caches principaux : l'un pour les certificats/politiques et l'un pour les listes de révocation.

La structure des caches du DAS est telle que chaque entrée contient des attributs d'authentification appartenant à une entité donnée. Ainsi, le DAS crée une nouvelle entrée pour chaque nouvelle entité pour laquelle il a obtenu des attributs d'authentification.

Puisque dans notre approche les entités sont identifiées par leur nom d'annuaire, les entrées des caches du DAS peuvent être représentées par des "tableaux de hash" de noms d'annuaire maintenus dans une zone de mémoire réservée au DAS. Chaque entrée de ce tableau pointe vers l'entrée réelle du cache contenant des attributs d'authentification d'une entité. Les cas de collision peuvent être traités par la technique de "résolution de collision avec chaînage" définie par Knuth [Knu93].

Nous utiliserons l'ASN.1 pour la définition de la structure des caches pour assurer la clarté de la description.

Puisqu'une entité peut avoir divers certificats, chaque entrée du cache de certificats et de politiques a la définition suivante :

```
CertificateCacheEntry :: = SET OF {
        certificate  [0] Certificate,
        policy       [1] Policy OPTIONAL,
        controls     [2] SEQUENCE {
                              n          [0] INTEGER,
```

[31] Il peut être aussi envisagé la partition des caches vis-à-vis des coûts associés à l'acquisition des données. Des études montrent [BA93] que cette approche permet d'augmenter la performance des opérations.

```
        c       [1] INTEGER,
        last    [2] UTCTime,
        cp      [3] BOOLEAN OPTIONAL } }
```

Où, les champs de l'élément *controls* sont interprétés de la façon suivante :

- *n* indique le nombre d'utilisations du certificat,
- *c* indique le coût d'acquisition du certificat,
- *last* indique la date de la dernière utilisation du certificat.
- *cp* indique si le cache auxiliaire contient des indicateurs de chemin de certification ayant ce certificat comme point initial. Cet élément est absent dans une entrée d'utilisateur final car ce dernier n'a de signification que pour les entrées des entités autorisées à certifier[32].

L'élément *policy* est optionnel car une même politique peut être associée à plus d'un certificat (voir 13.3.2.4).

La structure de chaque entrée du cache de listes de révocation est la suivante :

```
CrlCacheEntry :: = SEQUENCE {
    crl         PemCrl,
    controls    SEQUENCE {
                    n       INTEGER,
                    c       INTEGER,
                    last    UTCTime,
                    ttl     INTEGER } }
```

Puisqu'une même CA peut être le point initial de plusieurs chemins de certification, la structure de chaque entrée du cache auxiliaire a la définition suivante :

```
AuxiliarCacheEntry :: = SET OF {
    cp -indic   SEQUENCE OF { – cert-id
                    serialNumber        INTEGER,
                    subject     DistinguishedName,
                    issuer      DistinguishedName OPTIONAL},
    eval-data   SET {
                    metric  Metrics,
                    value   Evaluation Result,
                    date        UTCTime }
}
```

Chaque indicateur de chemin de certification (*cp-indic*) est associé aux données d'évaluation du chemin (*eval-data*), à savoir la métrique utilisée pour l'évaluation (*metric*), le résultat d'évaluation (*value*) et la date d'évaluation (*date*). Le mécanisme et la syntaxe des métriques et des résultats d'évaluation ont été presentés dans 13.4.2.

---

[32]L'élément *policy* permet de determiner si un certificat appartient à un utilisateur final ou à une CA.

15.4. Les Protocoles

Dans cette section nous décrirons la méthode d'accès et d'interaction du Service DAS.

15.4.1. Accès au DAS

Les DAS fournissent des services à ses utilisateurs au moyen du Protocole d'Accès au DAS (**DASAP** - ***Data Authentication Service Access Protocol***). Ce protocole représente donc, le point d'accès aux opérations du DAS. Ces opérations doivent récupérer une série ordonnée de clés publiques nécessaires à l'authentification d'un principal.

Cela est réalisé à travers l'interaction avec le service de noms et éventuellement avec d'autres DAS pour la récupération des certificats, la vérification de leur signature, de leur validité individuelle et la vérification que la séquence de certificats obtenue n'est pas en contradiction avec la politique de sécurité du KMD. Cette vérification doit prendre en compte le mécanisme d'évaluation de chemins de certification décrit dans le Chapitre 13.

Le modèle général adopté pour ce protocole est celui des clients réalisant des opérations sur des serveurs. Dans ce modèle, cela est achevé à travers l'action du client qui transmet une requête au serveur, spécifiant l'opération qui doit être effectuée par ce dernier. Le serveur est alors responsable de la collecte et du traitement de l'information nécessaire à la réalisation de l'opération. Après la réalisation de l'opération, le DAS doit retourner une réponse au client contenant les résultats ou les erreurs obtenus.

Bien que les DAS soient supposés retourner une réponse à toute requête d'un client, ce modèle n'implique un comportement synchrone ni de la part des serveurs ni de la part des clients. Les requêtes et les réponses à plusieurs opérations peuvent être échangées dans un ordre quelconque, pourvu que les clients reçoivent effectivement une réponse (résultat ou erreur) à toute demande envoyée au DAS.

L'exigence de base de ce protocole est la minimisation des coûts associés à l'acquisition, à la construction et à l'évaluation des chemins de certification nécessaires à l'authentification, pour faciliter un déploiement largement répandu de mécanismes de sécurité dans le contexte des systèmes ouverts. De plus, ce protocole doit répondre aux besoins des utilisateurs et des mécanismes d'authentification existants. Nous avons déjà décrit l'existence des besoins réels d'un tel service, les types de données que le DAS doit manipuler et les services qu'il doit fournir.

En principe, l'identification fournie comme entrée pour l'obtention des services du DAS suit l'identification des entités dans la structure certificat : le nom d'annuaire ou DN (**D**istinguished **N**ame). Toutefois, pour répondre aux besoins des utilisateurs de communautés diverses, on peut envisager l'utilisation d'autres attributs de recherche que le nom d'annuaire, par exemple l'adresse de messagerie X.400 ou basés sur les noms DNS, voire des numéros de téléphone, etc. Une autre évolution souhaitable est de permettre la manipulation d'un type quelconque de certificat, par exemple le format défini par PKCS #6.

Eléments du Protocole

Le protocole d'accès au DAS est constitué d'un seul message à travers lequel la fonctionnalité du Service DAS est offerte, à savoir, **DasMsg**. Ce message encapsule tous les éléments du protocole d'accès au DAS et permet à un client de demander la construction et l'évaluation de chemins de certification ou la récupération des certificats et des listes de révocation. La définition de ce message est la suivante :

```
DasMsg ::= SEQUENCE {
    dasOpId         [0] INTEGER,
    dasipTrace [1] DasTrace OPTIONAL, - - not used in DASAP
    dasOp           [2] CHOICE {
                            dasRequest      [0] DasRequest,
                            dasResult       [1] DasResult,
                            dasError        [2] DasError }
}
```

La fonction de *DasMsg* est d'offrir une enveloppe contenant les éléments communs nécessaires à tous les échanges du protocole d'accès au DAS. A présent, ce protocole dispose de deux éléments communs, à savoir, **dasOpId** et **dasOp**. Ces éléments sont décrits dans les paragraphes suivants.

L'élément *dasOpId* constitue un identificateur d'opération et doit être présent dans tout échange de données entre le client et le DAS. Le client doit fournir un identificateur pour chaque requête envoyée au DAS. La valeur de cet identificateur doit être différente de toute autre valeur d'identificateur de requête en cours. Les identificateurs seront renvoyés au client dans les réponses correspondantes retournées par le DAS.

L'élément *dasOp* implémente le modèle du protocole d'accès au DAS. Cet élément possède trois composants mutuellement exclusifs, à savoir, **dasRequest**, **dasResult** et **dasError**, qui seront décrits dans les paragraphes suivants.

L'élément *dasRequest* permet à un client d'envoyer sa requête de récupération ou d'évaluation au DAS. Cet élément a la définition suivante :

```
DasRequest ::= SEQUENCE {
    dasServiceFlags     [0] DasServiceFlags,
    dasTarget           [1] SEQUENCE OF DistinguishedName OPTIONAL,
    dasClientProposition [2] DasCertificates        OPTIONAL,
    certification_model [3] OBJECT IDENTIFIER       OPTIONAL,
    requested_key       [4] AlgorithmIdentifier     OPTIONAL,
    metrics             [5] Metrics                 OPTIONAL }
```

Le composant *dasServiceFlags* sert à fournir des indications nécessaires à l'exécution de l'opération. Cet élément est présent dans toute requête envoyée au DAS. Sa définition est la suivante :

```
DasServiceFlags ::= BIT STRING {
        forward_path            (0),
        reverse_path            (1),
        crl                     (2),
        policies                (3),
        dontUseCertCache        (4),
        dontUseCrlCache         (5)}
```

Où,

(i) **forward_path** indique que le client souhaite obtenir le chemin de certification A → B entre deux principaux A et B,

(ii) **reverse_path** indique que le client souhaite obtenir le chemin de certification B → A entre deux principaux A et B,

(iii) **crl** indique que les listes de révocation doivent être retournées,

(iv) **policies** indique que les politiques associées aux certificats doivent être retournées,

(v) **dontUseCertCache** indique que le DAS ne doit pas utiliser des informations du cache de certificats pour l'exécution de l'opération.

(vi) **dontUseCrlCache** indique que le DAS ne doit pas utiliser des informations du cache de listes de révocation pour l'exécution de l'opération.

Si (i), (ii) et (iii) ont la valeur zéro, le DAS assume que la requête constitue une simple demande de récupération des certificats appartenant aux principaux indiqués dans le composant *dasTarget*. De même, si (i) et (ii) ont la valeur zéro et (iii) a la valeur un, le DAS interprète la requête comme une simple demande de récupération des listes de révocation appartenant aux principaux indiqués dans le composant *dasTarget*.

Le composant *dasTarget* permet au client de spécifier les noms des principaux cibles de l'opération. Lorsqu'une requête constitue une demande de construction de chemin de certification, la séquence est composée des noms des deux principaux dont le premier correspond au point initial, et le deuxième correspond au point final du chemin de certification.

Lorsqu'une requête constitue une demande d'évaluation de chemin de certification, la séquence est constituée d'un seul élément, à savoir le nom du principal vers lequel le chemin doit être vérifié. Lorsqu'une requête constitue une demande de récupération de clés publiques ou de listes de révocation, un nombre quelconque de noms peut être spécifié.

Le composant *dasClientProposition* a la définition suivante :

```
DasCertificates ::= SEQUENCE OF Certificate
```

Ce composant peut avoir deux fonctions différentes. Si la requête constitue une demande d'évaluation, ce composant doit contenir la chaîne de certificats devant être évaluée. Si la requête constitue une demande de construction de chemin de certification, ce composant peut indiquer le certificat devant être le point initial du chemin. Ce composant est absent si la requête constitue une demande de récupération de listes de révocation ou de clés publiques.

Le composant *certification_model* permet au client d'indiquer au DAS le modèle de certification selon lequel le chemin doit être établi et/ou évalué. Les modèles de certification sont identifiés au moyen des identificateurs d'objet, conforme décrit dans la Section 13.3.2.1. Ce composant est absent si la requête constitue une demande de récupération de listes de révocation.

Le composant *requested_key* permet au client de choisir l'algorithme et/ou la taille minimale de la clé contenue dans le certificat du point final du chemin (*userCertificate* d'un *CertificationPath*). Cela est utile lorsque le client souhaite authentifier une clé publique pour un "rôle" spécifique ; par exemple, une clé de plus haute assurance pour l'envoi d'un message secret. Ce composant peut être spécifié dans les requêtes de construction de chemin de certification, ou de récupération de certificats.

Le composant *metrics* permet au client d'indiquer au DAS les métriques qui doivent être utilisées pour la construction et l'évaluation d'un chemin de certification. Ce composant peut être spécifié dans les requêtes de construction ou d'évaluation de chemin de certification. La définition, les semantiques des métriques ainsi que le mécanisme d'évaluation ont été presentés dans le Chapitre 13.

Le tableau ci-dessous résume la correspondance entre les services offerts et les valeurs des composants de l'élément *DasRequest*.

	serviceFlags	target	proposition	key	model	metrics
construction	forward : oui* reverse : oui* crl : opt policies : opt noCertCache: opt noCrlCache: opt	2	1 certificat max.	opt	opt	opt
évaluation	forward: non reverse : non crl : opt policies : opt noCertCache: opt noCrlCache: opt	1	chemin	absent	opt	opt
certificats	forward : non reverse : non crl : non policies : opt noCertCache: opt noCrlCache: non	undef.	absent	opt	opt	absent

* au moins l'un des deux bits doit être actif.

listes	forward : non reverse : non crl : oui policies : opt noCertCache: opt noCrlCache: opt	undef.	absent	absent	absent	absent

L'élément **DasResult** permet au DAS d'envoyer au client le résultat d'une requête. Cet élément a la définition suivante :

```
DasResult ::= SEQUENCE {
    forward_path    [0] DasResultingKeys    OPTIONAL,
    reverse_path    [1] DasResultingKeys    OPTIONAL,
    crls            [2] DasResultingCrls    OPTIONAL
    – at least one of these components must be present
}
```

Le composant *evaluation* permet au DAS d'indiquer le résultat d'évaluation d'un chemin en termes de niveau d'assurance. La syntaxe d'un resultat d'évaluation a été défini dans 2.4.2.

Les composants *forward_path* et *reverse_path* ont la définition suivante :

```
DasResultingKeys ::= SEQUENCE OF {
    path            [0] DasCertificates     OPTIONAL,
    policies        [1] DasPolicies         OPTIONAL,
    evaluation      [2] EvaluationResult    OPTIONAL
    – at least one of these components must be present
}
```

Où,

- *path* contient un chemin de certification,
- *policies* contient des politiques associées à des certificats et a la définition suivante :

  DasPolicies ::= SEQUENCE OF Policy

  La syntaxe d'une politique de certification (*Policy*) a été définie dans 2.3.2.
- *evaluation* permet au DAS d'indiquer le résultat d'évaluation d'un chemin. La syntaxe d'un resultat d'évaluation a été définie dans 2.4.2.

Le composant *forward_path* est utilisé par le DAS pour envoyer au client trois types de résultats différents, selon la requête du client :

- Si la requête constitue une demande d'établissement de chemin direct entre deux principaux, le composant (*path*) est utilisé pour transmettre un tel résultat,
- Si la requête comprend l'évaluation d'un chemin, le composant *evaluation* est utilisé pour transmettre un tel résultat. Si le DAS dispose d'un meilleur chemin (du point de vue des critères d'évaluation et des métriques) capable de satisfaire la demande du client, le DAS transmet ce chemin au client à travers le composant *path*,

- Si la requête constitue une demande de récupération de clés publiques (certificats), le composant *path* est utilisé pour transmettre un tel résultat.

Le composant *reverse_path* est utilisé par le DAS pour envoyer le(s) chemin(s) inverse(s) entre deux principaux si cela est demandé.

Le composant *crls* est utilisé par le DAS pour retourner des listes de révocation de certificats, lorsque cela est demandé par le client. La définition de ce composant est la suivante :

```
DasResultingCrls ::= SET OF PemCertificateRevocationList
```

La définition de *PemCertificateRevocationList* peut être trouvée dans l'Annexe B.

Il doit être noté que le DAS peut renvoyer des résultats incomplets lorsque la requête constitue une demande de récupération de clés publiques ou de listes de révocation. Cela se produit si le DAS ne peut pas trouver l'information demandée pour tous les principaux spécifiés.

L'élément **DasError** constitue les indications d'erreur retournées par le DAS comme réponse à l'exécution d'une opération. La définition de cet élément est la suivante :

```
DasError  ::= CHOICE {
        authErr             [0] DasAuthErr,
        accessErr           [1] DasAccessErr
}
```

Le composant *authErr* indique des problèmes liés à la manipulation de données publiques d'authentification. Ce composant a le nom d'annuaire de l'entité (*object*) à laquelle le problème se réfère et peut avoir un numéro de série (*serialNo*), si le problème est lié à la manipulation d'un certificat.

```
DasAuthErr    ::= SEQUENCE {
        object      [0] DistinguishedName,
        serialNo    [1] INTEGER          OPTIONAL,
        authPb      [2] INTEGER {
                            CertificationPolicyViolation    (0),
                            KMPViolation                    (1),
                            InvalidCrlSignature             (2),
                            InvalidCertificateSignature     (3),
                            ExpiredCertificate              (4),
                            ExpiredCrl                      (5),
                            RevokedCertificate (6) }
}
```

Le composant *accessErr* indique des problèmes liés à l'accès au DAS ou au service de noms.

```
DasAccessErr ::= INTEGER {
        NameError           (0),
        ServiceError        (1),
        LackOfInformation   (2),
        BadFormedRequest    (3)
}
```

NameError signale un problème relatif au nom fourni à titre d'argument pour l'opération. Par exemple, un des noms fournis ne concorde pas avec le nom d'un objet.

ServiceError signale un problème lié à l'accès au service de noms.

LackOfInformation indique que le DAS n'a pas pu obtenir toute l'information nécessaire à la construction ou à l'évaluation d'un chemin de certification.

BadFormedRequest signale un problème dans les données échangées entre le client et le DAS.

15.4.2. Interaction entre DAS

Les DAS réalisent leur interaction au moyen du Protocole d'Interaction entre DAS (**DASIP** - ***Data Authentication Server Interaction Protocol***). Ce protocole représente donc, le point d'accès à l'opération distribuée des DAS. Le protocole d'interaction entre DAS utilise les mêmes données comme spécifiées pour le protocole d'accès au DAS, mais suit un modèle différent. Le modèle général adopté pour ce protocole est celui des serveurs propageant une opération vers d'autres serveurs.

Dans le protocole d'interaction entre DAS, ces serveurs utilisent les données de protocole **DasMsg** pour propager une requête entre DAS. Une telle requête est normalement émise lorsqu'un Client DAS lance une opération *DasOp* vers le DAS de son KMD et ce domaine ne retient pas toute l'information nécessaire à l'exécution de la demande. Dans un tel cas, le DAS propage l'opération aux DAS adjacents en modifiant la requête initiale de façon à ce que seule la partie de l'opération qui n'a pas été effectuée soit réalisée par ces DAS adjacents.

Lorsqu'un DAS reçoit une opération entrante et qu'il est capable de satisfaire entièrement la demande (ou qu'une erreur s'est produite), ce serveur renvoie cette réponse au DAS expéditeur qui, à son tour le renverra, et ainsi de suite jusqu'à ce que le DAS expéditeur de la requête originelle soit achevé.

De même que dans le modèle d'accès au Service DAS, le modèle d'interaction entre DAS ne suppose pas un comportement synchrone entre ces serveurs. Les requêtes et les réponses à plusieurs opérations peuvent être échangées dans un ordre quelconque, pourvu qu'un DAS reçoive effectivement une réponse (résultat ou erreur) à toute opération propagée vers un autre DAS.

Le protocole d'interaction entre DAS est constitué des mêmes éléments du message *DasMsg* comme dans le protocole DASAP, mais contrairement à ce dernier, le protocole DASIP emploie l'élément *dasTrace*.

```
DasTrace ::= SEQUENCE OF SEQUENCE {
                    dasName     DistinguishedName,
                    fromCA      DistinguishedName,
                    toCA        DistinguishedName
}
```

Conforme nous l'avons décrit dans la Section 14.2.3.2.2, l'élément *dasTrace* sert au contrôle de bouclage pendant la navigation dans le Service DAS. Chaque DAS qui participe à l'exécution d'une opération doit ajouter à cet élément, son nom et la partie de l'opération qu'il a effectuée.

15.5. Assurance de la Protection du Service

Pour que le service du DAS puisse être fiable, les probables sources de vulnérabilités doivent être analysées. Puisque le service du DAS repose sur l'obtention de données pour l'authentification, nous devons analyser l'impact du service sur les données manipulées, à savoir :

- les certificats et les politiques,
- les listes de révocation de certificats,
- la connaissance du DAS.

15.5.1. Usurpation de l'identité du DAS ou du Service de Noms

Dans notre approche le DAS et le service de noms (dans les cas où il est employé) sont des intermédiaires entre les CA et les autres principaux dans le système. Ainsi, il est important de voir si l'introduction de ces deux éléments peut affaiblir les procédures d'authentification. Plus particulièrement, nous nous sommes attachés à vérifier les conséquences d'une possible usurpation d'identité (*mascarade*) de ces éléments.

En ce qui concerne les certificats et les descriptions de politiques, il s'agit des structures protégées par ses signatures digitales qui sont vérifiables de façon récursive. Ainsi, toute tentative d'usurpation de l'identité du service de noms ou du DAS n'entraîne pas de conséquences au niveau de l'authentification car ces données sont publiques et infalsifiables.

Par conséquent, les menaces mentionnées ci-dessus sont immédiatement neutralisées par les fondements des données échangées. Toute tentative d'usurpation de l'identité du service de noms peut être détectée par le DAS et toute tentative d'usurpation de l'identité du DAS peut être détectée par l'utilisateur final.

Reste le problème de la protection des clés d'amorçage : toute clé dont l'authenticité et l'intégrité ne sont pas vérifiables doit être conservée de façon sûre. Par exemple, les clés publiques des CA de plus haut niveau dans des modèles hiérarchiques. Nous pouvons envisager deux types de solution :

(1)le DAS conserve les clés d'amorçage pour les utilisateurs de son domaine,

(2)chaque utilisateur conserve ses clés d'amorçage.

La solution (1) implique l'attribution de confiance du type inconditionnelle au DAS (voir 11.1.1) car celui-ci serait capable de détourner des procédures d'authentification. En outre, selon le niveau de sécurité souhaité, cette solution demanderait l'établissement d'un canal sécurisé entre le DAS et ses utilisateurs finals pour protéger les résultats des opérations.

La solution (2) ne demande pas l'attribution de confiance à une source externe. Chaque utilisateur pourrait disposer d'un cache protégé de clés d'amorçage. Ce cache contiendrait la (les) clé(s) de sa

(ses) CA locale(s) et/ou les clés des CA de plus haut niveau auxquelles l'utilisateur souhaite faire confiance. La protection de ce cache pourrait se faire à travers une signature calculée au moyen de la clé privée de l'utilisateur.

Dans le cas des listes de révocation, un attaquant peut essayer de remplacer une liste demandée par une autre moins récente pour qu'une procédure d'authentification réussisse par erreur. Grâce à notre méthode de description de politiques de certification (voir 13.3), le DAS et l'utilisateur final sont capables d'estimer la "fraîcheur" d'une liste de révocation en comparant les dates d'émission et de mise à jour d'une liste avec la fréquence d'émission des listes que la CA concernée doit respecter.

La description de politiques de certification permet également d'identifier si une CA dispose d'un répondeur de listes de révocation. Dans un tel cas, le DAS peut décider de contacter directement ce répondeur, s'il ne peut pas faire confiance à des DAS intermédiaires. En effet, si les requêtes au répondeur demandent le retour d'une liste de révocation accompagnée d'une estampille de temps (voir 13.2), le problème de remplacement de listes pour d'autres moins récentes est éliminé (sauf par compromission de la clé privée du répondeur, ce qui peut être considéré comme un événement assez rare).

15.5.2. Protection de la Connaissance du DAS

Finalement, il reste à analyser la protection de la connaissance du DAS. Comme cela a été déjà mentionné, les informations qui constituent la connaissance du DAS ne sont pas confidentielles. En contrepartie, l'intégrité de ces données doit être maintenue.

Une façon de réaliser ceci consiste dans la génération d'une signature en utilisant une fonction de scellement à sens unique cryptographiquement forte, e.g., MD5 [Riv92], chiffrée ensuite au moyen d'un algorithme symétrique, e.g., DES [ANSI83]. La clé de chiffrement serait conservée par le DAS pendant son opération et aussi par l'administrateur du système. En cas d'arrêt du DAS, cette clé est supprimée et fournie au DAS par l'administrateur lors de son relancement.

Le service du DAS n'est pas garanti si ses informations de connaissance sont inconsistantes. Nous aborderons ce problème dans la section suivante.

15.5.3. Consistance et gestion de connaissance

Dans la Section 14.2.3.1 nous avons défini les informations que les DAS doivent conserver pour garantir la réalisation de leurs opérations, i.e., leur connaissance. Cependant, la distribution des clés publiques entre les KMD peut changer au cours du temps à cause des événements suivants :

- le changement interne des KMD,
- le changement d'interconnexion entre KMD,
- l'ajout des nouveaux KMD,
- la suppression des KMD.

Les DAS doivent être capables de gérer ces événements de manière à éviter l'inconsistance de leurs connaissances. L'inconsistance de connaissance se produit, par exemple, lorsque des références à des DAS sont incorrectes, i.e., des références qu'associent un DAS à des fausses responsabilités. Il est donc nécessaire de prévoir comment les DAS réagiront face à ce genre d'événement.

La connaissance globale est distribuée parmi les DAS qui, à leur tour, opèrent de façon très autonome. Par conséquent, la gestion de cette connaissance doit aussi être effectuée de façon distribuée et autonome.

15.5.3.1. Consistance et gestion de la connaissance interne et externe minimale

Ces connaissances garantissent le fonctionnement d'un DAS dans son KMD et spécifient les liaisons de base entre DAS adjacents qui assurent la procédure de navigation. Si ces connaissances deviennent inconsistantes, l'opération du DAS n'est plus garantie.

La question qui se pose est la suivante : Dans quelle mesure l'inconsistance des connaissances interne et externe minimale peut entraîner des vulnérabilités à la sécurité d'un KMD ? Le restant de ce paragraphe répond à cette question.

- **Inconsistance de la connaissance interne**

Ce type d'inconsistance découle du fait que la connaissance interne ne correspond pas à la configuration réelle du KMD. Cela peut se produire à cause des événements suivants lorsqu'ils ne sont pas reflétés dans la connaissance interne du DAS :

- Une CA est ajoutée au KMD

  Cela peut entraîner des problèmes de refus de service mais n'entraîne pas de problèmes d'authentification.

  Cette information fait partie de la KMP, qui à son tour est conservée par le DAS. Par conséquent, l'administrateur du domaine doit mettre à jour cette KMP.

- Une CA est supprimée du KMD

  Cela vient du fait que tous les certificats associés à cette entité ne sont plus valables (révoqués et/ou périmes). Or, la détection d'un tel événement fait partie du service fourni par le DAS. Donc, cela ne cause pas de problèmes d'authentification ni de refus de service.

- Modification de la politique de certification attribuée à une CA

  Puisque cette politique est strictement liée au certificat de la CA, sa modification implique la révocation du certificat correspondant. Ainsi, un tel événement a les mêmes conséquences décrites ci-dessus.

- Modification des critères locaux et des métriques d'évaluation du KMD

  Puisque ces informations font partie de la KMP conservée par le DAS, un tel événement ne peut se produire sans que le DAS ne soit averti. La modification des critères locaux et des métriques d'évaluation du KMD exigent la mise à jour de la KMP.

- Changement des références d'emplacement des certificats d'utilisateur

  Cela peut causer, dans des cas extrêmes, des problèmes de refus de service, lorsque des certificats accessibles par les références obsolètes sont révoqués ou périmés. Dans un tel cas, l'administrateur du domaine doit mettre la connaissance du DAS à jour.

- Changement des références d'emplacement des certificats et des listes de révocation des CA

  Dans le cas des certificats de CA, un tel événement subit les mêmes conséquences décrites ci-dessus.

  En contrepartie, dans le cas des listes de révocation, un tel événement peut amener à de graves conséquences car le DAS risque de baser son opération sur des listes de révocation obsolètes. En effet, la question de la récupération de listes de révocation constitue un problème général. Ce travail propose une solution indépendante du reste de cette approche dans le Chapitre 13 (voir 13.2).

♦ **Inconsistance de la connaissance externe minimale**

Ce type d'inconsistance peut se produire à cause des événements suivants lorsqu'ils ne sont pas reflétés dans la connaissance externe minimale du DAS :

- Suppression d'une CA d'interconnexion

  Cela découle du fait que le(s) certificats(s) qui fournissent l'interconnexion à un autre KMD sont révoqués et/ou périmes. Puisque la détection d'un tel événement fait partie du service fourni par le DAS, cela ne cause pas de problèmes d'authentification ni de refus de service.

  Pour éviter un impact défavorable sur la performance des services du DAS, la connaissance externe du DAS doit être mise à jour suite à un tel événement.

- Ajout d'une CA d'interconnexion

  Cela découle du fait qu'une CA appartenant au KMD, et qui n'était pas considérée comme une CA d'interconnexion auparavant, a désormais établi une certification croisée avec des CA appartenant à d'autres KMD.

  En fait, le DAS est capable de détecter un tel événement durant la réalisation de son service. Le problème se pose lors de l'identification du DAS adjacent qui doit être contacté pour la suite de l'opération. Ainsi, une telle inconsistance peut causer des problèmes de refus de service. Par conséquent, la connaissance externe du DAS doit être mise à jour suite à un tel événement.

Les connaissances interne et externe minimale doivent être maintenues dans un état consistant. Des mécanismes d'administration doivent être crées de façon à permettre la gestion de ces connaissances. Des mécanismes spéciaux peuvent être envisagés de façon à ce que le DAS puisse signaler l'anomalie à l'administrateur du service pour qu'il procède à la mise à jour de ces connaissances.

La gestion de la connaissance interne ne demande pas de coopération d'autres DAS. En contrepartie, la gestion de la connaissance externe minimale comprend des références à des DAS adjacents et, pourtant, sa gestion peut exiger la coopération de ces derniers pour la correction des inconsistances.

En tout cas, l'administration des domaines doit prendre des mesures pour garantir que ces connaissances restent dans un état consistant en face d'un changement de la distribution des données publiques d'authentification. Cela doit être fait avec un minimum de surcharge et de coopération entre DAS. Les changements des connaissances doivent être vérifiés et propagés aux autres DAS concernés lorsque nécessaire.

15.5.3.2. Consistance et gestion de la connaissance externe ajoutée

Comme il l'a été déjà mentionné, la connaissance externe ajoutée peut être obtenue à travers le cache de références reçues dans des résultats des opérations. Une telle méthode passe la responsabilité de maintenance des ces informations au DAS qui les conserve. Les DAS ne peuvent pas être certains de l'exactitude de ces informations, et la possibilité de croissance et de diversité de cette connaissance peut exiger un grand effort de maintenance.

Dans ce contexte, deux types d'inconsistance existent, à savoir, l'attribution incorrecte d'adresses et l'attribution incorrecte de responsabilité. L'attribution incorrecte d'adresse se produit lorsqu'une référence désigne un DAS qui n'existe plus ou qui a changé son adresse. L'attribution incorrecte de responsabilité se produit lorsqu'un KMD servi par un certain DAS a changé sa configuration, et la référence à ce DAS (et donc au KMD) n'a pas été mise à jour.

L'attribution incorrecte d'adresse est reconnue lorsque la référence est utilisée pour contacter un DAS. Dans ce cas, l'opération retournera une "erreur adresse", indiquant que la référence est incorrecte. L'attribution incorrecte de responsabilité est reconnue lorsqu'un DAS est contacté pour la réalisation d'une certaine opération dans son domaine et les indications de la requête n'ont pas de rapport avec son KMD. Dans ce cas, ce DAS retournera une erreur indiquant le problème.

L'algorithme de navigation doit être capable de détecter et de probablement corriger de tels problèmes. En particulier, les DAS doivent être capables de fonctionner en présence de connaissance externe ajoutée inexacte en supposant que les connaissances interne et externe minimale sont disponibles et maintenues consistantes.

Une solution générale à ce problème est que le DAS exécute les étapes de navigation en utilisant les connaissances externe minimale et ajoutée jusqu'à ce qu'une erreur soit détectée. Lorsqu'une erreur est détectée, la navigation continue alors en utilisant seulement la connaissance externe minimale car celle-ci est censée être exacte. La procédure de navigation est donc assurée car l'inconsistance de la connaissance externe ajoutée peut être détectée.

PARTIE V

CONCLUSION

A l'heure actuelle, nous sommes confrontés à une expansion extraordinaire de l'utilisation et des capacités des réseaux de télécommunications. Des réseaux isolés sont, de nos jours, en train de s'intégrer pour former un réseau à la fois universel, dans la mesure où il est constitué d'un nombre de réseaux interconnectés, et virtuel, en tant que constitué d'un ensemble de points d'accès logiques permettant une configuration mutable basée sur une intégration variable des réseaux séparés.

La mise en réseau du monde amène toutefois une préoccupation qui lui est inhérente : l'ouverture des systèmes et l'accroissement des enjeux supportés par les systèmes d'information implique une augmentation des besoins en sécurité.

Les réseaux de communication ouvrent deux types de menaces fondamentalement différentes aux ressources et aux informations. D'une part, les réseaux sont un moyen d'accès aux systèmes locaux, d'autre part ils constituent eux-mêmes le lieu des communications. Ainsi, non seulement le danger de manipulation des systèmes locaux et des informations y sont conservées augmente considérablement avec l'interconnexion des systèmes. Il en est de même pour les ressources des systèmes de communication, et l'information transmise par les réseaux comme acte de

communication. Ces ressources et informations sont aussi des cibles pour une manipulation abusive.

Ce travail est ainsi basé sur les besoins en sécurité liés à l'ouverture actuelle des systèmes et sur la tendance de l'évolution des réseaux publics et privés. Nous avons soulevé le fait que les réseaux interconnectés répandus à travers le monde ne peuvent pas être considérés comme sûrs, bien que la nécessité d'authenticité et d'intégrité des données qui y transitent soit de plus en plus pressante. Des systèmes d'information sécurisés doivent être développés pour fonctionner sur des réseaux non fiables. De plus, des mesures de protection physique et cryptographique locales ne suffisent pas à elles seules pour assurer la sécurité des échanges dans un environnement ouvert. Nous avons besoin d'une infrastructure globale pour la mise en place des services de sécurité couplée à des mesures de sécurité de bout-en-bout. Notre travail vise à satisfaire ces deux besoins. Ceux-ci constituent les deux axes de notre travail.

Si d'une part la cryptologie conventionnelle s'est développée pour des environnements clos, notamment pour les besoins de confidentialité à des fins militaires, d'autre part la technologie des télécommunications s'est généralement développée sans égard aux aspects de sécurité. Avec l'émergence de la cryptographie à clé publique, une nouvelle ligne de recherche est en train de se constituer, celle qui reliera ces deux extrêmes afin de permettre le développement des systèmes ouverts sécurisés. Cette nouvelle ligne de recherche est donc celle qui permettra la construction d'une économie "digitale" où des engagements légaux pourront être établis par la voie électronique. Notre travail se situe dans cette nouvelle ligne de recherche en formation, placée entre la cryptologie et les systèmes de communication à grande échelle.

Le but de ce travail n'est pas d'être dépendant de crypto-systèmes spécifiques, car ceux-ci sont vulnérables à un changement continu en fonction des progrès en cryptologie, tant en cryptographie qu'en cryptanalyse. Cependant des cryptosystèmes existants ont été étudiés et intégrés de façon à lier ce travail plus fermement à la réalité et à montrer sa viabilité dans un contexte réel.

En outre, il est tout de même important d'être en mesure d'évaluer les techniques cryptographiques pour être capable de les choisir, de les adapter ou de concevoir des protocoles de sécurité applicables aux protocoles de communication utilisés dans les systèmes ouverts. En reliant ces deux extrêmes, une plate-forme viable pour des implémentations opérationnelles peut être créée.

Un trait de ce travail est la conception et le développement de l'architecture de sécurité *SecUcomx*, qui ne représente pas un modèle spécifique mais une architecture générale à partir de laquelle toute une gamme de services de sécurité peut être implémenté. SecUcomx offre ainsi les services essentiels pour la sécurité tels que l'authentification, l'intégrité et la confidentialité. Services à partir desquels tous les autres services de sécurité peuvent être développés, par exemple le contrôle d'accès et la non-répudiation.

L'architecture SecUcomx a été développée et intégrée dans l'environnement OSI, dans le cadre du projet européen *Password*. Le modèle de référence OSI a été utilisé comme base de cette étude car il représente un cadre de travail globalement accepté. Toutefois, nous avons maintenu une vision critique de ce modèle et comme résultat, nous avons proposé des modifications et des extensions à ce cadre. Notre étude peut servir ainsi de contribution à l'évolution et au débat actuels des systèmes ouverts, afin qu'ils puissent continuer leur croissance tout en préservant la

sécurité des communications. Ce travail n'est pas pour autant dépendant ni spécifique au modèle OSI ; les solutions proposées sont applicables à d'autres architectures de télécommunications.

Cette architecture est compatible avec le cadre X.509. Les aspects de gestion de clés, superficiellement discutés dans ce cadre, ont été étudiés et des solutions viables ont été proposées, tant au niveau de l'activation de ses fonctions qu'au niveau des mécanismes d'administration des clés asymétriques.

Notre travail a étudié le concept et le rôle d'une Autorité de Certification ainsi que ses fonctionnalités. Ces autorités de certification remplaceront les tierces parties (serveurs de clés symétriques) utilisées dans des schémas conventionnels comme les points de confiance des systèmes. Les autorités de certification seront les responsables de l'administration répartie des clés publiques nécessaires au bon déroulement des mécanismes de sécurité. Cela comprend la certification des entités du système et d'autres autorités afin de permettre la connectivité globale en ce qui concerne les échanges sécurisés.

Les modules de SecUcomx ont été intégrés dans des applications existantes, à savoir l'annuaire réparti X.500 et la messagerie électronique X.400, de façon à implémenter leurs modèles de sécurité et à montrer les avantages apportés par cette architecture dans le cadre des applications ouvertes. Ceci nous a poussé à détecter des lacunes et des défaillances dans les modèles de sécurité de ces applications, pour lesquelles nous avons proposé des solutions. Nous avons également proposé des solutions à la sécurisation d'applications ne disposant pas d'extensions de sécurité dans leurs propres modèles.

La plupart des solutions que nous avons présentées ont été testées dans un environnement pilote dans le cadre du projet Password. Puisque ce projet repose sur les services et mécanismes standard et en raison du temps limité du projet, nous n'avons pas pu déployer d'autres extensions dans ce même environnement. Bien que des résultats encourageants aient été obtenus, l'effort de ce groupe de travail doit évoluer pour aboutir à la normalisation de solutions viables.

L'un des points clé de l'architecture SecUcomx est la conception d'un nouvel élément, à savoir le DAS, un Serveur de Données pour l'Authentification, dont le rôle est de construire et de vérifier les chemins de certification nécessaires aux procédures d'authentification. Cela nous a amené à questionner l'approche X.509 en tant que cadre de base pour l'implémentation d'une infrastructure commune pour les services d'authentification. Ceci nous a fait détecter des problèmes ouverts ou négligés par cette recommandation.

Vu l'importance du service d'authentification, et puisque ce service est à la base de tous les services de sécurité, nous nous sommes concentrés sur la viabilité d'implantation effective d'un tel service à grande échelle et dans un environnement hétérogène.

Ayant comme point de départ l'étude des besoins, ainsi que des modèles et architectures de sécurité existants, nous avons remarqué que si tous ces systèmes ont répondu dans un premier temps au besoin de sécurité de groupes d'individus ayant les mêmes exigences et caractéristiques, ils n'en constituent pas moins des mondes isolés les uns des autres. Or, pour pouvoir engendrer sa propre évolution, la création d'un réseau universel doit être accompagnée de la constitution d'une infrastructure permettant l'établissement d'un canal de communication sécurisé entre deux entités quelconques.

Nous avons donc conçu des mécanismes permettant la mise en place d'une infrastructure globale pour le déploiement de services de sécurité ayant comme base l'authentification. Ces mécanismes visent à compléter le cadre X.509 de manière à corriger les défaillances de cette approche et à comporter une multitude de modèles de certification tout en respectant les critères locaux à chaque domaine de sécurité.

Tout au long de la Partie IV de ce travail, de nouveaux concepts et méthodes ont été définis, tels que les répondeurs de CA, les méthodes de description des politiques de certification et de critères locaux d'évaluation de certificats, la méthode d'évaluation de chemins de certification. Tout cela a été matérialisé à travers l'évolution du concept du DAS. Cette évolution comprend, entre autres, une approche distribuée pour le fonctionnement des DAS. Ceci permet l'indépendance vis-à-vis de la méthode de distribution des données publiques nécessaires à l'authentification.

L'évolution du concept du DAS a donc approfondi les concepts d'obtention et d'évaluation de clés publiques, et a introduit un nouveau mécanisme d'évaluation de certificats basé sur la description des politiques de certification et sur les critères locaux à chaque domaine de sécurité. Les composants de l'approche sont tout de même indépendants les uns des autres. En particulier, la réalisation de l'approche n'exige pas l'implémentation de la coopération entre DAS.

Notre travail vise ainsi à aller au-delà du cadre X.509 en proposant une infrastructure indépendante de modèle de certification et de service de noms. Une telle infrastructure peut comporter des types quelconques de certificat tout en respectant les critères locaux d'évaluation de clés publiques et le niveau de sécurité souhaité dans chaque domaine de sécurité. Ainsi, ce travail est fortement relié à la tendance actuelle d'interconnexion et d'ouverture des systèmes hétérogènes, dans la mesure où il offre les moyens pour établir une infrastructure capable d'harmoniser les modèles publics et privés. Autrement dit, dès les modèles dictés par les Etats jusqu'à ceux définis par les *cypherpunks*.

Ce travail offre de directions tant au niveau architectural qu'au niveau d'implémentation pour l'établissement d'applications sécurisées, sans pour autant nuire à la fonctionnalité et à l'ouverture des systèmes, constituant ainsi un pas vers la sécurisation des systèmes ouverts. Ainsi , des travaux futurs vont s'orienter dans plusieurs directions.

L'une des directions tend vers la sécurité des autorités de certification. Puisque ces autorités sont les points de confiance du système, elles doivent être fortement protégées pour limiter les possibilités de compromission des modèles de confiance. D'autres recherches doivent s'orienter vers la conception de ces modèles afin de minimiser les possibilités et les conséquences d'actions malveillantes, ce qui peut permettre de réduire le capital de confiance que chaque utilisateur doit accorder au modèle.

En outre, l'implémentation de mécanismes de sécurité à grande échelle doit viser à la réduction des coûts liés à la réalisation de ces services. Ceci non seulement au niveau des algorithmes cryptographiques, mais aussi au niveau de l'obtention des données publiques nécessaires aux mécanismes de sécurité. Par exemple, selon Altarah [Alt92], des algorithmes efficaces de construction de chemins de certification peuvent être conçus en adaptant des algorithmes de routage, bien que cette approche ne soit pas encore implémentée et validée.

En ce qui concerne le cadre X.509, en plus de la normalisation de notre méthode de description de politiques de certification, il serait souhaitable que le format de certificats soit étendu pour comporter d'autres crypto-systèmes asymétriques que l'algorithme RSA.

En ce qui concerne l'architecture SecUcomx, les services offerts par ses modules permettent la sécurisation des échanges de données sur un réseau ouvert. L'ensemble de la gestion du service est pris en compte et est géré par les éléments du module. Toutefois, afin de permettre la mise en oeuvre de ce service au sein d'un environnement particulier, les spécifications et le modèle de l'architecture permettent de remplacer des éléments de cette architecture par des éléments disponibles dans un environnement ciblé. Par exemple, la substitution du GS par un autre élément de gestion d'accès aux clés privées basé sur l'utilisation d'une carte à puce. La possibilité d'adaptation implique des développements.

D'autres travaux doivent suivre pour approfondir les fonctionnalités du DAS. En particulier, en ce qui concerne le protocole d'accès à ce serveur, pour répondre aux besoins d'utilisateurs de communautés diverses, on peut envisager l'utilisation d'autres attributs de recherche que le nom d'annuaire. Par exemple, l'adresse de messagerie X.400 ou basés sur les noms DNS, voire des numéros de téléphone, etc. Une autre évolution souhaitable est de permettre la manipulation d'un type quelconque de certificat, par exemple le format défini par PKCS #6. Cela permettrait d'élargir la base d'évaluation des protocoles du DAS en les mettant en œuvre à une plus large échelle. Nous considérons que de telles évolutions sont fondamentales pour l'établissement d'une infrastructure universelle pour les services d'authentification.

ANNEXE A

Le Cadre d'Authentification X.509 en ASN.1

Cette annexe inclut la totalité des définitions des types et des valeurs contenues dans le cadre d'authentification X.509 dans le langage ASN.1 [IS8824-1, IS8824-2, IS8824-3, IS8824-4]. Ces définitions correspondent à la version 93/94 de cette recommandation.

```
AuthenticationFramework
        {joint-iso-ccitt ds(5) modules(1) authenticationFramework(7) 2}
DEFINITIONS ::=
    BEGIN

EXPORTS - - All - -

    AlgorithmIdentifier, AuthorityRevocationList, CACertificate, Certificate,Certificates,
    CertificationPath, CertificateRevocationList, UserCertificate, CrossCertificatePair,
    UserPassword, ALGORITHM, ENCRYPTED, HASHED, SIGNATURE, SIGNED;

IMPORTS

    id-at, informationFramework, upperBounds,
    selectedAttributeTypes, basicAccessControl
      FROM UsefulDefinitions {joint-iso-ccitt ds(5)modules(1) usefulDefinitions(0) 2}
```

```
        Name, ATTRIBUTE
          FROM InformationFramework informationFramework

        ub-user-password
          FROM Upper Bounds upperBounds

        AuthenticationLevel
          FROM BasicAccessControl basicAccessControl

        UniqueIdentifier
          FROM SelectedAttributeTypes selectedAttributeTypes;
```

- - types - -

- - Certificate - -

```
        Certificate         ::=   SIGNED {SEQUENCE{
                    version                 [0] Version DEFAULT 1988,
                    serialNumber            CertificateSerialNumber,
                    signature               AlgorithmIdentifier,
                    issuer                  Name,
                    validity                Validity,
                    subject                 Name,
                    subjectPublicKeyInfo    SubjectPublicKeyInfo,
                    issuerUniqueIdentifier  [1] IMPLICIT UniqueIdentifier OPTIONAL,
                                            - - if present, version must be v2
                    issuerUniqueIdentifier  [2] IMPLICIT UniqueIdentifier OPTIONAL
                                            - - if present, version must be v2 }}

        Version     ::=  INTEGER { v1(0), v2(1)}

        CertificateSerialNumber     ::=  INTEGER

        Validity    ::=  SEQUENCE {
                           notBefore  UTCTime,
                           notAfter   UTCTime}

        SubjectPublicKeyInfo  ::= SEQUENCE {
                           algorithm  AlgorithmIdentifier,
                           subjectPublicKey      BIT STRING}

- - 1988 Definition

- -      AlgorithmIdentifier  ::= SEQUENCE {
- -           algorithm          OBJECT IDENTIFIER,
- -           parameters         ANY DEFINED BY algorithm OPTIONAL}

        AlgorithmIdentifier  ::= SEQUENCE {
             algorithm          ALGORITHM.&id ({SupportedAlgorithms}),
```

```
        parameters      ALGORITHM.&Type ({
                            SupportedAlgorithms}{ @algorithm}) OPTIONAL}

    -- Certificates --

    Certificates        ::=   SEQUENCE {
            certificate             Certificate,
            certificationPath       ForwardCertificationPath OPTIONAL}

    ForwardCertificationPath    ::= SEQUENCE OF CrossCertificates

    CrossCertificates           ::= SET OF Certificate

    -- CertificationPath --

    CertificationPath ::=   SEQUENCE{
            userCertificate         Certificate,
            theCACertificates       SEQUENCE OF CertificatePair OPTIONAL}

    CertificatePair     ::= SEQUENCE{
                forward     [0]     Certificate OPTIONAL,
                reverse     [1]     Certificate OPTIONAL
                --at least one of the pair must be present --}

    -- CertificateList --

    CertificateList     ::= SIGNED SEQUENCE{
                        signature               AlgorithmIdentifier,
                        issuer                  Name,
                        lastUpdate              UTCTime,
                        revokedCertificates     SIGNED { SEQUENCE OF SEQUENCE{
                                signature               AlgorithmIdentifier,
                                issuer                  Name,
                                userCertificate     CertificateSerialNumber,
                                revocationDate      UTCTime}} OPTIONAL}}

-- attribute types --

    UserCertificate   ATTRIBUTE ::=   {
            WITH SYNTAX     Certificate
            ID              id-at-userCertificate

    CACertificate     ATTRIBUTE ::=   {
            WITH SYNTAX     Certificate
            ID              id-at-cACertificate

    CrossCertificatePair    ATTRIBUTE ::=       {
            WITH SYNTAX     CertificatePair
            ID              id-at-crossCertificatePair
```

```
CertificateRevocationList    ATTRIBUTE ::= {
        WITH SYNTAX    CertificateList
        ID             id-at-certificateRevocationList

AuthorityRevocationList    ATTRIBUTE ::= {
        WITH SYNTAX    CertificateList
        ID             id-at-authorityRevocationList

UserPassword    ATTRIBUTE ::=    {
        WITH SYNTAX    OCTETSTRING(SIZE(0...ub-user-password))
        EQUALITY MATCHING RULE octetStringMatch
        ID             id-at-userPassword
```

- - information object classes - -

```
ALGORITHM ::= TYPE-IDENTIFIER
```

- - parameterized types (they correspond to the asn.1 **macros**} **- -**

```
HASHED {ToBeHashed} ::=                OCTET STRING ( CONSTRAINED-BY {
  - - must be the result of applying a hashing procedure - -
  - -  to the DER-encoded octets of a value of - - ToBeHashed} )

ENCRYPTED {ToBeEnciphered} ::=      BIT STRING ( CONSTRAINED-BY {
  - - must be the result of applying an encipherment procedure - -
  - -  to the BER-encoded octets of a value of - - ToBeEnciphered} )

SIGNED (ToBeSigned)    ::=          SEQUENCE {
  ToBeSigned,
  COMPONENTS OF SIGNATURE { ToBeSigned } }

SIGNATURE (OfSignature)      ::=      SEQUENCE {
  AlgorithmIdentifier,
  ENCRYPTED  { HASHED { OfSignature} }
```

- - object identifiers assigments - -

```
id-at-userCertificate                OBJECT IDENTIFIER      ::=  {id-at 36}
id-at-cACertificate                  OBJECT IDENTIFIER      ::=  {id-at 37}
id-at-crossCertificatePair           OBJECT IDENTIFIER      ::=  {id-at 38}
id-at-certificateRevocationList      OBJECT IDENTIFIER      ::=  {id-at 39}
id-at-authorityRevocationList        OBJECT IDENTIFIER      ::=  {id-at 39}
id-at-userPassword                   OBJECT IDENTIFIER      ::=  {id-at 35}
```

END **- - of Authentication Framework Definitions**

ANNEXE B

Architecture SecUcomx en ASN.1

Le present annexe comprend la totalité des définitions de types et de valeurs conçus pour la architecture SecUcomx dans le langage ASN.1 [X.208].

```
SecUcomx DEFINITIONS ::=
BEGIN

EXPORTS

    GsClientRequest, GsClientResponse, CertificationRequestMessage,
    CACertificationRequestMessage, DASmsg, AuthenticationSegment,
    SignatureData, EntityName, EntityPass,
    PEMCertificateRevocationList, PkcsBt1Seq;

IMPORTS

    informationFramework, authentificationFramework, directoryAbstractService
      FROM UsefulDefinitions {joint-iso-ccitt ds(5)modules(1) usefulDefinitions(0)}

    Attribute, DistinguishedName
      FROM InformationFramework informationFramework

    AttributeError, NameError, Referral, SecurityError, ServiceError, AbandonFailed,
    BindError, DsaReferral
      FROM DirectoryAbstractService directoryAbstractService
```

```
    Certificate, Certificates, CertificationPath,
    AlgorithmIdentifier, SubjectPublicKeyInfo
      FROM AuthentificationFramework authentificationFramework;
```

- - Messages between the GS and its client

```
    GsClientRequest ::= ENCRYPTED SEQUENCE {
          appli-name        OCTET STRING,
          object            OCTET STRING,
          time              UTCTime,
          random            BIT STRING }

    GsClientResponse ::= ENCRYPTED SEQUENCE {
          encrypted         ENCRYPTED object,
          random            BIT STRING }
```

- - Certification Request Message

```
    CertificationRequestMessage ::= SIGNED SEQUENCE {
      publicKey           [0]   SubjectPublicKeyInfo,
      authSegm            [1]   ENCRYPTED AuthenticationSegment,
      entitySignature     [2]   SIGNATURE SignatureData }

    AuthenticationSegment ::= SEQUENCE {
      signData        SignatureData,
      userName        EntityName,
      userPass        EntityPassword,
      signKey         BIT STRING }

    SignatureData ::= SEQUENCE {
        date        INTEGER,
        random      BIT STRING }

    EntityName ::= OCTET STRING(SIZE(0..ub-entity-name))

    EntityPassword ::= OCTET STRING(SIZE(0..ub-entity-pass))

    ub-entity-name INTEGER ::= 64

    ub-entity-pass INTEGER ::= 64
```

- - Variation of the CertificationRequestMessage for CA Certification

```
    CACertificationRequestMessage ::= SIGNED SEQUENCE {
      publicKey                 [0]   SubjectPublicKeyInfo,
      authSegm                  [1]   ENCRYPTED CAAuthenticationSegment,
      entitySignature           [2]   SIGNATURE SignatureData }
```

```
CAAuthenticationSegment ::= SEQUENCE {
  CASign                SignatureData,
  CAName                DistinguishedName,
  CAManagerName         DistinguishedName,
  CAPassword            EntityPassword,
  signKey               BIT STRING }
```

- - DAS Protocol Messages (Initial Version)

```
DASmsg ::= SEQUENCE {
  dasId     INTEGER,
  dasOp     CHOICE {
                dasEstablishOp          [0] DasEstablishOp,
                dasVerifyOp             [1] DasVerifyOp,
                dasCertificatesOp       [2] DasCertificatesOp,
                dasCompleteOp           [3] DasCompleteOp,
                dasPubKeyOp             [4] DasPubKeyOp} }

DasEstablishOp ::= CHOICE {
                dasEstablishArg         [0] DasEstablishArg,
                dasEstablishRsp         [1] DasEstablishRsp,
                dasError                [2] DasError}

DasVerifyOp ::= CHOICE {
                dasVerifyArg            [0] DasVerifyArg,
                dasVerifyRsp            [1] DasVerifyRsp,
                dasError                [2] DasError}

DasCertificatesOp ::= CHOICE {
                dasCertificatesArg      [0] DasCertificatesArg,
                dasCertificatesRsp      [1] DasCertificatesRsp,
                dasError                [2] DasError}

DasCompleteOp ::= CHOICE {
                dasCompleteArg          [0] DasCompleteArg,
                dasCompleteRsp          [1] DasCompleteRsp,
                dasError                [2] DasError}

DasPubKeyOp ::= CHOICE {
                dasPubKeyArg            [0] DasPubKeyArg,
                dasPubKeyRsp            [1] DasPubKeyRsp,
                dasError                [2] DasError}
```

- - DasEstablishOp

```
DasEstablishArg ::= SEQUENCE {
  from      DistinguishedName,
  to        DistinguishedName}
```

```
DasEstablishRsp ::= SEQUENCE {
  forward  CertificationPath,
  reverse   CertificationPath}
```

- - DasVerifyOp

```
DasVerifyArg ::= CertificationPath

DasVerifyRsp ::= NULL
```

- - DasCertificatesOp

```
DasCertificatesArg ::= DistinguishedName

DasCertificatesRsp ::= Certificates
```

- - DasCompleteOp

```
DasCompleteArg ::= SEQUENCE {
  from              Certificates,
  to                DistinguishedName}

DasCompleteRsp ::= NULL
```

- - DasPubKeyOp

```
  DasPubKeyArg ::= DistinguishedName

  DasPubKeyRsp ::= Attribute       - - in fact a set of SubjectPublicKeyInfo
```

– Service and Security Errors

```
  DasError ::= CHOICE {
       dapPb      [0]   DapPb,
       authPb     [1]   AuthPb}

  DapPb ::= CHOICE {
       attErr[0] AttributeError,
       nameErr    [1] NameError,
       srvErr     [2] ServiceError,
       ref        [3] Referral,
       dsaRef     [4] DsaReferral,
       secErr     [5] SecurityError,
       abnFail    [6] AbandonFailed,
       bindErr    [7] BindError}

  AuthPb  ::= SEQUENCE {
       obj        [0]   DistinguishedName,
       serialNo   [1]   INTEGER,
```

```
authPb      [2]   INTEGER {
                      AUTH-PB-INV-CREDENTIALS (2),
                      AUTH-PB-INV-SIGNATURE        (4),
                      AUTH-PB-EXPIRED-CERT         (6),
                      AUTH-PB-REVOKED-CERT         (7)}}
```

- - PEM Certificate Revocation List (for replacing the X.509 "bugged" format)

```
PEMCertificateRevocationList ::= SIGNED SEQUENCE {
  signature                 AlgorithmIdentifier,
  issuer                    DistinguishedName,
  lastUpdate                UTCTime,
  nextUpdate                UTCTime,
  revokedCertificates       SEQUENCE OF SEQUENCE {
                            userCertificate    CertificateSerialNumber,
                            revocationDate     UTCTime } OPTIONAL }
```

- - X.509 macros redefinitions- -

```
SIGNED MACRO ::=
BEGIN
    TYPE NOTATION     ::=   type (ToBeSigned)
    VALUE NOTATION    ::=   value (VALUE SEQUENCE{
                      to-be-signed          ToBeSigned,
                      algo-id               AlgorithmIdentifier,
                      encrypted             [UNIVERSAL 11] IMPLICIT BIT STRING
                                            - - where the bit string is the result
                                            - - of the hashing of the value of
                                            - - "ToBeSigned"– })
END - - of SIGNED

SIGNATURE MACRO ::=
BEGIN
    TYPE NOTATION     ::=  type (OfSignature)
    VALUE NOTATION    ::=  value (VALUE SEQUENCE{
                      algo-id               AlgorithmIdentifier,
                      encrypted             [UNIVERSAL 11] IMPLICIT BIT STRING
                                  - - where the bit string is a function
                                  - - (e.g. a compressed or hashed version)
                                  - - of the value "OfSignature", which may
                                  - - include the identifier of the algorithm used
                                  - - to compute the signature– })
END - - of SIGNATURE
```

- - Useful definitions for PKCS #1

```
PkcsBt1Seq ::= SEQUENCE {
  digestAlgo          AlgorithmIdentifier,
  digest              OCTET STRING}

END - - of SecUcomx definitions
```

ANNEXE C

Définitions ASN.1 pour l'Infrastructure Global pour les Services d'Authentication

Le présent annexe résume les définitions établies dans la partie IV de ce travail. Ceci sous la forme d'un module ASN.1 contenant les extensions nécessaires à la mise en oeuvre d'une infrastructure globale pour les services d'authentification.

```
GlobalAuthenticationInfrastructure
DEFINITIONS ::=
BEGIN

-- EXPORTS all

IMPORTS

    informationFramework, authentificationFramework, distributedOperations
      FROM UsefulDefinitions {joint-iso-ccitt ds(5)modules(1) usefulDefinitions(0)}

    ATTRIBUTE, ATTRIBUTE-SYNTAX, OBJECT-CLASS,
    AttributeType, AttributeValue, AttributeValueAssertion, DistinguishedName
      FROM InformationFramework informationFramework

    AccessPoint
      FROM DistributedOperations distributedOperations

    PemCertificateRevocationList
      FROM SecUcomx
```

```
Certificate, CertificateSerialNumber, AlgorithmIdentifier
  FROM AuthentificationFramework authentificationFramework;
```

- - Definitons for Certification Policy Description - -

```
Policy ::= SIGNED SEQUENCE {
        version             [0] INTEGER { 1994(0) } DEFAULT 1994,
        policyDescription       [1] PolicyDescription,
        signedBy            [2] DistinguishedName,
        subject             [3] DistinguishedName,
        othersAttributes    [4] SET OF AttributeValueAssertion OPTIONAL,
        relates_to          [5] SEQUENCE OF SerialNumber,
}

PolicyDescription ::= SEQUENCE {
        certifier_has_trusted_key       [0] BOOLEAN OPTIONAL,
        entityType                      [1] EntityType,
        certificateSemantic             [2] CertificateSemantic,
        assuranceLevel                  [3] AssuranceLevel OPTIONAL,
        rulesforCertification           [4] CertificationRules OPTIONAL
                                -- absent if the subject is not allowed to
                                -- certify or if it is a cross or reverse certificate }
```

- - certification models identification - -

```
CertificationModel ::= OBJECT IDENTIFIER
```

- - entity type identification - -

```
EntityType ::= OBJECT IDENTIFIER
```

- - certificate semantic - -

```
CertificateSemantic ::= ENUMERATED {
    indicates_a_policy_object           (1),
    allows_authority_delegation         (2),
    indicates_authority_delegation      (3),
    indicates_a_reverse_certificate     (4),
    indicates_a_cross_certificate       (5),
    certification_disallowed            (6) }
```

- - assurance level - -

```
AssuranceLevel ::= ENUMERATED {
                    high        (0),
                    medium      (1),
```

```
                low          (2),
                unknown      (3) }
```

- - certification rules - -

```
CertificationRules ::= SEQUENCE {
  jurisdiction                          [0] SET OF {
                                            entities     NamingRules,
                                            semantic     CertificateSemantic},
  revocation_frequency                  [1] INTEGER, – in days
  crl_responder_address                 [2] PresentationAddress OPTIONAL,
                                        – absent if not used
  certificate_validity                  [3] INTEGER, – in days
  key_restriction                       [4] SEQUENCE OF AlgorithmIdentifier,
                                        – parameters may be used to
                                        – specify the minimum key size
  distinct_keys_for_different_issuers   [5] BOOLEAN OPTIONAL,
                                        – absent if the certified key is
                                        – one of the master keys
  how_many_levels_up_to_the_leaf        [6] INTEGER  OPTIONAL,
                                        – absent if undefined }
```

- - jurisdiction identification - -

```
NamingRules ::= CHOICE {
  item        [0]   NamingRulesItem,
  and         [1]   SET OF NamingRules,
  or          [2]   SET OF NamingRules,
  not         [3]   NamingRules }

NamingRulesItem ::= CHOICE {
  subordination  [0]  NULL,
  present        [1]  AttributeType,
  equality       [2]  AttributeValueAssertion,
  substrings     [3]  SEQUENCE {
                              type          AttributeType,
                              strings       SEQUENCE OF CHOICE {
                                            initial       [0] AttributeValue,
                                            any           [1] AttributeValue,
                                            final         [2] AttributeValue }},
  objectClass    [4]  OBJECT IDENTIFIER
  entityType     [5]  OBJECT IDENTIFIER }
```

- - Policy Description Emplacement : object classes redefinition

```
strongAuthenticationUser   OBJECT-CLASS
    SUBCLASS OF top
```

```
        MUST CONTAIN { userCertificate, policy }
    ::= new-oid-to-be-defined

    certificationAuthority   OBJECT-CLASS
        SUBCLASS OF top
        MUST CONTAIN        { cACertificate, certificateRevocationList, policy }
        MAY CONTAIN   { crossCertificatePair }
    ::= new-oid-to-be-defined

    policy     ATTRIBUTE
        WITH ATTRIBUTE-SYNTAX Policy
    ::= oid-to-be-defined
```

- - END OF Policy Description definitions

- - Definitions for Local Domaine Evaluation Criteria Description

```
        DomainePolicyDescription ::= SIGNED SEQUENCE {
                accept          [0] SET OF DomainePolicy OPTIONAL,
                refuse          [1] SET OF DomainePolicy OPTIONAL,
                strict_policy   [2] BOOLEAN OPTIONAL,
                crl_retrieva    [3] INTEGER – frequency in hours
                signedBy        [4] DistinguishedName,
                signedAt        [5] UTCTime
        }
```

- - Domaine Policy

```
        DomainePolicy ::= SEQUENCE {
                model           [0] CertificationModel OPTIONAL,
                certificate_id  [1] SEQUENCE {
                                        number      SerialNumber,
                                        issuer          DistinguishedName
                                } OPTIONAL,
                subject         [2] DistinguishedName OPTIONAL,
                policy          [3] PolicyDescription OPTIONAL
                – at least one of them must be present –
        }
```

- - END OF Local Domaine Definitions

- - KMP Description and DAS Knowledge Definitions

- - Kmp Description

```
KmpDescription::= SEQUENCE {
    responsability    SET OF DistinguishedName,
    localCriteria     DomainPolicyDescription,
    weighting         CaClassesWeighting }

CaClassesWeighting ::= SEQUENCE OF {
    caClass    OBJECT IDENTIFIER,
    weight     INTEGER }
```

- - Internal Knowledge

```
InternalKnowledge ::= SEQUENCE {
    kmp        KmpDescription,
    kmdData    KmdDataRetrieval }

KmdDataRetrieval ::= SEQUENCE {
    userDataRetrieval     UserDataRetrieval,
    caDataRetrieval       CaDataRetrieval }

UserDataRetrieval ::= UniqueChoice

CaDataRetrieval ::= CHOICE {
    allCA      [0] UniqueChoice,
    eachCA     [1] SET OF EachCAChoice }

UniqueChoice ::= SEQUENCE {
    type       [0] INTEGER { das(0), x500(1), solo(2),
                             gopher(3), whois(4) },
    reference  [1] AccessPoint OPTIONAL – absent if "das" is chosen }

EachCAChoice ::= SEQUENCE {
    ca         SET OF DistinguishedName,
    choice     UniqueChoice }
```

- - External Knowledge

```
MinimalExternalKnowledge ::= SET OF ExternalReference

ExternalReference::= SEQUENCE {
    dasAcess              AcessPoint,
    interconnexionCA      SET OF DistinguishedName }
```

- - END OF KMP Description and DAS Knowledge

- - DAS Caches Definitions

- - main cache

```
CertificateCacheEntry :: = SET OF {
  certificate[0] Certificate,
  policy    [1] Policy OPTIONAL,
  controls  [2] SEQUENCE {
                  n          [0] INTEGER,
                  c          [1] INTEGER,
                  last       [2] UTCTime,
                  cp         [3] BOOLEAN OPTIONAL } }

CrlCacheEntry :: = SEQUENCE {
  crl       PemCertificateRevocationList,
  controls  SEQUENCE {
                  n     INTEGER,
                  c     INTEGER,
                  last  UTCTime,
                  ttl    INTEGER } }
```

- - auxiliar cache

```
AuxiliarCacheEntry :: = SET OF {
  cp -indic  SEQUENCE OF { -  cert-id
                  serialNumber     INTEGER,
                  subject          DistinguishedName,
                  issuer           DistinguishedName OPTIONAL },
  eval-data SET {
                  metric     Metrics,
                  value      Evaluation Result,
                  date       UTCTime } }
```

- - END OF DAS Caches Definitions

- - DAS Protocols

- - DASAP : Data Authentication Service Access Protocol

```
DasMsg ::= SEQUENCE {
  dasOpId             [0] INTEGER,
  dasipTrace          [1] DasTrace OPTIONAL, - - not used in DASAP
  dasOp               [2] CHOICE {
                            dasRequest    [0] DasRequest,
                            dasResult         [1] DasResult,
                            dasError          [2] DasError }
  }
```

- - DAS Request

```
DasRequest ::= SEQUENCE {
  dasServiceFlags          [0] DasServiceFlags,
  dasTarget                [1] SEQUENCE OF
                                   DistinguishedName OPTIONAL,
  dasClientProposition     [2] DasCertificates        OPTIONAL,
  certification_model      [3] OBJECT IDENTIFIER      OPTIONAL,
  requested_key            [4] AlgorithmIdentifier    OPTIONAL,
  metrics                  [5] Metrics                OPTIONAL
}

DasServiceFlags ::=  BIT STRING {
                     forward_path          (0),
                     reverse_path          (1),
                     crl                   (2),
                     policies              (3),
                     dontUseCertCache      (4),
                     dontUseCrlCache       (5) }

DasCertificates ::= SEQUENCE OF Certificate
```

- - Evaluation Metrics

```
Metrics ::= SEQUENCE {
  path-length          [0] INTEGER OPTIONAL,
  cp-assurance         [1] CpAssuranceIndication OPTIONAL,
  both                 [2] BOOLEAN OPTIONAL }

CpAssuranceIndication ::= SEQUENCE {
  globalCpAssurance        [0] AssuranceLevel OPTIONAL,
  specificsCaAssurance     [1] SEQUENCE OF {
                                  caClass   OBJECT IDENTIFIER,
                                  value     AssuranceLevel } OPTIONAL,
  minimumCAAssurance       [2] AssuranceLevel OPTIONAL }
```

- - DAS Result

```
DasResult ::= SEQUENCE {
  forward_path         [0] DasResultingKeys   OPTIONAL,
  reverse_path         [1] DasResultingKeys   OPTIONAL,
  crls                 [2] DasResultingCrls   OPTIONAL
  – at least one of these components must be present }

DasResultingKeys ::= SEQUENCE OF {
  path                     [0] DasCertificates        OPTIONAL,
  policies                 [1] DasPolicies            OPTIONAL,
  evaluation               [2] EvaluationResult       OPTIONAL
  – at least one of these components must be present }
```

```
DasPolicies ::= SEQUENCE OF Policy

DasResultingCrls ::= SET OF PemCRL

EvaluationResult ::= REAL

  - - DAS Error

DasError  ::= CHOICE {
  authErr   [0] DasAuthErr,
  accessErr [1] DasAccessErr }

DasAuthErr    ::= SEQUENCE {
  object      [0] DistinguishedName,
  serialNo    [1] INTEGER     OPTIONAL,
  authPb      [2] INTEGER {
                  CertificationPolicyViolation   (0),
                  KMPViolation                   (1),
                  InvalidCrlSignature            (2),
                  InvalidCertificateSignature    (3),
                  ExpiredCertificate             (4),
                  ExpiredCrl                     (5),
                  RevokedCertificate             (6) } }

DasAccessErr ::= INTEGER {
                  NameError           (0),
                  ServiceError        (1),
                  LackOfInformation   (2),
                  BadFormedRequest    (3) }
```

- - DASIP : Data Authentication Service Interaction Protocol - -

```
- - It uses the same definitions as for DASAP - -

DasTrace ::= SEQUENCE OF SEQUENCE {
               dasName  DistinguishedName,
               fromCA   DistinguishedName,
               toCA     DistinguishedName }
```

- - END OF DAS Protocols Definitions

- - Protocols for Certificate Revocation Lists Retrieval - -

```
revokedCertificates ::= SEQUENCE OF revokedCertificate

revokedCertificate     SIGNED SEQUENCE {
  issuer               Name,
```

```
    subject             Name,
    certificate         CertificateSerialNumber,
    revocationDate      UTCTime }
```

END OF - - **GlobalAuthenticationInfrastructure**

BIBLIOGRAPHIE

[Ade88] L. Adelman. An Abstract Theory of Computer Viruses. in Advances in Cryptology: Proceedings of CRYPTO '88, a Workshop on the Theory and Application of Cryptology, Santa Barbara, CA, August 1988, pp. 354-374. Berlin/New York: Springer-Verlag, 1988.

[Akl83] S. G. Akl. *Digital signatures: a tutorial survey.* IEEE Computer Magazine, Vol. 16, No. 2, pp. 15-24, February 1983.

[Alt92] A. Altarah. *Chimaera : Un Modèle pour la Sécurité des Systèmes Ouverts*, Phd Thesis, Université de Nice Sophia-Antipolis, December 1992.

[AM91] D. Arlabosse et F. Marathée. Messagerie Electronique et Réseaux de la Recherche. Centre Spatial de Toulouse (CNES) - Service Télématique. Version 1. Mars 1991.

[AML+93] F. Anklesaria, M. McCahill, P. Lindner, D. Johnson, D. Torrey & B. Albert, D. *The Internet Gopher Protocol (a distributed document search and retrieval protocol).* RFC 1436. March 1993.

[ANSI81] ANSI X3.92, *Data Encryption Algorithm.* American National Standards Institute, New York, 1981.

[ANSI83] ANSI X3.106, *Data Encryption Algorithm - Modes of Operation.* American National Standards Institute, New York, 1983.

[ANSI85] X9.17-1985, *Financial Institution Key Management (Wholesale)*, American National Standard, American Bankers Association, Washington, DC, 1985.

[Atz94] J-M. Atzel. *Quel niveau de risque accepter?*. 01 Informatique. 3 Juin 1994.

[AW93] ACTUEL-WIRED. *Actuel plonge dans le Cyber*. Actuel. No. 34. pp. 28-74. Octobre 1993.

[BA93] J-C. Bolot and H. Afifi. *Evaluating Caching Schemes for the X.500 Directory System*. in proceedings of 13th IEEE International Conference on Distributing Computing Systems (ICDCS 13). Pittsburgh, PA. pp. 112-119. May 1993.

[Bal93] D Balenson. *Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers*. RFC 1423. TIS. February 1993.

[BAN89] M. Burrows, M. Abadi, R. M. Needham. *A Logic of Authentication*. ACM Transactions on Computer Systems, Vol. 8, No. 1, February 1990, pp. 18-36.

[BDB92] M. V. D. Burmester, Y. G. Desmedt and T. Beth. *Efficient Zero-knowledge Identification Schemes for Smart Cards*. Computer Journal, Vol. 35, No. ?, pp. 21-29, ? 1992.

[Ben89] S. Benford. Navigation and Knowledge Management within a Distributed Directory System. in the proceedings of IFIP '89 Conference on Message Handling Systems and Distributed Applications. Elsevier Science. 1989.

[BF92] N. Borenstein, N. Freed. *MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies*. RFC 1341. June 1992.

[BG84] M. Blum and S. Goldwasser. *An Efficient Probabilistic Public-Key Encryption Scheme which Hides all Partial Information*. in Advances in Cryptology: Proceedings of CRYPTO 84, a Workshop on the Theory and Application of Cryptology, Santa Barbara, CA, August 19-22, 1984, pp. 289-299. Berlin/New York: Springer-Verlag, 1985.

[Bis91] M. Bishop. *An Overview of Computer Viruses in a Research Environment*. Department of Mathematics and Computer Science. Darmouth College. 1991.

[BM90] S. M. Bellovin and M. Merrit. *Limitations of the Kerberos Authentication System*. Computer Communication Review. Vol. 20, No. 5, pp. 119-132, October 1990.

[BO88] E. F. Brickell and A. M. Odlyzko. *Cryptanalysis: a survey of recent results*. Proceedings of the IEEE, Vol. 76, No. 5, May 1988, pp. 578-593.

[Boo81] K. S. Booth. *Authentication of signatures using public key encryption*. Communications of the ACM, Vol. 24, No. 11, pp. 772-774, November 1981.

[Boy93] C. Boyd. *Security Architectures Using Formal Methods.* IEEE Journal on Selected Areas in Communications, Vol. 11, No. 5, pp. 694-701, June 1993.

[Bra88] G. Brassard, Lecture Notes in Computer Science Vol. 325: *Modern Cryptology: a Tutorial.* Berlin/New York: Springer-Verlag, 1988.

[BS90] E. Biham and A. Shamir. *Differential Cryptanalysis of DES-like Cryptosystems.* in Advances in Cryptology: Proceedings of CRYPTO'90, a Workshop on the Theory and Application of Cryptology, Santa Barbara, CA, August 1990, pp. 2-21, Berlin/New York: Springer-Verlag.

[BS91] E. Biham and A. Shamir. *Differential Cryptanalysis of Feal and N-hash.* in Advances in Cryptology - EUROCRYPT '91, a Workshop on the Theory and Applications of Cryptology, Berlin, 1992, Springer-Verlag.

[CEC91] Commission of the European Communities, Directorate-General XIII, *Information Technology Security Evaluation Criteria (ITSEC)*, 1991.

[CO90] J, Crowcroft and J. Onions. *Network Time Protocol over the OSI Remote Operations Service.* RFC 1165, 1990.

[Coh84] F. Cohen. *Computer Viruses: Theory and Experiments.* Proceddings of the 7th National Computer Security Conference. pp. 240-263. 1984.

[Coh87] F. Cohen. *Computer Viruses: Theory and Experiments.* Computers & Security. Vol. 6. No. 1. pp. 22-35. February1987.

[Coh88] F. Cohen. *On the Implications of Computer Viruses and Methods of Defense.* Computers & Security. Vol. 7. No. 2. pp. 167-184. April 1988.

[Coh89] F. Cohen. *Practical Defenses against Computer Viruses.* Computers & Security. Vol. 8. No. 2. pp. 325-344. April 1989.

[Coh90] F. Cohen. *A Short Course of Computer Viruses.* ASP Press. Pitsburgh, PA 1990.

[Cro82] D. Crocker. *Standard for the format of ARPA Internet text messages.* RFC 822. Updated by RFC1138 and RFC1148. August 1982

[CFSD90] J.D. Case, M. Fedor, M.L. Schoffstall, C. Davin. *Simple Network Management Protocol (SNMP).* RFC 1157. May 1990.

[DAN91] DANET. *OSITEL/400: Functional Description, Version 2.0.* Danet. 1991.

[Dav88] J. David. *Treating Viral Fever.* Computers & Security. Vol. 7. No. 2. pp. 255-258. April 1988.

[Den88] D. Denning. *The Science of Computing: Computer Viruses*. American Scientist Vol. 76. No. 3. pp. 236-238. May 1988.

[Den90] D. Denning. *Computers under Attack: Intruders, Worms and Viruses*. Addison-Wesley Publishing Co., Reading, MA, 1990.

[Des92] Y. Desmedt, "Breaking Traditional Computer Security Barriers", Computer Security: Proceedings of ESORICS 92, a European Symposium on Research in Computer Security, pp. 125-138, Berlin, Spring-Verlag, 1992.

[Dif88] W. Diffie. *The first ten years of public-key cryptography*. Proceedings of the IEEE, Vol. 76, No. 5, May 1988, pp. 560-577.

[DH76] W. Diffie and M. E. Hellman, *New Directions in Cryptography*. IEEE Transactions on Information Theory, Vol. IT-22, No. 6, November 1976, pp. 644-654.

[DH79] W. Diffie and M. E. Hellman, *Privacy and Authentication: an Introduction to Cryptography*. Proceedings of the IEEE, Vol. 67, No. 3, March 1979, pp. 397-427.

[DM83] R. DeMillo and M. Merritt. *Protocols for Data Security*. IEEE Computer Magazine, Vol. 16, No. 2, pp. 39-51, February 1983.

[DO85] Y. Desmedt and A. M. Odlyzko. *A Chosen Text Attack on the RSA Cryptosystem ans Some Discrete Logarithm Schemes*. in Advances in Cryptology - CRYPTO '85, proceedings of a Conference on the Theory and Applications of Cryptology. Santa Barbara, CA, August 1985, pp. 516-521. Berlin/New York: Springer-Verlag, 1986.

[DoD85] Department of Defence Standard. *Trusted Computer System Evaluation Criteria. "Orange Book"*. DOD 5200.28-STD, December 1985.

[DP84] D. W. Davies and W. L. Price. *Security for Computer Networks*. John Wiley and Sons, 1984.

[DS81] D. E. Denning and G. M. Sacco, *Timestamps in Key Distribution Protocols*. Communications of the ACM, Vol. 24, No. 8, August 1981, pp. 533-536.

[DVG84] Y. Desmedt, J. Vandewalle, and R. J. M. Govaerts. *A critical analysis of the security of knapsack public key algorithms*. IEEE Transactions on Information Theory, Vol. IT-30, No. 4, pp. 601-611, July 1984.

[E3X92a] E3X. *X.500: Manuel d'Administration*. Document Interne E3X.1992.

[E3X92b] E3X. *Présentation Générale de UCOMX.400*. Document Interne E3X.1992.

[ECMA89] ECMA/TC32/TG9/89. *Security in Open Systems. Data Elements and Service Definition.* ECMA Standard 1989.

[ElG85] T. ElGamal. *A public key cryptosystem and a signature scheme based on discrete logarithms.* IEEE Transactions on Information Theory, Vol. IT-31, No. 4, pp. 469-472, July 1985.

[Fah92] P. Fahn. *Answers to Frequently Asked Questions about Today's Cryptology,* RSA Laboratories, September 1992.

[FFS87] U. Feige, A. Fiat, and A. Shamir. *Zero-knowledge Proofs of Identity.* in Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, New York, NY, May 25-27, 1987, pp. 210-217. New York: ACM, 1987.

[FL93] W. Fumy and P. Landrock. *Principles of Key Management.* IEEE Journal on Selected Areas in Communications, Vol. 11, No. 5, pp. 785-793, June 1993.

[Fra83] B. D. Fraim. *Scomp: a Solution to the Multilevel Security Problem.* Computer, Vol. 16, No. 7, pp. 26-34, July 1993.

[FS86] A. Fiat and A. Shamir. *How to Prove Yourself: Practical Solutions to Identification and Signature Problems.* in Lecture Notes in Computer Science Vol. 263: Advances in Cryptology - CRYPTO '86, proceedings of a Conference on the Theory and Applications of Cryptology, Santa Barbara, CA, August 11-15, 1986, pp. 186-194. Berlin/New York: SpringerVerlag, 1987.

[GM84] S. Goldwasser and S. Micali. *Probabilistic Encryption.* Journal of Computer and System Sciences, Vol. 28, No. 2, April 1984, pp. 270-299.

[GM92] D. M.. Gordon and K. S. McCurley. *Massively Parallel Computation of Discrete Logarithms.* in Advances in Cryptology: Proceedings of CRYPTO '92, a Workshop on the Theory and Application of Cryptology. Santa Barbara, CA, August 1992. Berlin/New York: Springer-Verlag.

[GM93] J. Galvin and K. McCloghrie. *Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2).* RFC 1446. April 1993.

[GMD92] J. Galvin, K. McCloghrie and J. Davin. *SNMP Security Protocols.* RFC 1352. July 1992.

[Goo91] D. Goodman. *PARADISE: the COSINE X.500 Pilot Service.* Computer Networks and ISDN Systems. Vol. 23. No 1-3. pp. 111-114. November 1991.

[Gri89] R. Grimm. *Security on Networks: Do we really need it?.* Computer Networks and ISDN Systems 17, pp. 315-323, 1989.

[Gri91] R. Grimm et al. *SECUDE, Principles of Security Operations (V.1), Specifications of Security Interfaces (V.2), Unix User Manual (V.4)*. GMD, Darmstadt, 1991

[GW93] J. Gargano and K. Weiss. W*hois and Network Information Lookup Service Whois++*. Internet Draft. WNILS Working Group. IETF. July 1993.

[Has88] J. Hastad. *Solving simultaneous modular equations of low degree*. SIAM Journal on Computing, Vol. 17, No. 2, pp. 336-341, April 1988.

[Hel87] M. E. Hellman. *Commercial Encryption*. IEEE Network Magazine. Vol. 1 No 2. pp. 6-10. April 1987.

[Hel93] M. E. Hellman. *Clipper Chip*. Stanford University. 18 April 1993.

[Hig88a] H. Highland. *Random Bits and Bytes: Case History of a Virus Attack.* Computers & Security. Vol. 7. No. 1. pp. 3-5. Febraury 1988.

[Hig88b] H. Highland. *The Brain Virus: Fact and Fantasy*. Computers & Security. Vol. 7. No. 4. pp. 367-370. August 1988.

[HL93] L. Harn and H-Y. Lin. *Key Management for Descentralized Computer Network Services*. IEEE Transactions on Communications, Vol. 41, No. 12, pp. 1777-1779, May 1989.

[HPZW93] C. Huitema, P-A. Pays, A. Zahm and A. Woermann. *Simple Object Look-up Protocol (SOLO)*. Internet Draft. Network Working Group. December 1993.

[Hui91a] C. Huitema. *PIZARRO, general presentation.* Internal Document. Project RODEO. INRIA.1991.

[Hui91b] C. Huitema. *MAVROS, Highlights on an ASN.1 Compiler*. Internal Document. Project RODEO. INRIA.1991.

[IM90] C. I'Anson, C. Mitchell. *Security Defects in CCITT Recommendation X.509 - The Directory Authentication Framework.* ACM Computer Communication Review. Vol. 20, No. 2, April 1990, pp. 30-34.

[IS8208] ISO. Information Processing Systems, Open Systems Interconnection, *X.25 Packet Level Protocol for Data Terminal Equipment*, ISO 8208 (CCITT X.25), 1990.

[IS8613] ISO . Information Processing, Text and Office Systems, *Office Document Architecture (ODA) and Interchange Format*, ISO 8613, 1988.

[IS8613AD] ISO. Information Processing, Text and Office Systems, *Office Document Architecture (ODA) and Interchange Format - Addendum 4: Security*, ISO 8613/DAD 4, 1990.

[IS8472] ISO. Information Technology, Security Techniques, *Modes of Operation for a 64-bit Block Cipher Algorithm*, ISO/IEC, ISO 8472, 1989.

[IS8473] ISO. Information Processing Systems, Open Systems Interconnection, *Protocol for Providing the Connctionless Mode Network Service (Internetwork Protocol)*, ISO 8473.

[IS8824-1] ISO. Information Technology. Open Systems Interconnection - Abstract Syntax Notation One (ASN.1): Specification of Basic Notation. ISO/IEC 8824-1.1993. Presently at the stage of draft.

[IS8824-2] ISO. Information Technology. Open Systems Interconnection - Abstract Syntax Notation One (ASN.1): Information Object Specification. ISO/IEC 8824-2.1993. Presently at the stage of draft.

[IS8824-3] ISO. Information Technology. Open Systems Interconnection - Abstract Syntax Notation One (ASN.1): Constraint Specification. ISO/IEC 8824-3.1993. Presently at the stage of draft.

[IS8824-4] ISO. Information Technology. Open Systems Interconnection - Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1. ISO/IEC 8824-1.1993. Presently at the stage of draft.

[IS9594-8] ISO . Information Processing Systems, Open Systems Interconnection, *The Directory - Authentication Framework.* ISO/IEC 9594-8 (Recommendation **X.509** 1993 E). 14 February 1993.

[IS10116] ISO. Information Technology, Security Techniques, *Modes of Operation for a N-bit Block Cipher Algorithm*, ISO/IEC, DIS 10116, 1990.

[IS10181-1] ISO. Information Technology, OSI Security Model, Part 1: Security Framework, ISO DP 10181-1.

[IS10181-1] ISO. Information Technology, OSI Security Model, Part 2: Authentication Framework, ISO DP 10181-1.

[ISN2559] ISO. JTC1/SC6/N2559: Draft Network Layer Security Protocol. Draft November 1990.

[ISN6285] ISO. JTC1/SC6/N6285: Draft Transport Layer Security Protocol. Draft November 1990.

[JM91] P. Janson and R. Molva. *Security in Open Networks and Distributed Systems.* Computer Networks and ISDN Systems 22, pp. 323-346, 1991.

[Kah67] D. Kahn. *The Codebreakers.* Macmillan Co.: New York. 1967. (Traduction française : *La Guerre des Codes Secrets.* Paris. InterEditions. 1980)

[Kal92] B. Kaliski. *The MD2 Message-Digest Algorithm.* RFC 1319. April 1992.

[Kal93] B. Kaliski. *Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services.* RFC 1424. RSA Laboratories. February 1993.

[Kar91] A. Karila. *Open Systems security - an Architectural Framework.* Doctoral Dissertation, Helsinki University of Technology. March 1991.

[Ken93a] S. Kent. *Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management.* RFC 1422. BBN. February 1993.

[Ken93b] S. Kent. *Understanding the Internet Certification System.* in the proceedings of INET'93, the International Networking Conference. Internet Society. San Francisco - CA. August 1993.

[KN93] J. Kohl and C. Neuman. *The Kerberos Network Authentication Service (V5).* RFC 1510. September 1993.

[KNT91] J. Kohl, C. Neuman and T. Ts'o. *The Evolution of the Kerberos Authentication Service.* in an IEEE Computer Society - June 1992, and also in the proceedings of EurOpen, an UNIX Distributed Open Systems in Perspective. Tromso, Norway. p. 295-313. May 1991.

[Knu73] D. E. Knuth. *The Art of Computer Programming, Vol. 3: Sorting and Searching.* Addison-Wesley. 1973.

[KO91] S.E. Kille and J.P. Onions. *The PP Manual, Version 6.* UCL, 1991.

[Kol87] N. Kolblitz. *Elliptic Curve Cryptosystems.* Mathematics of Computation, Vol. 48, No. , pp. 203-209, 1987.

[Kra86] E. Kranakis. *Primality and Cryptography.* Chichester/New York: John Wiley & Sons, 1986.

[KRRT91] S. E. Kille, C. Robins, M. Roe, and A. Turland. *The ISO Development Environment: User's Manual, Volume 5: QUIPU.* University College London. 1991.

[KSW92] P. Kirstein, D. F. Sadok, P. Williams. *PASSWORD R1.2: Document Services Requirements and Policy.* December 1992.

[LABW92] Lampson, B.; Abadi, M.; Burrows, M.; Wobber, E. *Authentication in distributed systems: theory and practice*. ACM Transactions on Computer Systems Vol: 10 Iss: 4 p. 265-31. November 1992.

[Lem79] A. Lempel. *Cryptology in transition*. ACM Computing Surveys, Vol. 11, No. 4, pp. 285-30. December 1979.

[Lin93] J. Linn. *Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures*. RFC 1421. DEC. February 1993.

[LLH89] C. Laih, J. Lee, L. Harn and Y. Su. *Linearly Shift Knapsack Public-Key Cryptosystem*. IEEE Journal on Selected Areas in Communications, Vol. 7, No. 4, pp. 534-547, May 1989.

[LM90] X. Lai and J. L. Massey. *A proposal for a New Block Encryption Standard*. in Advances in Cryptology - EUROCRYPT '90, a Workshop on the Theory and Applications of Cryptology, pp. 389-404, Berlin, 1991, Springer-Verlag.

[LS88] W. Lu, M.K. Sundarehan. *A Model For Multilevel Security In Computer Networks*. in proceedings of IEEE INFOCOM 88. USA. 1988.

[LS89] W. Lu, M.K. Sundareshan. *Secure Communication in Internet Environments: A Hierarchical Key Management Scheme for End-To-End Encryption*. IEEE Trans. Comm, 37, October 1989.

[Lu86] W-P. Lu. *Secure of Communication in Computer Networks*. Master Thesis. University of Arizona. 1986.

[Lub93] Christoph Lubbert. *PASSWORD R1.2: MHS Requirements and Policy*. February 1993.

[Lue93] J. Luehe. *PASSWORD R2.6: User Requirements*. March 1993.

[LW93] J. Luehe and W. Schneider. *PASSWORD R1.2: X.500 Requirements and Policy*. March 1993.

[Mas88] J. L. Massey. *An Introduction to Contemporary Cryptology*. Proceedings of the IEEE, Vol. 76, No. 5, pp. 533-549, May 1988.

[MD79] E. J. MacCauley and P. J. Dronyonski. *Ksos - the Design of a Secure Operation System*. In AFIPS National Computer Conference. pp. 345-353. Montvale, N. J. AFIPS press. June 1979.

[Men93] S. Mendes. *SecUcomx: A Security Architecture for Open Applications*. Technical Report EMSE, Septembre 1993.

[Mer78a] R. C. Merkle. *Secure communications over insecure channels*. Communications of the ACM, Vol. 21, No. 4, pp. 294-299, April 1978.

[Mer78b] R. C. Merkle and M. E. Hellman. *Hiding information and signatures in trapdoor knapsacks*. IEEE Transactions on Information Theory, Vol. 24, No. 5, September 1978, pp. 525-530.

[Mer82] R. C. Merkle. *Secrecy, Authentication, and Public Key Systems*. Ann Arbor: UMI Research Press, 1982.

[Mer89] R. C. Merkle. *One way hash functions and DES*. Advances in Cryptology - CRYPTO '89, proceedings of a Conference on the Theory and Applications of Cryptology Techniques, Santa Barbara, CA, August 1986, pp. 428-446. Berlin/New York: Springer-Verlag, 1989.

[Mes92] E. Messmer. *NIST stumbles on Proposal for Public-key Encryption*. Network World, Vol. 9, No. 30, July 27, 1992.

[MOI90] S. Miyaguchi, K. Ohta and M. Iwata. *128-bit hash function (N-Hash)*. in proceedings of SECURICOM' 90. March 1990.

[Mil85] V. S. Miller. *Use of Elliptic Curves in Cryptography*. in Advances in Cryptology - CRYPTO '85, proceedings of a Conference on the Theory and Applications of Cryptology, Santa Barbara, CA, August 1985, pp. 417-426. Berlin/New York: Springer-Verlag, 1986.

[Moc87a] P. Mockapetris. *Domain names - concepts and facilities*. RFC 1034. November 1987.

[Moc87b] P. Mockapetris. *Domain names - implementation and specification*. RFC 1035. November 1987.

[Moo88] J. H. Moore, *Protocol Failures in Cryptosystems*. Proceedings of the IEEE, Vol. 76, No. 5, May 1988, pp. 594-602.

[MRW88] C. Mitchell, D. Rush and M. Walker. *A Secure Messaging Architecture Implementing the X.400-1988 Security Features*. Technical Memo. Hewlett-Packard Company. November 1988.

[MT79] R. Morris and K. Thompson. *Password Security: a case history*. Communications of the ACM, Vol. 22, No. 11, pp. 594-597, November 1979.

[MW88] C. Mitchell and M. Walker. *Solutions to the Multidestination Secure Electronic Mail Problem*. Computers & Security, Vol. 7, No. 5, pp. 483-488, 1988.

[NADF91] The North American Directory Forum. *A Naming Scheme for c=US*. RFC 1218. April 1991.

[NBS77] National Bureau of Standards, Federal Information Processing Standards (FIPS) Publication 46: *Annoucing the Data Encryption Standard*, January 15, 1977.

[NBS80] National Bureau of Standards, Federal Information Processing Standards (FIPS) Publication 81: *DES Modes of Operation*, December 2, 1980.

[NBS81] National Bureau of Standards, Federal Information Processing Standards (FIPS) Publication 74: *Guidelines for Implementing and Using the NBS Data Encryption Standard*, April 1, 1981.

[NIST92a] National Institute of Standards and Technology (NIST). *Annoucement and Specifications for a Digital Signature Standard (DSS)*. August 19, 1992.

[NIST92b] National Institute of Standards and Technology (NIST). *Annoucement and Specifications for a Secure Hash Standard (SHS)*. January 22, 1992.

[NIST92c] National Institute of Standards and Technology (NIST). *The Digital Signature Standard, proposal and discussion.* Communications of the ACM, Vol. 35, No. 7,pp. 36-54, July 1992.

[NIST93] National Institute of Standards and Technology (NIST) News. *Clipper Announce*. The White House, Office of the Press Secretary. NIST News. 16 April 1993.

[NS78] R. M. Needham and M. D. Schroeder. *Using Encryption for Authentication in Large Networks of Computers*. Communications of the ACM, Vol. 21, No. 12, pp. 993-999, December 1978.

[NS87] R. M. Needham and M. D. Schroeder. *Authentication Revisited.* ACM Operating Systems Review, Vol. 21, No. 1, January 1987, pp.7.

[Odl84] A. M. Odlyzko. *Discrete logarithms in finite fields and their cryptographic significance*. in Advances in Cryptology: Proceedings of EUROCRYPT 84, a Workshop on the Theory and Application of Cryptology, Paris, France, April 1984, pp. 224-314. Berlin/New York: SpringerVerlag.

[OT89] E. Okamoto and K. Tanaka. *Key Distribution System based on Identification Information.* IEEE Journal on Selected Areas in Communications, Vol. 7, No. 4, pp. 481-485, May 1989.

[OIW91] OSI Implementors' Workshop. *Implementation Agreements for Open Systems Interconnection Protocols : Part 8 - Message Handling Systems*. December 1991.

[PA94] J. Postel and C. Anderson. *White Pages Meeting Report*. RFC 1588. February 1994.

[Par90] G. Parulkar. *The Next Generation of Internetworking.* ACM Computer Communication Review. Vol. 20, No. 1, January 1978, pp. 18-43.

[PK79] G. L. Popek and C. S. Kline, *Encryption and Secure Computer Networks.* ACM Computing Surveys, Vol. 11, No. 4, pp. 331-356, December 1979.

[PKK+79] G. L. Popek, M. Kampe, C. S. Kline, A. Soughton, M. Urban and E. J. Walton. *UCLA Secure Unix.* In AFIPS National Computer Conference. pp. 355-364. Montvale, N. J. AFIPS press. June 1979.

[PKCS #1] RSA Laboratories. PKCS #1: *RSA Encryption Standard.* Version 1.5, November 1993.

[PKCS #3] RSA Data Security, Inc. PKCS #3: *Diffie-Hellman Key-Agreement Standard.* Version 1.4, November 1993.

[PKCS #5] RSA Data Security, Inc. PKCS #5: *Password-Based Encryption Standard.* Version 1.5, November 1993.

[PKCS #6] RSA Laboratories. PKCS #6: *Extended-Certificate Syntax Standard.* Version 1.5, November 1993.

[PKCS #7] RSA Laboratories. PKCS #7: *Cryptographic Message Syntax Standard.* Version 1.5, November 1993.

[PKCS #8] RSA Data Security, Inc. PKCS #8: *Private-Key Information Syntax Standard.* Version 1.2, November 1993.

[PKCS #9] RSA Laboratories. PKCS #9: *Selected Attribute Types.* Version 1.1, November 1993.

[PKCS #10] RSA Laboratories. PKCS #10: *Certification Request Syntax Standard.* Version 1.0, November 1993.

[Pos82] J.B. Postel. *Simple Mail Transfer Protocol.* RFC 821. August 1982.

[QC82] J.-J. Quisquater and C. Couvreur. *Fast decipherment algorithm for RSA public-key cryptosystem.* Electronics Letters, Vol. 18, No. 21, pp. 905-907, October 14, 1982.

[QG89] J.-J. Quisquater, L. Guillou. *How to Explain Zero-knowledge Protocols for your Children.* Advances in Cryptology - CRYPTO '89, proceedings of a Conference on the Theory and Applications of Cryptology Techniques, Santa Barbara, CA, August 1986, pp. 628-631. Berlin/New York: Springer-Verlag, 1989.

[Rab78] M. O. Rabin. *Digitalized signatures* in R. A. DeMillo, D. P. Dobkin, A. K. Jones, and R. J. Lipton, Eds., Foundations of Secure Computation, pp. 155-168. New York: Academic Press, 1978.

[Rab79] M. O. Rabin. *Digitalized signatures and public-key functions as intractable as factorization.* MIT Laboratory for Computer Science, Technical Report LCS/TR-212, January 1979.

[RD90] J. T. Robinson and M. V. Devarakonda. *Data Cache Management using Frequency-based Replacement.* in proceedings of ACM SIGMETRICS '90. Boulder , CO. pp. 134-142. May 1990.

[RHAL92] R. L. Rivest, M. Hellman, J. Anderson and J. Lyons. *Responses to NIST's Proposal.* Communications of the ACM. Vol. 35, No. 7, pp. 41-54, July 1992.

[Riv90] R. L. Rivest, *The MD4 Message Digest Algorithm.* in Advances in Cryptology: Proceedings of CRYPTO'90, a Workshop on the Theory and Application of Cryptology, Santa Barbara, CA, August 1990, pp. 303-311, Berlin/New York: Springer-Verlag.

[Riv92] R. L. Rivest, *The MD5 Message Digest Algorithm.* RFC 1321. Internet Activities Board, April 1992.

[Roe92] M. Roe. PASSWORD R2.5: Certification Authority Requirements, November 1992.

[Roe93] M. Roe. PASSWORD R1.2: ACSE Requirements and Policy, January 1993.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.* Communications of the ACM, Vol. 21, No. 2, pp. 120-127, February 1978.

[RSK92] M. Roe, W. Schneider, P. Kirstein, C. Huitema and P. Williams. PASSWORD R1.1: Service Requirements, August 1992.

[Rue91] R. A. Rueppel. *A Formal Approach to Security Architectures.* in Advances in Cryptology - EUROCRYPT '91, a Workshop on the Theory and Applications of Cryptology, pp. 387- 397. Berlin, 1992, Springer-Verlag.

[Sat89] M. Satyanarayan. *Integrating Security in a Large Distributed System.* ACM Transactions on Computer Systems. Vol. 7. No 3, 247, 1989. (verificar vol et num)

[SB88] M. E. Smid and D. K. Branstad. *The Data Encryption Standard: past and future.* Proceedings of the IEEE, Vol. 76, No. 5, pp. 550-559, May 1988.

[Sch89] C. P. Schnorr. *Efficient Identification and Signatures for Smart Cards.* Advances in Cryptology - CRYPTO'89, a Workshop on the Theory and Applications of Cryptology, Santa Barbara, CA, August 1989, pp. 239-251. Berlin/New York: Springer-Verlag.

[SDN701] SDNS Protocol and Signaling Working Group. SDN 701: Message Security Protocol. August 1989.

[SDN702] SDNS Protocol and Signaling Working Group. SDN 702: Directory Specifications for Utilisation with the SDNS Message Security Protocol. August 1989.

[Sha84] A. Shamir. *Identity-based Cryptosystems and Signature Schemes.* in Lecture Notes in Computer Science Vol. 196: Advances in Cryptology: Proceedings of CRYPTO 84, a Workshop on the Theory and Application of Cryptographic Techniques, Santa Barbara, CA, August 19-22, 1984, pp. 47-53. Berlin/New York: Springer-Verlag, 1985.

[Sha49] C. Shannon. *Communication Theory of Secrecy Systems.* Bell System Technical Journal. Vol 28, Oct. 1949, pp. 656-715.

[SHF89] E. H. Spafford, K. Heaphy and D. Ferbrache. *Computer Viruses: Dealing with Electronic Vandalism and Programmed Threats.* ADAPSO, Arlington, VA 1989.

[Sim79] G. J. Simmons. *Symmetric and Asymmetric Encryption.* ACM Computing Surveys, Vol. 11, No. 4, pp. 305-330, December 1979.

[SM87] A. Shimizu and S. Miyaguchi. *Fast Data Encipherment Algorithm FEAL.* in Advances in Cryptology: Proceedings of EUROCRYPT 87, a Workshop on the Theory and Application of Cryptology, pp. 267, Paris, France, April 1987, Berlin/New York, Springer-Verlag.

[Sme91] B. Smetaniuk. *Distributed Operation of the X.500 Directory.* Computer Networks and ISDN Systems 21, pp. 17-40, 1991.

[Smi81] A. J. Smith. *Optimization of I/O systems by cache disks and file migration: a survey.* Performance Evaluation. Vol. 1 No. 4. pp. 249-262. November 1981.

[Spa88] E. H. Spafford. *The Internet Worm Program: an Analysis.* Techinical report CSD-TR-823. Department of Computer Sciences. Purdue University. December 1988.

[Spa91] E. H. Spafford. *The Internet Worm Incident.* Techinical report CSD-TR-933. Department of Computer Sciences. Purdue University. September 1991.

[SRC84] J. Saltzer, D. Reed and D. Clark. *End-to-End Arguments in System Design.* ACM Transactions on Computer Systems, Vol. 2. pp. 195-206. 1984.

[Tan89] A. S. Tanenbaum. *Computer Networks.* Second Edition. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[Ter87] D. B. Terry. *Caching Hints in Distributed Systems.* IEEE Transactions on Software Engineering, Vol. 13, No. 1, pp. 48-54, January 1987.

[Tou93] M. J. Toussaint. *A New Method for Analaysing the Security of Cryptographic Protocols.* IEEE Journal on Selected Areas in Communications, Vol. 11, No. 5, pp. 702-714, June 1993.

[Tsu92] G. Tsudik. *Message Authentication with One-Way Hash Functions.* in the proccedings of IEEE INFOCOM'92 a Conference on Computer Communications. May 1992.

[Val93] L. Valois. *Le Chiffrement, une arme pour protéger ses données.* 01 Informatique. 3 Septembre 1993.

[TS94] Telesystèmes. *Le point sur la sécurité logique.* Supplément aux affaires à l'honneur No 39. 1994.

[VK83] V. L. Voydock and S. T. Kent. *Security Mechanisms in High-Level Network Protocols.* ACM Computing Surveys, Vol. 15, No. 2, pp. 135-171, June 1983.

[WFS93] C. Weider, J. Fullton and S. Spero. *Architecture of the Whois++ Index Service.* Internet Draft. WNILS Working Group. IETF. October, 1993

[Wil92] P. Williams et al.. *OSISEC User's Manual, Volume 0: Package Concepts.* Universty College London, July 1992.

[Wiz94] Wizman, A. *Internet : dans les filets du Minitel mondial.* Actuel. . No. 36-37. pp. 54-61. Janvier 1994.

[WR90] S. Wilbur and M. Roe. *A Study of Security in Directory and Eletronic Mail Services.* University College London, October 1990

[X.200] CCITT. Recommendation X.200: Reference Model of Open Systems Interconnection for CCITT Applications (**IS7498**),1984.

[X.208] CCITT. Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1), 1988.

[X.209] CCITT. Recommendation X.209: Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1), 1988.

[X.217] CCITT. Recommendation X.217. Réseaux de Communication de Données. *Définition de Service pour l'Element de Service de Contrôle d'Association* (ACSE), 1992.

[X.227] CCITT. Recommendation X.227. Réseaux de Communication de Données. *Spécification du Protocole en Mode Connexion Applicable à l'Element de Service de Contrôle d'Association* (ACSE), 1992.

[X.219] CCITT. Recommendation X.219. Data Communication Networks. *Remote Operations : Model, Notation and Service Definition* (ROSE), 1988.

[X.229] CCITT. Recommendation X.229. Data Communication Networks. *Remote Operations : Protocol Spécification* (ROSE), 1988.

[X.40088] CCITT. Recommendation X.400: *Message Handling System and Service Overview*, 1988.

[X.402] CCITT. Recommendation X.402: *Message Handling System - Overall Architecture, 1988.*

[X.411] CCITT. Recommendation X.411: *Message Handling Systems - Message Transfer System: Abstract Service Definition and Procedures*, 1988.

[X.413] CCITT. Recommendation X.413: *Message Handling Systems - Message Store: Abstract Service Definition*, 1988.

[X.435] CCITT. Recommendation X.435: *Message Handling Systems - EDI Messaging System: Protocol Specifications,* 1991.

[X.500] CCITT. Recommendation X.500: *The Directory - Overview of Concepts, Models and Services*, 1988.

[X.50093] CCITT. Recommendation X.500: *The Directory - Overview of Concepts, Models and Services*, 1993.

[X.509] CCITT. Recommendation X.509: *The Directory - Authentication Framework*, 1988.

[X.511] CCITT. Recommendation X.511: *The Directory - Abstract Service Definition*, 1988.

[X.51193] CCITT. Recommendation X.511: *The Directory - Abstract Service Definition*, 1993.

[X.520] CCITT. Recommendation X.520: *The Directory - Selected Attribute Types*, 1988.

[X.521] CCITT. Recommendation X.521: *The Directory - Selected Object Classes*, 1988.

[X.800] CCITT. Recommendation X.800. Data Communication Networks : Open Systems Interconnection (OSI); Security, Structure and Applications. *Security Architecture for Open Systems Interconnection for CCITT Applications* (**ISO 7498-2**), 1991.

[Zim93] P. Zimmermann. *Pretty Good Privacy User's Guide - Vol. II: Special Topics*. Phil's Pretty Good Software. Jun 93.

LISTE DES ILLUSTRATIONS

Figure **Page**

Figure 1. Modèle de Chiffrement Symétrique....24
Figure 2. Modèle de Chiffrement Asymétrique....28
Figure 3. Signatures Numériques....38
Figure 4. Exemple Fictif de Hiérarchie de CA....52
Figure 5. Exemple de Distribution non Hiérarchique de CA....52
Figure 6. Authentification Simple Protégée....55
Figure 7. Exemple Fictif du Modèle de Certification PEM....63
Figure 8. Certification de Clés Publiques dans PGP....66
Figure 9. Modèle de Certification PGP....67
Figure 10. Modèle de Certification Chimaera....69
Figure 11. Modèle Fonctionnel de l'Annuaire X.500....87
Figure 12. Modèle Fonctionnel du MHS....92
Figure 13. Modèle de Certification Password....108
Figure 14. Génération , Stockage et Certification de Clés....131
Figure 15. Le Fonctionnement du Gestionnaire de Secrets....133
Figure 16. Modèle Fonctionnel du DAS....149
Figure 17. Rapport entres les éléments de l'architecture et X.500....152
Figure 18. Rapport entre les éléments de l'architecture et X.400....158
Figure 19. Mécanisme de Preuve de Remise....176
Figure 20. Service de Multi-Signatures....178
Figure 21. Utilisateur confronté aux Graphes de Certification....190
Figure 22. Structure d'un KMD....194
Figure 23. Rapport entre une CA, son répondeur et les utilisateurs....202

Figure 24. Rapport entre le DAS et les répondeurs de CA....202
Figure 25. Intégration des Répondeurs de CA dans le DAS....202
Figure 26. Rapport entre une PCA et une CA dans le modèle PEM....204
Figure 27. Rapport entre deux CA de Haut Niveau dans le Modèle Chimæra....204
Figure 28. Interprétation d'un Certificat X.509....205
Figure 29. Rapport entre les Politiques de Certification, les Critères Locaux et le Fonctionnement du DAS....208
Figure 30. Attribution de Politique de Certification....212
Figure 31. Attribution d'Identificateurs d'Objet aux Modèles de Certification....217
Figure 32. Modèle Fonctionnel du Service DAS....235
Figure 33. Environnement Distribué des DAS....238
Figure 34. Modèle Distribué du Service DAS....240
Figure 35. Navigation d'une Opération entre DAS....244
Figure 36. Exemples de Chemins de Certification....249

GLOSSAIRE

ACSE	Association Control Service Element
API	Application Programming Interface
ASE	Application Service Element
ASN.1	Abstract Syntax Notation One
BBS	Bulletin Board System
BER	Basic Encoding Rules
CA	Certification Authority
CBC	Cipher Block Chaining
CFB	Cipher FeedBack
CIC	ContentIntegrityCheck
CLNP	ConnectionLess Network Procotol
CRC	Cyclic Redundancy Check
CRL	Certificate Revocation List
CS	Certification System (partie integrante de l'architecture SecUcomx)
DAP	Directory Access Protocol
DAS	Data Authentication Server ou Serveur de Données pour l'Authentification
DASAP	Data Authentication Service Access Protocol
DASIP	Data Authentication Server Interaction Protocol
DEK	Data Encrypting Key
DES	Data Encryption Standard

DIB	Directory Information Base
DIT	Directory Information Tree
DN	Distinguished Name
DNS	Internet Domain Name System
DSA	Digital Signature Algorithm
DSA	Directory Service Agent
DSP	Directory Service Protocol
DSS	Digital Signature Standard
DUA	Directory User Agent
ECB	Electronic Code Book
EDI	Electronic Data Interchange
EDT	Electronic Data Transfer
FEAL	Fast Encryption ALgorithm
GS	Gestionnaire de Secrets (partie integrante de l'architecture SecUcomx)
IAB	Internet Architecture Board
ICV	Integrity Check Vector
IDEA	International Data Encryption Algorithm
IETF	Internet Engineering Task Force
IHM	Interface Homme-Machine (
IK	Interchange Key
IPES	Improved Proposed Encryption Standard
IPN	InterPersonal Notification
IPRA	Internet Policy Registration Authority
ISDN	Integrated Services Digital Network
KMD	Key Management Domain ou Domaine de Gestion de Clés
KMP	Key Management Policy ou Politique de Gestion de Clés
KS	Key Generation Service (partie integrante de l'architecture SecUcomx)
LAN	Local Area Network
MAN	Metropolitan Area Network
MDC	Modification Detection Code

MHS	**M**essage **H**andling **S**ystem
MIC	**M**essage **I**ntegrity **C**heck
MIME	**M**ultipurpose **I**nternet **M**ail **E**xtensions
MOA	**M**essage **O**rigin **A**uthentication Check
MS	**M**essage **S**tore
MTA	**M**essage **T**ransfer **A**gent
MTS	**M**essage **T**ransfer **S**ystem
NSAP	**N**etwork **S**ervice **A**ccess **P**oint
ODA	**O**pen **D**ocument **A**rchitecture
OFB	**O**utput **F**eed**B**ack
P1	Protocole de Transfert de Messages
P2	Protocole de Messages Interpersonnels
P3	Protocole d'Accès au MTS
P7	Protocole d'Accès au MS
PASSWORD	**P**iloting **A**uthentication and **S**ecurity **S**ervices **w**ithin **O**SI Applications for the **R**esearch and **D**evelopment Community in Europe
PCA	**P**olicy **C**ertification **A**uthority
PEM	**P**rivacy **E**nhanced **M**ail.
PGP	**P**retty **G**ood **P**rivacy
PIN	**P**ersonal **I**dentification **N**umber
PKCS	**P**ublic-**K**ey **C**ryptography **S**tandards
PoD	**P**roof **o**f **D**elivery
PSRG	**P**rivacy and **S**ecurity **R**esearch **G**roup
RDN	**R**elative **D**istinguished **N**ame
RFC	**R**equest **f**or **C**omments
RNG	**R**andom **N**umber **G**enerators
ROSE	**R**emote **O**pérations **S**ervice **E**lement
RSA	Algorithme de chiffrement asymétrique développé par **R**ivest **S**hamir et **A**dleman en 1978.
SHS	**S**ecure **H**ash **S**tandard
SNMP	**S**ecure **N**etwork **M**anagement **P**rotocol

SOLO	Simple Object LOok-up
TLCA	Top Level CA
TSAP	Transport Service Access Point
UA	User Agent
WAN	Wide Area Network
WWW	World Wide Web