Traçabilité sécurisée embarquée : authentification autonome d'objets et de systèmes embarqués

Abdourhamane Idrissa

Ecole Doctorale ED488 "Sciences, Ingénierie, Santé"

Thése présentée en vue de l'obtention du diplôme de
Docteur de l'Université Jean Monnet
Spécialité Image, Vision et Signal

Traçabilité - Sécurisée - Embarquée :
Authentification autonome d'objets et de systèmes embarqués

par

Abdourhamane Idrissa

Soutenance publique le 20 Septembre 2012 devant le jury composé de :

Président du jury :	**Viktor Fisher** Professeur, Université J. Monnet de Saint-Etienne
Rapporteurs :	**Marine Minier** Maître de conférences-HDR, INSA de Lyon
	Lionel Torres Professeur, Université de Montpellier 2
Examinateurs :	**Pascal Lafourcade** Maître de conférences, Université J. Fourier de Grenoble
	Rolland Gillard Professeur honoraire, Institut Fourier de Grenoble
Directeur :	**Thierry Fournel** Professeur, Université J. Monnet de Saint-Etienne
Co-directeur :	**Alain Aubert** Maître de conférences, Université J. Monnet de Saint-Etienne

à la mémoire de ma mère,
à mon père ...

Remerciements

Je ne saurais commencer ces lignes de remerciement sans citer la Région Rhône-Alpes qui a supporté mes travaux de thèse à travers le cluster ISLE. En effet, cette thèse vient en aval d'une autre thèse également financée par la Région Rhône-Alpes à travers le même cluster ISLE au sein du projet « Sécurité Informatique et Sûreté des Systèmes ». Il s'agit de la thèse de M. Jean Lancrenon codirigée par M. Roland Gillard (Institut Fourier) et M. Xavier-François Roblot (Institut Camille Jordan) dans le cadre de l'Action MorphoSecure, dédiée aux problématiques d'authentification d'objets selon une approche inspirée de celle développée par la société Signoptic. Le sujet de la présente thèse, proposé par M. Thierry Fournel, vise non plus une authentification à distance mais une authentification locale avec vérification humaine du lecteur. Le rapprochement avec le projet «SEMBA» du même cluster et avec M. Alain Aubert ont permis de définir les architectures cibles pertinentes. Cette collaboration réussie m'a permis de réaliser mes travaux dans un cadre de recherche assez collaboratif.

Je tiens à remercier tous les membres du jury qui ont accepté d'évaluer mon travail. Un très grand merci à mes rapporteurs : Mme Marine MINIER et M. Lionel TORRES, pour le temps qu'ils ont consacré à rapporter cette thèse et toute l'attention et la pertinence qui ont été les leurs. Merci à M. Viktor FISHER d'avoir accepté de présider le jury de thèse. Toute ma reconnaissance à l'endroit de M. Rolland Gillard et M. Pascal Lafourcade qui ont accepté d'examiner mes travaux. Ils n'ont jamais hésité à nous accueillir à Grenoble pour discuter avec nous sur nos travaux de recherche.

Je remercie chaleureusement M. Thierry FOURNEL et M. Alain AUBERT d'avoir été présents tout au long de ces années de thèse, et cela en semaine et comme en weekend. Je souhaite vous remercier, à travers ces mots, pour toute l'attention dont vous m'avez témoigné. Je vous saurai gré de m'avoir permis de réaliser cette thèse et je vous serai infiniment reconnaissant pour votre confiance... Un sincère remerciement aux membres de l'équipe SES (Secure Embedded Systems), aux anciens comme aux nouveaux, merci donc de m'avoir accepté parmi vous. Vos conseils et nos différents échanges ont été fructueux pour mes travaux de thèse, je vous en remercie. Je vous saurai gré pour cette bonne ambiance qui règne au sein de l'équipe, je garderai de bons souvenirs de ma vie parmi vous. Je remercie également chacun des membres du Laboratoire Hubert Curien, sans oublier l'équipe de football du Jeudi.

Je profite de cette page pour exprimer ma gratitude envers l'Etat malien qui a soutenu financièrement mes études universitaires à travers le « programme 300 jeunes cadres pour le Mali ». Cette bourse a été une chance pour moi, l'opportunité de faire des études supérieures en toute autonomie. Un grand merci à mes amis (indénombrables), à mes camarades de la CPD (Cellule Pour le Développement) pour leur disponibilité et leur fraternité. Je remercie particulièrement mes amis de mon «grin» de Saint-Etienne, vous avez étés disponibles pour faciliter mon passage sur Saint-Etienne.

Je tiens enfin à remercier ma grand-mère Sorho Bazzi, mes frères, mes soeurs, mes cousins et toute ma famille, de Gabero à Bamako, pour leur soutien et leur confiance sans lesquels je n'aurai jamais pu tenir la distance. Une pensée très émue pour ceux qui nous ont quittés, notamment mon oncle feu Minkeila Bellah qui m'a tant conseillé.

En ce moment, je ne pourrai terminer cette page sans avoir une prière pour mon Gao natal et pour mon pays, le Mali. Ainsi, j'espère que le Mali retrouvera, dans un futur proche, sa grandeur et redeviendra une terre africaine de progrès et d'espérance !

Résumé

Résumé

L'authentification homme-machine est une problématique largement développée pour les télécommunications. Une authentification dans le sens "machine-homme" permettra d'assurer l'utilisateur humain assermenté du fonctionnement intègre d'une machine lors, par exemple, d'une session de vote électronique ou d'une vérification d'objet en traçabilité sécurisée. Cette thèse se focalise sur la traçabilité sécurisée sans accès (systématique) à un canal de communication.

Nous décrivons différentes techniques d'authentification de produits manufacturés en nous concentrant sur une méthode de caractérisation de motifs imprimés. Pour effectivement authentifier un objet, nous montrons qu'un agent vérifieur doit s'assurer de l'intégrité du tiers et du système électronique utilisée pour la vérification.

Cependant l'authenticité du système électronique lui-même reste à vérifier. La question que nous adressons alors est la suivante : comment un être humain peut-il se convaincre de l'intégrité et de l'authenticité d'un système embarqué dans un mode hors ligne ? Nous définissons deux familles de solutions. Dans la première, l'utilisateur fait appel, pour les calculs, à un dispositif auxiliaire tandis que dans la seconde l'utilisateur ne fait usage que d'un papier et d'un crayon. Pour chacune des deux familles, nous proposons un protocole d'authentification d'un système embarqué dont la puce, typiquement un FPGA ou un microcontrôleur, dépend de la configuration ou de la programmation d'une mémoire RAM.

Mots-clés :

Authentification, Authentification d'objet, Traçabilité sécurisée, Protocole d'authentification, Authentification Machine-Homme, FPGA, Microcontrôleur.

Abstract

"Human-to-Machine" authentication is widely developed for modern telecommunications. A "Machine-to-Human" authentication will ensure the trusted human user about the integrity of the machine, for example during an electronic voting session or object verification in secure traceability. This work is focused on secure traceability without any systematic access to a communication network.

We depict different technics for goods authentication and we focus on a method based on the characterization of printed patterns. To completely authenticate an object, we show that a human verifier has to be confident in the integrity of the third party and the electronic system involved in the verification phase.

However, the authenticity of the electronic system itself has also to be verified. We address here the following question : how a human being can convince himself about the integrity and the authenticity of an embedded system in an off-line environment ? We propose two groups of solutions. In the first one, an auxiliary electronic device is used to perform computing operations. In the second one, the human capability (memory and computational abilities) is exploited. In each group, we propose a protocol to authenticate embedded systems for which the chip (typically an FPGA (Field Programmable Gate Array) or a microcontroller) is initialized according to the configuration or programming of its RAM memory.

Table des matières

Introduction

La contrefaçon n'épargne aujourd'hui aucun secteur manufacturier. Les progrès de la technologie et son accessibilité tendent à amplifier le phénomène tout en permettant aux contrefacteurs d'approcher une certaine qualité de reproduction. La détection de telles copies peut exiger des moyens sophistiqués de caractérisation des objets. Un système embarqué d'authentification d'objet est alors nécessaire et intervient comme une aide à la décision pour un agent contrôleur humain.

Dans ce contexte de traçabilité sécurisée, cet agent va constater le résultat du système mais comment être sûr du bon fonctionnement du dispositif ? Plus précisément quel procédure suivre pour s'assurer au préalable que le système embarqué est en conformité avec les spécifications associées à son identifiant ? La question du protocole d'authentification du dispositif se pose tout particulièrement lorsque pour des raisons de sécurité des infrastructures, géographiques ou tout simplement pratiques où aucune télécommunication n'est possible.

Cette problématique devient même une nécessité légale dans le domaine du vote électronique. La machine ne peut être branchée à un quelconque réseau de communication et il revient au Président de session de suivre une procédure de test de son fonctionnement. Il est en effet crucial de rendre l'humain actif dans ce type de vérification alors que c'est en définitive lui qui supervise les opérations. Si des solutions partielles existent en authentification machine-homme c'est à dire de la machine par l'homme, une solution sûre impliquant de façon active voire exclusive un humain vérifieur reste à imaginer.

Dans ces travaux, nous commençons par appréhender la notion d'authentification d'objets et les différents contextes de vérification, avant d'illustrer la notion d'authentification locale par le biais des "e-tickets" (chapitre I). Au chapitre II, nous introduisons la notion d'authentification autonome qui inclut la vérification du système embarqué destiné à tester les objets. Les protocoles proposés comme solutions d'authentification autonome de système embarqué, sont détaillés au chapitre IV après avoir dressé un état de l'art sur l'authentification des systèmes embarqués au chapitre III.

Première partie

Traçabilité sécurisée et autonome

Chapitre I

L'authentification d'objets

I.1 Introduction : identification et authentification

Ce chapitre a pour objectif de préciser la notion d'authentification, en particulier dans le cas des objets. Cette notion sous-tend celle d'identification. En effet dès que plusieurs entités interviennent avec des rôles différents dans un protocole, il est nécessaire de les distinguer individuellement et/ou par groupe d'appartenance.
Ainsi par exemple pour authentifier une personne, on lui demandera de décliner au préalable son identité. Lorsqu'on établit une communication avec un service clientèle, une banque ou un opérateur téléphonique pour effectuer une télé-opération, une des premières informations à renseigner est généralement le numéro de compte du titulaire. Cette information permet une **identification**. La requête d'une donnée connue d'un nombre restreint d'individus ou d'entités, comme une date d'anniversaire ou un mot de passe, relève elle de l'**authentification** proprement dite, destinée à vérifier l'identité déclarée. De même, avant de se poser la question de savoir si un billet de banque est un faux, une simple lecture visuelle indique immédiatement la valeur supposée du billet tandis qu'un examen de cette valeur va déterminer si elle est factice ou non.

Pour pouvoir identifier un objet ou une personne, on lui associe des éléments d'identification connus (connus dans le sens où ces éléments ne sont pas supposés secrets) qui permettront de le distinguer. Ces éléments appelés **identifiants** constituent alors l'**identité** de l'objet ou de la personne. Par exemple, la valeur faciale d'un billet de banque est un identifiant de son type (5 euros, 1 dollar, etc). Chaque billet admet aussi un numéro unique qui permet, si nécessaire, de l'identifier parmi tous les billets du même type. Dans l'exemple de l'appel d'un service clientèle en ligne, les éléments d'identification d'un client sont : son numéro de compte (ou son numéro de téléphone pour un opérateur téléphonique) et son nom, souvent complétés par un numéro de dossier-client (générer à l'enregistrement du client). Chacune de ces informations peut suffire à identifier un client, à condition qu'elle soit unique.

Les éléments d'identification d'une entité donnée se caractérisent par le fait qu'ils

sont connus par toutes les entités avec lesquelles elle peut interagir. En milieu de confiance i.e. lorsque toutes les entités susceptibles de jouer un rôle dans un protocole sont honnêtes, les identifiants suffisent à authentifier l'entité déclarée. Pour contrer une entité malveillante qui essaierait d'usurper l'identité d'une autre, il est nécessaire de prévoir un mécanisme sûr de vérification des identifiants, c'est à dire une **authentification**.

Concernant l'authentification d'objets, elle est devenue centrale dans nos organisations socio-économiques que ce soit au niveau des populations, des industriels, des fournisseurs ou des consommateurs. La traçabilité des produits manufacturés est une forme d'identification de ces produits tout au long de leur fabrication, acheminement ou usage. Une traçabilité sécurisée consiste à apporter une procédure et des moyens techniques visant à réaliser leur authentification, plus précisément celle du constituant et des indications fournies (origine, ...).

Seule une procédure rigoureuse peut constituer une aide à la décision quant à l'authenticité du produit. Les moyens déployés pour sa protection vont en pratique dépendre de sa valeur marchande, du type de produit, des conditions dans lesquelles la vérification peut s'effectuer. L'ajoût d'un dispositif anti-contrefaçon peut induire un coût supplémentaire, en terme de temps et/ou de moyens. Par exemple, rares sont les commerçants qui cherchent à authentifier les billets de $5, 10$ ou de 20 euros (or selon la BCE, les billets les plus contrefaits demeurent les coupures de 20 et 50 euros [Ban12]).

Dans ce chapitre, nous définissons la notion d'authentification dans le cadre de la traçabilité sécurisée. Nous proposons ensuite, à travers la littérature sur l'authentification biométrique, une classification des processus d'authentification d'objets en fonction des ressources et du mode de vérification de l'identité des objets.

I.2 Notion d'authentification dans le cas des objets

Nous généralisons ici la notion d'authentification à tout processus qui permet à un instant donné, d'avoir une confirmation de l'identité d'une entité aussi diverse qu'une personne humaine, un produit manufacturé, une machine électronique ou une application logicielle. Cela va nous permettre d'appréhender tout mécanisme de lutte contre la contrefaçon de produits sous l'angle d'un processus d'authentification d'objets.

I.2.1 Définition

Nous définissons un processus d'authentification comme tout mécanisme qui à un temps t, permet à une entité B de vérifier l'*état* d'une entité A, par rapport à un *état de référence connu de* A. On dit alors que B authentifie A, ou que A s'authentifie auprès de B. L'entité B est appelé **vérifieur** et A **l'entité à authentifier**.

L'**état de** A, que nous notons $\mathcal{E}^A$, est défini par l'**identité de** A (c'est à dire les identifiants de A) plus un **élément de vérification** V^A.

L'élément de vérification est basé soit sur une information que possède A et que B peut vérifier, soit sur une caractéristique particulière de l'entité A que B peut vérifier.

Nous appelons **phase d'enregistrement** d'un processus d'authentification, l'étape d'enregistrement/ référencement de l'état $\mathcal{E}_r^A$ d'une entité A, et **phase de vérification** l'étape réalisée par l'entité vérifieur B consistant à vérifier l'état de l'entité supposée être A par comparaison avec l'état de référence de l'entité A.

I.2.2 Exemples

Cette section et le tableau I.1 donnent quelques exemples de processus d'authentification.

I.2.2.1 Appel d'un service clientèle

Comme nous l'avons dit précédemment, toute communication d'une personne auprès d'un service clientèle exige une identification de l'appelant par son interlocuteur qui peut être soit une personne humaine soit un système automatisé. Lorsque l'objet de la communication le requiert, une phase de vérification (de l'identité du client) fait suite à son identification. Selon la définition ci-dessus, cette vérification est la phase de vérification d'un processus d'authentification entre le client appelant (l'entité A) et le destinataire (l'entité B). Les identifiants du client sont généralement : un numéro de compte, un nom, un prénom, une adresse et un numéro de dossier-client. L'élément de vérification de l'identifié peut alors être un mot de passe, un code PIN (Personal Identification Number), ou une date d'anniversaire.

I.2.2.2 Détection de faux billets

La détection de faux billets de banque par un humain ou une machine, définit la phase de vérification du billet considéré comme entité ayant une valeur fiduciaire donnée (inscrite sur ses faces). La phase d'enregistrement est elle définie par la banque centrale émettrice. Pour l'oeil de l'utilisateur, l'identifiant du billet est sa valeur faciale. Par contre les éléments de vérification sont nombreux et variés concernant le marquage d'objets). Ils font appel à des moyens plus ou moins sophistiqués.

I.2.2.3 Accès à un ordinateur personnel

Contrairement à un ordinateur qui est mis à la disposition du public, un ordinateur personnel a un accès restreint aux utilisateurs régis par l'administrateur où un unique couple «Nom d'Utilisateur & Mot de passe» (*login* & *password*) définit les accès et les droits sur la machine de l'utilisateur en question. Le processus de contrôle d'accès est alors un processus d'authentification : l'ordinateur vérifie l'identité de l'utilisateur. L'initialisation du compte utilisateur (sous le contrôle de l'administrateur) est la phase

d'enregistrement tandis que chaque session fait l'objet d'une phase préalable de vérification. Un nom d'utilisateur (*login*) constitue l'identifiant et le mot de passe associé l'élément de vérification. Une telle authentification n'est fiable que si le mot de passe est construit correctement et reste secret.

I.2.2.4 Lutte contre la contrefaçon de produits

Toute saisie de produits par une autorité de contrôle, telle que les services des douanes, se traduit par un prélèvement d'échantillons qui sont analysés et contrôlés dans le but de constituer des éléments de preuve de la contrefaçon. L'identité indiquée par l'étiquette que porte le produit est ainsi soumise à vérification par l'agent des douanes. L'identifiant peut être un ensemble de données telles que le fabriquant, l'origine, le numéro de série du produit. Un produit devient suspect dès que l'identifiant n'a pu être vérifié. Pour lever l'ambigüité, on peut alors recourir à des moyens dédiés disponibles chez le fabriquant supposé du produit ou des moyens plus lourds disponibles en laboratoire.

Dans le cas des objets à grande valeur commerciale et/ou artistique, un usage fréquent est de faire appel à une expertise humaine.

I.2.2.5 Authentification par PUF

On peut authentifier de façon sûre une machine électronique, à partir d'une fonction PUF (Physical Unclonable Function en anglais) associée à un ou plusieurs composants électroniques qu'elle intègre [PRTG02, LLG+05, SD07]. Nous reviendrons plus en détail sur la notion de PUF à la section III.4 du chapitre III. Dans un tel processus d'authentification, l'élément de vérification est le résultat de la fonction PUF, ce résultat est par définition identique si les caractéristiques et fonctionnalités de la machine restent inchangées par rapport aux caractéristiques et fonctionnalités d'origine. L'identité de la machine dépend non seulement de ses caractéristiques (numéro unique associé à la machine) mais aussi de ses fonctionnalités (la version de la tâche qu'elle doit exécuter par exemple). La nature de l'entité vérifieur peut être différente selon le protocole dans lequel intervient l'authentification par PUF. Mais en tous les cas, cette entité vérifieur doit avoir l'intelligence et les moyens techniques nécessaires pour vérifier la fonction PUF à travers ses résultats : l'ensemble des couples Challenges-Réponses qui la caractérise.

	A : Entité à Authentifier				B : Entité Vérifieur	
	Type de l'entité	**Rôle de l'entité**	**Les identifiants**	**Élément de vérification**	**Rôle de l'entité**	**Type de l'entité**
Appel à un service clientèle	Personne	Client	– Numéro Compte, – Nom, – Prénom, – Adresse, – Numéro de Dossier	– Mot de Passe, – Code PIN, – Date d'anniversaire	Service Client	– Personne – Système automatique
Détection de faux billets par un commerçant	Objet	Représentation d'une valeur fiduciaire	Valeur du Billet	Eléments de sécurité du billet	Commerçant	– Personne – Système automatique
Accès à un ordinateur personnel	Personne	Utilisateur	Non d'Utilisateur	Mot de Passe	Ordinateur	Système automatique
Produits contrefaits	Objet	Produit ayant une valeur marchande	– Références du Fabriquant, – Origine, – Numéro de Série, – Référence de l'Autorisation de Mise sur le Marche	– Nature du Produit, – Fait intervenir une Autorité Experte	– Autorité de contrôle	– Personne – Groupes de Personnes
Authentification par PUF	Électronique	Tâche à exécuter	Numéro du Circuit, Version de la Fonction	PUF	Vérifieur	Système automatique et Intelligent

Tableau I.1 Exemple de processus d'authentification

I.2.3 Authentification à partir d'une information partagée

Nous retrouvons là la notion d'authentification cryptographique [MVOV96, SV96], où l'authentification d'une entité A par une entité B se définit comme une vérification basée principalement sur le fait que l'entité A connait une ou plusieurs informations secrètes, comme une clé ou un mot de passe, et que B a les moyens de vérifier ces informations. Une telle authentification est mise en oeuvre chaque fois que l'utilisateur accède à son ordinateur ou se connecte à un site en ligne, à travers un mot de passe ou encore lorsque deux machines établissent une communication sécurisée.

En cryptographie moderne, divers protocoles permettent d'assurer une authentification entre deux **entités intelligentes** ou **machines**, c'est à dire des entités qui possèdent des moyens performants pour effectuer des opérations cryptographiques ou/et une mémoire pour conserver secrète une information comme une clé. Le but de ce document n'est pas d'énumérer de tels protocoles, ni de les étudier. Pour plus de détails sur l'authentification en cryptographie, nous renvoyons par exemple aux ouvrages [MVOV96, SV96].

La propriété remarquable ici est qu'en cryptographie, l'authentification auprès d'une machine (un ordinateur personnel, une application informatique en ligne ou tout autre type de terminal électronique) est basée sur la vérification d'un mot de passe ou d'une clé secrète que doit connaitre la personne ou la machine qui s'authentifie. Nous qualifierons ce type d'authentification : authentification entre **entités intelligentes** basée sur la connaissance d'un secret partagé. Nous en utiliserons le principe dans le prochain chapitre (*cf.* section II.6 du chapitre II) pour assurer une authentification mutuelle entre une machine électronique et un serveur distant.

Dans la seconde partie de cette thèse, nous reviendrons sur cette forme d'authentification entre entités intelligentes, où une entité être-humain (l'entité B) veut authentifier une machine électronique (l'entité A) avec comme contrainte que ces deux entités évoluent dans un environnement propre, fermé à l'autre (par exemple en hors-ligne).

I.2.4 Authentification à partir d'une caractéristique

Aujourd'hui la connexion à son ordinateur peut [partiellement] s'effectuer par prise d'empreinte digitale, reconnaissance vocale ou faciale. Dans ces conditions, l'authentification auprès de la machine est basée sur une caractéristique ou un comportement biométrique propre à l'individu [JRP04].

Pour authentifier une machine électronique, on peut également se baser sur une caractéristique propre associée à un ou plusieurs composants électroniques qu'elle intègre [LLG+05, SD07]. Cette donnée caractéristique provient d'une fonction physiquement non reproductible ou **PUF** (*Physical Unclonable Fonction*) qui sous stimulation

de l'électronique, fournit toujours une même réponse qui servira d'empreinte.

Concernant les objets de la vie courante, comme les produits manufacturés, qui ne sont ni des personnes ni des machines électroniques, leur authentification unitaire est basée sur des caractéristiques physiques uniques. Gérer du secret autour de ces caractéristiques comme dans [DFM98] induit des contraintes sur la phase de vérification. Ici, nous considérons les caractéristiques physiques comme publiques ainsi que les méthodes utilisées pour en extraire des empreintes. Ces caractéristiques sont soit intrinsèques à l'objet soit issues d'un marquage (ou impression) de l'objet. Comparable dans son principe à celui de la biométrie, l'*authentification intrinsèque* d'objets inertes présentent cependant vis à vis de l'authentification biométrique des propriétés particulières. L'authentification intrinsèque peut porter sur un jeton (ou "tag" en anglais) fabriqué selon un procédé spécifique, à sceller à l'objet pour lui associer ses caractéristiques.
De façon différente, l'*authentification par marquage* repose sur les variabilités induites par le procédé de marquage de l'objet à authentifier. Comme pour les jetons, on fait en sorte que le marquage soit difficile à séparer de l'objet marqué. On peut citer le marquage par ajout d'*ADN de synthèse* ou encore l'impression de motifs en limite de résolution comme décrit à la section I.4 du chapitre II.
Implicitement les caractéristiques utilisées doivent être difficile à reproduire. Un protocole d'authentification sûr pourra être établi sur une empreinte unique de l'objet à tracer.

Nous appelons ainsi **authentification d'objets**, une authentification basée sur une caractéristique physique de l'objet à authentifier (l'entité A) par opposition à une authentification intelligente. Donc, nous qualifions l'entité à authentifier A d'objet, dans les cas où son authentification par un vérifieur B dépend d'une caractérisation de A. En d'autres termes, si une personne s'authentifie en utilisant une clé ou un mot de passe alors elle suit une procédure d'authentification intelligente. Par contre, si la personne utilise une empreinte biométrique, alors, nous considérons son authentification comme une authentification d'objet. Nous regroupons ainsi sous le terme d'authentification d'objet, les authentifications biométrique, par PUF ou via une caractéristique physique. C'est en définitive la nature de l'élément de vérification qui détermine la classe du processus d'authentification (intelligente ou d'objet).

Cependant, en pratique dès que cela est possible, l'authentification d'une entité A est basée sur la combinaison de données de sources différentes, comme la combinaison d'une empreinte digitale et d'un mot de passe (authentification forte[1]). Il est aussi possible d'extraire une connaissance partagée telle une clé cryptographique à partir d'une empreinte biométrique [ST96, DFM98, MRLW01, ST96], d'une PUF - [SD07] (concernant les PUF nous reviendrons sur ce point dans le chapitre IV sur l'authentification de systèmes électroniques) ou d'un objet [**?**].

1. Est appelée authentification forte (*Two-factor authentication* en anglais), tout type d'authentification qui est basé sur deux facteurs de sources et de natures différentes (`http://fr.wikipedia.org/wiki/Authentification_forte`)

I.2.5 Authentification à travers un tiers

L'authentification comme nous l'avons définie à la section I.2.1 ci-dessus suppose qu'une entité vérifieur B ait les moyens et l'intelligence nécessaires pour vérifier l'authenticité d'une entité A. La question qui se pose maintenant est : comment prendre en compte le fait que le vérifieur B n'a pas accès aux éléments de vérification de l'état de référence de l'entité A ou le fait qu'il ne dispose pas de moyens techniques pour effectuer la vérification ?

Une solution à cette problématique est que le vérifieur B puisse passer par un tiers de confiance C susceptible de jouer le rôle de vérifieur : c'est une forme usuelle d'authentification. Si vis à vis de la définition donnée à la section I.2.1 le principe reste le même, concrètement le vérifieur B va devoir soit faire confiance au tiers C (à supposer que faire confiance est une forme d'authentification sûre mais nous reviendrons sur ce point) soit l'authentifier pour pouvoir lui demander de vérifier l'entité B. C'est cette transitivité qui permet une extension des processus d'authentification entre entités de natures différentes.

En cryptographie ce tiers de confiance est appelé autorité d'authentification. Dans les situations où les entités interagissent à distance sur un réseau, leurs authentifications nécessitent une autorité reconnue pour contrôler les interactions entre de multiples entités de différents types. C'est le cas du protocole Kerberos ou des protocoles basés sur une architecture PKI à clefs publiques [MVOV96, SV96]. Dans de tels protocoles, il n'intervient que des entités intelligentes possédant toutes les moyens techniques pour effectuer les opérations cryptographies nécessaires et stocker de façon sûre des clés secrètes ou privées.

Un tiers de confiance s'impose aussi dès qu'une entité n'a pas suffisamment d'"intelligence" i.e. de moyens pour réaliser complètement ou dans des conditions convenables l'authentification d'une entité. C'est le cas par exemple lorsqu'un commerçant (entité B) veut authentifier un billet de banque (entité A). Généralement, il se contente d'une vérification sommaire des caractéristiques visuelles et tactiles du billet pour en déduire si le billet est un faux ou non. Pour les grandes valeurs faciales, le commerçant pourra de plus faire appel à un détecteur de faux fourni par la banque centrale, en l'occurrence le tiers de confiance (entité C). L'authentification du détecteur (authentification de C par B) se réduit ici à la confiance que porte le commerçant en son détecteur. La confiance du commerçant vient du fait qu'il a acquis son détecteur auprès d'un fournisseur agréé. Dans cette configuration, le commerçant n'a pas vraiment d'autres possibilités que sa propre analyse et/ou le fait de faire confiance à l'instrument de vérification.

Cela nous montre que dès lors qu'un vérifieur ne peut être autonome dans sa vérification, il lui faut recourir à (au moins) un tiers pour vérifier l'authenticité d'une entité. En traçabilité sécurisée, c'est un être humain qui décide au final (un commerçant,

un agent des douanes, un expert assermenté auprès des tribunaux, ...) le caractère authentique ou non d'un produit. Si le contrôle est non autonome, le tiers de confiance peut mettre à disposition différentes ressources. Dans le chapitre suivant nous proposons de distinguer trois formes d'authentification d'objet en fonction de la nature de ces ressources.

I.3 Différents contextes de vérification

Dans la suite nous nous focaliserons sur l'authentification spécifique aux objets c'est à dire sur une vérification basée sur des caractéristiques associées à l'état de l'objet à authentifier, caractéristiques que nous définirons comme étant son *empreinte*. La vérification de l'empreinte va mobiliser des ressources qui peuvent être de différents gabarits suivant les caractéristiques utilisées. Du point de vue des protocoles, nous sommes amenés à différencier les formes d'authentification d'objets *en fonction des conditions d'accès à l'empreinte de référence lors de la vérification*. Cette dernière définit l'élément de vérification de l'état de référence de l'entité objet. Son stockage à un instant t_0 de la vie de l'objet - typiquement à la fabrication d'un produit manufacturé - va permettre son authentification à un instant t ultérieur en la comparant avec l'empreinte prise à l'instant t.

Nous classifions ainsi l'authentification d'objets en trois catégories : l'*authentification en laboratoire* pour laquelle la vérification va s'effectuer en milieu sûr chez ou par un tiers de confiance, l'*authentification à distance* pour laquelle une requête est adressée à un tiers et l'*authentification autonome* pour laquelle la vérification est faite en place sans accès à un réseau de communication.

I.3.1 Authentification d'objets en laboratoire

Dans une authentification en laboratoire, schématisé à la Figure I.1), soit l'objet est transporté chez un tiers de confiance qui en assure la vérification dans ses locaux, soit un agent est dépêché par le tiers de confiance sur le lieu de contrôle de (ou des) l'objet(s). Le premier cas correspond usuellement à la nécessité de recourir à des moyens lourds de vérification (appareillages spéciaux et/ou procédures spéciales). Le second correspond à la mise en oeuvre de dispositifs non standards ou au besoin d'un savoir-faire spécifique / une expertise.

Nous appelons un tel tiers Laboratoire d'Authentification (LA). Le LA possède une base de connaissances qui dans le cas d'objets courants est une base de données de toutes les empreintes de référence des objets enregistrés. Dans ce cas, le LA doit par ailleurs être en mesure d'assurer une prise d'empreinte dans les mêmes conditions que celles utilisées lors de la prise de référence. Ce type d'authentification revient pour le vérifieur B à envoyer l'objet chez le LA qui effectue une prise d'empreinte avant de la comparer aux références enregistrées dans sa base de données. Pour le vérifieur B, le LA et le canal de transport sont de confiance : il suppose donc que toutes les opérations

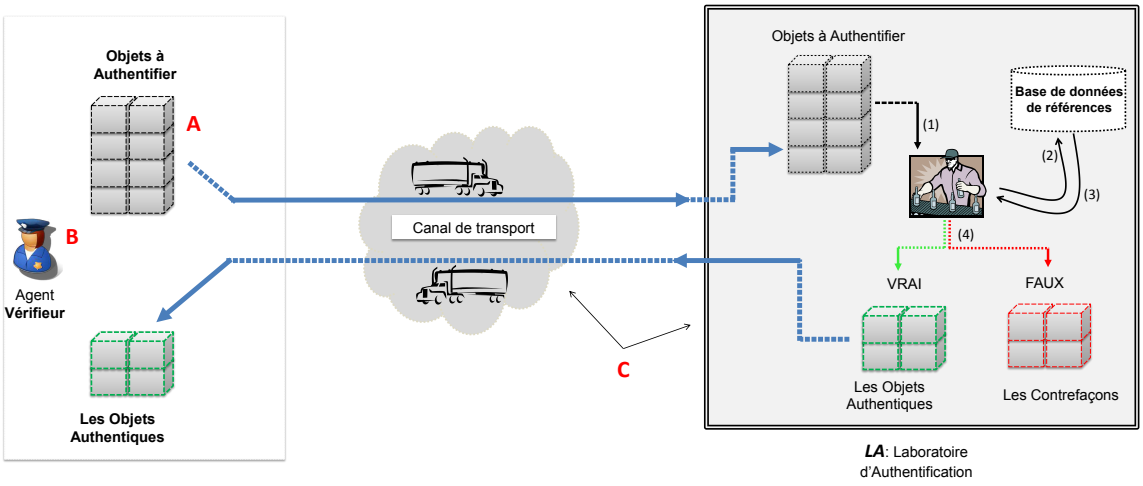

Figure I.1 *Authentification d'objets (entités* ***A****) par un agent vérifieur (entité* ***B****) en passant par un tiers de confiance (entité* ***C****) que constituent le LA et le canal de transport.*

– o –

effectuées par le LA sont correctes et que la réponse binaire, objet A authentique ou non, est celle fournie par le LA.

Malgré la lourdeur imposée par la contrainte d'un déplacement, ce processus d'authentification est couramment utilisés. On le retrouve dans le fait que toutes saisies de produits manufacturés par les autorités de contrôle (la police ou la douane), se traduit par un prélèvement d'échantillons qui sont analysés et contrôlés par le fabriquant ou son représentant officiel, afin qu'il produise le cas échéant un élément de preuve de la contrefaçon. Ce processus est également le plus adapté aux objets qui ont une grande valeur commerciale ou artistique : seul le fabriquant, le concepteur de l'objet ou des experts reconnus peuvent certifier l'authenticité de tels objets, car leur authentification est basée sur des facteurs qui ne sont pas standards ou relèvent d'une expertise.

L'avantage réel d'un tel procédé pour l'entité B est la fiabilité du résultat de l'authentification. En effet sous l'hypothèse d'un canal de transport sûr, le LA n'a pas d'intérêts à déclarer authentique un objet qui ne l'est pas et le LA a par définition toute l'expertise nécessaire pour donner un résultat fiable en s'aidant des empreintes de référence qu'il détient. Cependant, il n'est pas toujours possible ou rentable d'organiser un déplacement (de l'objet ou de l'expert). Dans pareil cas, un vérifieur B va devoir sur tout lieu de vérification, accéder aux empreintes ou à l'empreinte de référence.

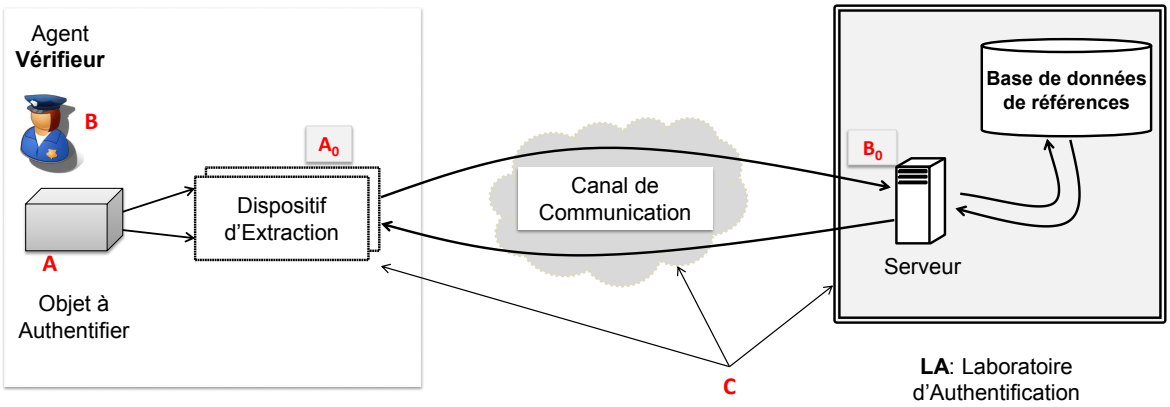

Figure I.2 *L'authentification d'un objet (entité **A**) par vérifieur (entité **B**) à travers un dispositif de lecture, un canal de communication et un serveur d'authentification du LA (entité **C**). Cette authentification, nécessite au préalable une authentification du dispositif de lecture (entité A_0) par le serveur du LA (entité B_0).*

– ○ –

I.3.2 Authentification d'objets à distance

L'authentification d'objets à distance a recours à un serveur distant hébergé par un tiers de confiance qui détient la base de données des empreintes de référence. Par analogie avec la section précédente (section I.3.1), ce serveur est une application experte (pour l'authentification) associée à la base de données de référence du laboratoire d'authentification LA, mais cette fois-ci un serveur est à disposition pour une consultation à distance. Comme le LA, ce serveur évolue dans un environnement sécurisé situé par exemple dans les locaux sûr du LA comme sur la Figure I.2.

Si la technologie utilisée pour la caractérisation des objets le permet, une authentification à distance d'un objet A consiste pour le vérifieur à extraire ses identifiants et empreintes pour les transmettre au serveur sous forme de requête, à travers un **canal de communication**. Le serveur authentifie l'émetteur de la requête (i.e. le vérifieur), effectue la comparaison avec la référence correspondant à l'identifiant puis renvoie sa réponse au vérifieur B. L'authentification de l'objet A est donc faite par le serveur C sur la base des données relatives à l'objet A obtenues par l'entité B au moyen d'un **dispositif de lecture**.

Ce processus d'authentification implique donc au moins deux nouveaux participants : le dispositif de lecture qui fournit au vérifieur les données sur l'objet à authentifier et le canal de communication entre le dispositif de lecture et le serveur d'authentification. Le vérifieur doit être assuré de l'intégrité du dispositif de lecture

et de la sécurisation des communications avec le serveur distant. Lors d'une authentification en LA, le problème de l'intégrité du dispositif de lecture ne se pose pas, car ce dispositif évolue dans les locaux du LA, et hérite de ce fait de la confiance que l'entité vérifieur B porte au LA. Cependant dans une authentification à distance, le dispositif de lecture est tout comme l'est l'objet à authentifier lui-même, potentiellement sujet à une falsification malveillante. Dans un tel mode de vérification, il appartient au serveur B_0 du laboratoire LA de vérifier en premier lieu (i.e. avant de vérifier l'objet A) l'authenticité du dispositif de lecture A_0 via le canal de communication.

Des exemples de protocoles d'authentification à distance nous viennent de la biométrie. En effet les travaux comme ceux de Boyen *et al.* [BDK+05] et de Bringer *et al.* [BCI+07, BC08] proposent des protocoles d'authentification à distance des empreintes biométriques. Dans ces protocoles, c'est un humain qui joue le rôle de l'objet à authentifier auprès d'un serveur distant. En reprenant les notations du schéma (Figure I.2) de la section précédente, l'entité B disparaît et c'est A qui s'authentifier uniquement auprès de C. Cela constitue une différence entre l'authentification biométrique et la traçabilité sécurisée (authentification sûre d'objets manufacturés) qui suppose que l'entité vérifieur de A est un objet passif. Cependant, dans les deux situations l'empreinte de référence de la personne ou de l'objet à authentifier est obtenue à distance i.e. extraite en local puis vérifiée à distance. A noter que dans le schéma discuté, présenté ci-dessous, l'humain vérifieur B est un simple spectateur : il porte sa confiance aux autres acteurs du schéma et à leurs protocoles de communication.

Les protocoles biométriques assurent une authentification sûre d'une personne à partir de ses caractéristiques biométriques, à travers un canal de communication (à priori) non sûr. Cette solution permet ainsi à une personne d'utiliser sa biométrie pour s'authentifier auprès d'un serveur distant en complément ou à la place de l'usuel mot de passe. Cela présente l'avantage d'une vérification implicite d'une présence humaine contrairement à l'usage d'un mot de passe.

Les travaux de Lancrenon *et al.* [LGF09, LGF11] concernent précisément l'authentification d'objets à distance. Les auteurs s'inspirent du protocole biométrique de Bringer *et al.* (*cf.* Figure I.2) où la confidentialité sur l'entité testée est assurée côté serveur, en faisant une analogie entre les caractéristiques biométriques et les caractéristiques *morphométriques* que l'on peut extraire d'un produit. Le système de chiffrement homomorphique utilisé y est remplacé par un système basé sur les courbes elliptiques de façon à diminuer la taille des clés (tout en conservant le même niveau de sécurité) et la quantité de calculs côté lecteur. Les protocoles proposés incluent alors un mécanisme de contrôle du non rejeu et de l'intégrité des messages reçus par le serveur.

Cependant ces protocoles ne traitent pas de l'intégrité et de la certification du dispositif de lecture (appelé *lecteur* dans [LGF09]). L'interconnexion entre le dispositif de lecture et le serveur d'authentification peut en fait être mise à profit pour certifier l'intégrité du premier, du second. En résumé une authentification mutuelle

est nécessaire entre le dispositif de lecture et le serveur d'authentification, avant toute authentification à distance. Nous reviendrons sur ce point dans le chapitre III sur l'authentification de système embarqué, et aussi à la section II.6 (chapitre II) où nous évoquerons une certification à distance d'un dispositif d'authentification autonome.

Vérifier un objet sans avoir à perturber son acheminement coïncide avec une démarche de traçabilité des produits manufacturés. Vérifier à distance exige de s'assurer de l'intégrité du dispositif de lecture et d'établir une connexion sûre avec la base de données contenant les empreintes de référence. Le dispositif de lecture peut d'une autre façon tout comme l'empreinte de référence, être vérifiée localement sans connexion réseau.

I.3.3 Authentification autonome d'objets

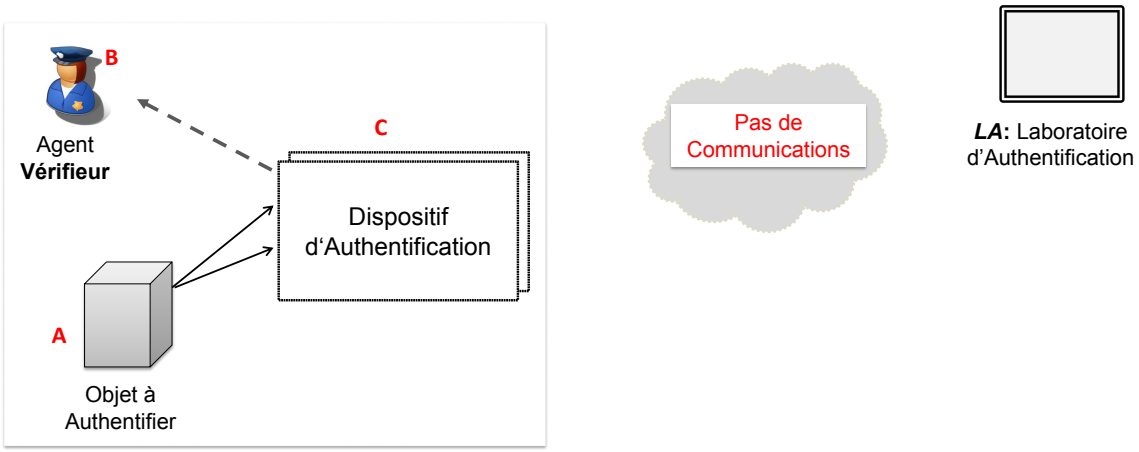

Figure I.3 *L'authentification d'un objet (entité **A**) par un agent vérifieur (entité **B**) avec ou sans un dispositif authentification (entité **C**), et sans contacter à distance le LA.*

– ○ –

Nous qualifions de *locale* une authentification d'objet où l'empreinte de référence est stockée avec l'objet lors de la phase d'enregistrement et où l'empreinte extraite est vérifiée par comparaison avec la référence stockée.

En particulier, de tels procédés d'authentification n'utilisent pas de bases de données ni de déplacement vers un laboratoire d'authentification. Ils se conforment à certains procédés d'authentification biométrique[2] où l'empreinte de référence est

2. Encore une fois une personne n'est pas un objet ☺, mais faire cette analogie nous permet de différentier les authentifications basées sur un mot de passe de celles basées sur des caractéristiques biométriques.

stockée et protégée contre les modifications sur une carte à puce, sur une bande magnétique [DFM98] ou sur tout support de stockage fiable et amovible. Le principe est de stocker sur ce support à la fois des informations sur l'identité de la personne et l'empreinte de référence avec leur signature électronique. Ce principe d'authentification implique la vérification de l'intégrité de l'empreinte de référence stockée sur le support.

Cet usage de la signature électronique se généralise avec la création d'un certificat électronique (comme un certificat X.509 *cf.* [CSF+08, MVOV96]) au sein d'une infrastructure à clé publique (*Public Key Infrastructure* ou PKI). La phase de vérification commence alors par valider la chaîne de vérification du certificat joint. Un certificat valide assure donc l'intégrité de l'empreinte de référence de l'entité à authentifier. Ainsi, l'authentification proprement dite peut être réalisée par extraction d'empreinte et comparaison avec l'empreinte de référence. Ce principe est adopté par les derniers standards de passeports électroniques, encore appelés passeports biométriques ou *e-passports*, définis par l'ICAO [3][ICA06, HHJ+06]. Le passeport biométrique sert ainsi de certificat électronique pour le voyageur.

La vérification de certificats électroniques nécessite une accessibilité sûre et fiable au certificat contenant la clé publique de l'Autorité de Certification (CA). Pour simplifier, la CA représente ici l'autorité qui signe les certificats des objets. Le certificat du CA doit donc être accessible lors de la phase de vérification : il peut être stocké de façon sécurisée afin de prévenir toute modification ou obtenu à travers une connexion distante mais dans ce dernier cas l'authentification n'est plus autonome.

Comme pour l'authentification à distance d'objets, le vérifieur devrait avant tout et en toute rigueur, s'assurer de l'authenticité et de l'intégrité du dispositif de lecture qu'il utilise. Nous qualifierons d'*autonome* ou de *hors-ligne* (*off-line* en anglais), une authentification locale d'objet dont la phase de vérification se déroule sans communication externe et inclut une vérification préalable du dispositif de lecture. Tout protocole destiné à régir un tel procédé d'authentification devra donc s'affranchir de la possibilité d'établir une communication entre le dispositif de lecture et un tiers de confiance. Avant de développer la notion d'authentification autonome au chapitre suivant, illustrons le cas d'une authentification locale à travers les "Print signatures" introduites dans [ZWK03].

I.4 Authentification locale par impression laser

Les "*Print signatures*" résultent d'un marquage par impression laser, introduit par Zhu *et al.* dans [ZWK03] pour l'authentification de documents imprimés.

L'impression de points de taille microscopique (des disques de diamètre de quelques

3. **ICAO** :International Civil Aviation Organization

centaines de microns) au moyen d'une imprimante laser constitue un marquage unique de documents du fait des aléas du processus d'impression des imprimantes lasers. Après impression, chaque point observé au microscope présente une forme caractéristique de telle sorte que toute impression de disque est unique et permet de constituer une empreinte physique (Figure I.4). Une empreinte numérique, une "Print signature" [ZWK03], pourra en être extraite via le profil du contour du point imprimé obtenu après binarisation (Figure I.4) et permettre un "e-ticketing" c'est à dire une vérification locale du document imprimé par le biais d'une signature électronique.

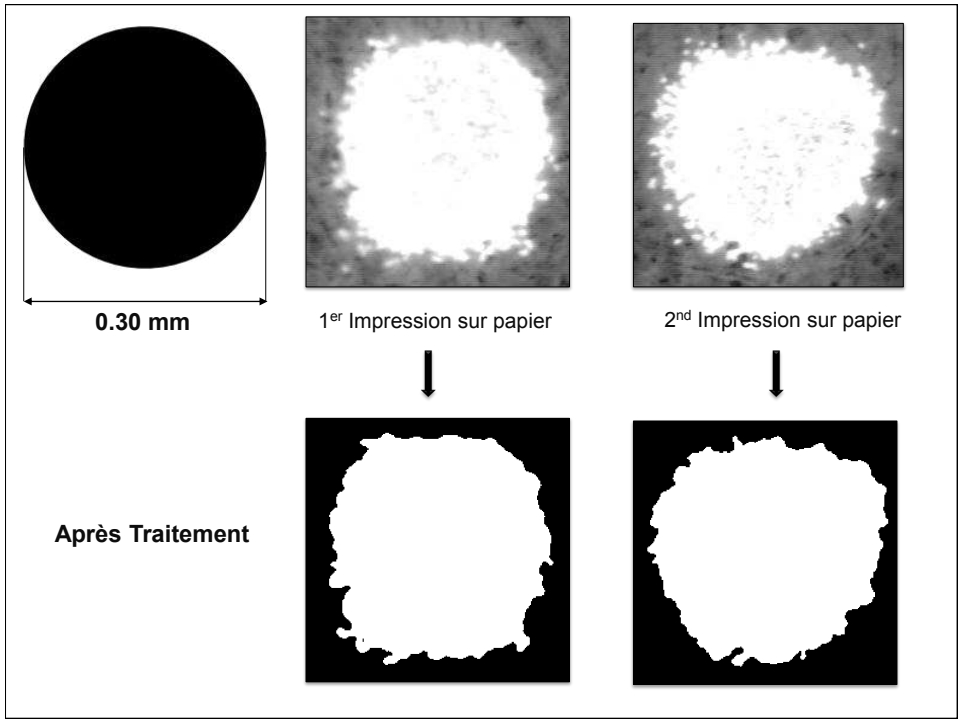

Figure I.4 *Forme du point utilisé comme marquage de documents imprimés*

– ∘ –

Dans cette section, nous présentons brièvement la méthode d'authentification locale de Zhu *et al.* [ZWK03] puis deux méthodes que nous avons développées, visant à gagner en robustesse en lissant le contour.

I.4.1 Caractérisation en terme de rayons selon [ZWK03]

Pour décrire la forme binarisée issue de l'impression d'un disque, Zhu *et al.* considèrent un ensemble de rayons séparant le barycentre du contour échantillonné

angulairement (*cf.* Figure I.5). Un compromis est ainsi à trouver entre finesse de description du contour, donnée par le nombre n de rayons (Figure I.6) et robustesse de cette description lors de l'acquisition de différentes images du même point. En effet la diffusion de la lumière dans l'encre et le papier conduisent à l'engraissement des points d'encre, phénomène connu sous le nom de "optical dot gain" qui bruite l'acquisition des images au microscope et leur binarisation. L'influence de ce bruit est renforcée par la présence de points d'encre périphériques qui suivant le trajet de la lumière vont être ou non connexes au point après binarisation. Dans [SA10] et [PSA11], l'impression laser est exploitée avec comme principe, la nécessité d'effectuer une impression de haute qualité pour limiter les points parasites. L'utilisation de formes autres qu'un disque, comme par exemple les caractères alphabétiques **e**,**a** ou le **s** est une autre piste.

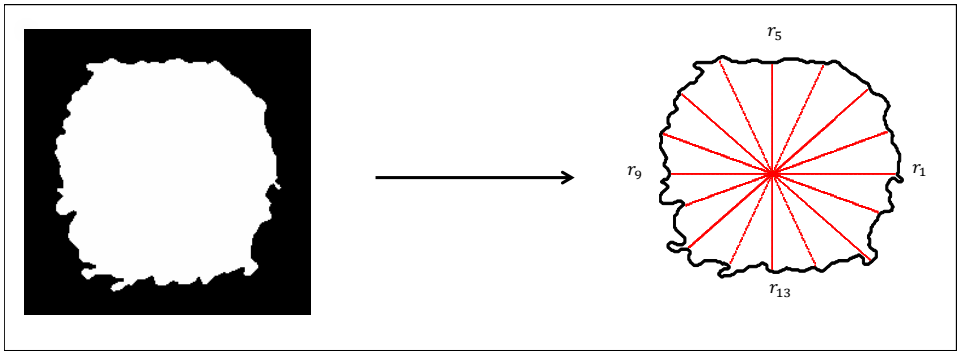

Figure I.5 *Extraction de la forme des points selon [ZWK03] ($n = 16$ rayons)*

– ○ –

Le profil de la forme est là représenté par les nombres réels $(r_1, r_2, \cdots, r_n)$, valeurs des rayons successifs. Pour s'affranchir de toute homothétie, l'empreinte numérique est définie par le vecteur $\mathcal{R}_n$ (équation I.1) des rayons normalisés par le rayon moyen. Un tel descripteur est également invariant en translation mais pas en rotation. C'est pourquoi un élément de repère a été également imprimé au voisinage du point.

$$\mathcal{R}_n = (\frac{r_1}{\bar{r}}, \cdots, \frac{r_n}{\bar{r}}) \qquad avec \quad \bar{r} = \frac{1}{n}\sum_{i=1}^{n} r_i. \tag{I.1}$$

La similarité de deux profils se mesure en évaluant la distance euclidienne entre les deux vecteurs servant d'empreintes numériques.

Les mesures portées à la Figure I.6 ont été réalisées à partir de séries d'images de points de 300 μm de diamètre, que nous avons réalise en microscopie standard à une résolution de 1.7 μm par pixel. Les distances entre empreintes extraites d'images

d'un même point (distribution des authentiques) ont été mesurées sur 50 images (1225 mesures). Les distances entre empreintes extraites d'images de points différents (distribution des imposteurs) ont été mesurées sur 30 images (435 mesures). On constate que ces dernières distances sont très étalées. On note sur nos mesures la présence de points aberrants principalement causée par le "dot gain" optique [IFA10]. Il résulte que les distributions des authentiques et imposteurs sont mal séparées.

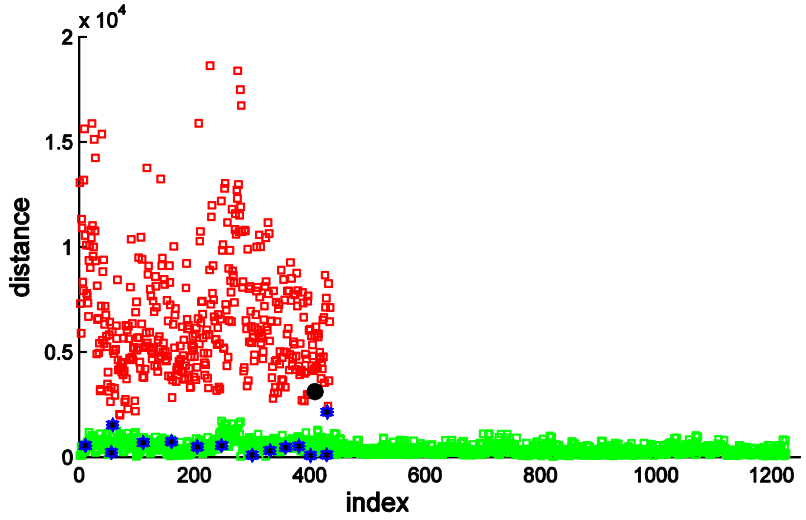

Figure I.6 *Distances mesurées entre empreintes selon [ZWK03] à $n = 72$ rayons, extraites d'images d'un même point (authentiques en vert) - les points en bleu correspondent à des mesures aberrantes [IFA10] - resp. de points différents (imposteurs en rouge).*

– ∘ –

I.4.2 Caractérisation du contour par morceaux

En vue de limiter l'influence du "dot gain" optique et améliorer la séparation des deux classes, authentiques et imposteurs, nous avons partitionné et lissé le contour par morceaux. L'ensemble C des pixels du contour, de coordonnées (x, y), est subdivisé en $p \geq 2$ portions notées C^j, chacune d'elles étant caractérisée séparément par l'ensemble D^j des m premiers coefficients $d^j_{i_x}$ (resp. $d^j_{i_y}$) de la transformée en cosinus (DCT) des coordonnées x, resp. y. A chaque fois, la forme est recalée en rotation et centrée sur son *barycentre*.

L'empreinte numérique $\mathcal{R}^p_m$ est alors constituée du vecteur résultant de la concaténation des p m-uplets paires de coefficients :

$$\mathcal{R}^p_n = (d^1_{1_x}, d^1_{1_y}, \cdots, d^1_{m_x}, d^1_{m_y}, \cdots, d^p_{1_x}, d^p_{1_y}, \cdots, d^p_{m_x}, d^p_{m_y}) \quad \text{(I.2)}$$

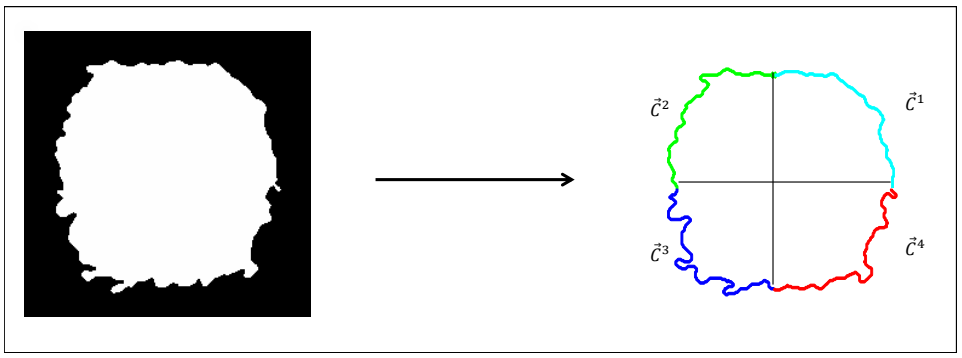

Figure I.7 *Extraction de la forme des points en partitionnant le contour en $p = 4$ portions C^1, C^2, C^3 et C^4, chacune décrite par les $m = 8$ premiers coefficients de sa DCT en x, resp. en y.*

– ○ –

Comme précédemment, la similarité de deux profils est mesurée en évaluant la distance euclidienne entre deux vecteurs servant d'empreintes numériques.

Les mesures réalisées sur les images de la section I.4.1, avec $p = 4$ morceaux et $m = 8$ coefficients, sont portées à la figure I.8. On constate que les distances entre empreintes extraites d'images de points différents sont en relatif moins étalées. Cependant des points aberrants subsistent indiquant que le dot gain optique affecte de façon notable la séparation entre authentiques et imposteurs.

I.4.3 Caractérisation en termes de rayon après convexification du contour

Pour gommer davantage l'influence du "dot gain" optique tout en conservant le caractère discriminant de la forme, nous avons finalement procédé à un lissage global plus important de la forme en la rendant convexe. La méthode que nous avons employée avec Daniela Coltuc, Professeur au Politehnica de Bucarest, et Mihai Petrovici, pour faire ce lissage est directement issue de la méthode de représentation dans l'espace des échelles de courbure (CSS), utilisée dans le standard MPEG 7 (où la signature est l'échelle de passage à une courbure nulle en fonction de l'abscisse curviligne). Le contour est itérativement filtré avec un noyau gaussien monodimensionnel d'écart-type croissant jusqu'à obtenir une forme convexe.

L'empreinte numérique résulte de la mesure de n rayons issus du barycentre comme dans le cas de [ZWK03]. Là encore la méthode est invariante par translation et une correction en rotation est effectuée au moyen de l'élément de repère. Un seuillage de la valeur du rayon courant par rapport au rayon moyen rend la normalisation des rayons inutile et conduit à une empreinte numérique binaire.

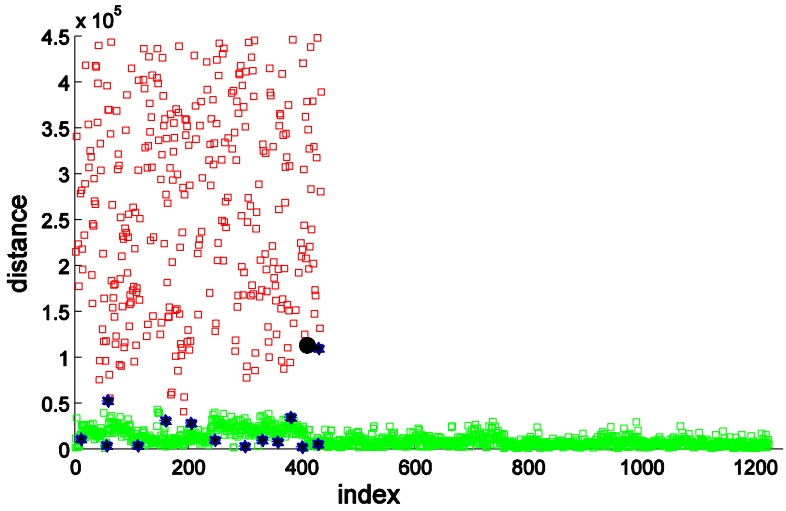

Figure I.8 *Distances mesurées entre empreintes formées des $m = 8$ premiers coefficients de la DCT de $p = 4$ morceaux de contour, extraites d'images d'un même point (authentiques en vert) -les points en bleu correspondent à des mesures aberrantes - resp. de points différents (imposteurs en rouge).*

– ○ –

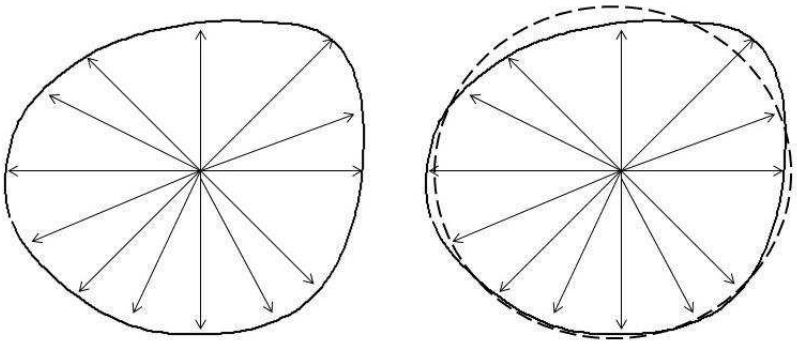

Figure I.9 *Extraction de la forme des points après convexification ($n = 12$ rayons)*

– ○ –

La similarité de deux profils est alors mesurée en évaluant la distance de Hamming normalisée entre deux empreintes numériques.

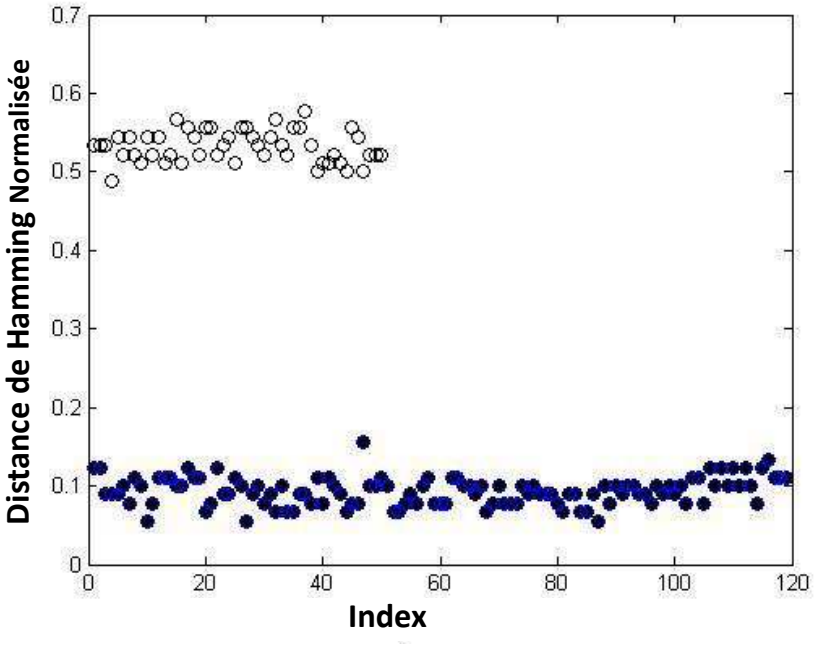

Figure I.10 *Distances mesurées entre empreintes binaires à $n = 90$ rayons, extraites d'images d'un même point (authentiques en sombre), resp. de points différents (imposteurs en clair), après convexification des formes.*

– ○ –

Outre les deux séries exploitées jusque là, une troisième série de 121 images d'un second point telles que décrites à la section I.4.1 est utilisée, pour mesurer des signatures de 90 bits. Sur la Figure I.10, on peut noter deux classes ici bien séparées, sans points aberrants, mesurées sur les deux séries de 50 et 121 points (correspondant chacune à un point distinct), en prenant la première image de la deuxième série comme référence. La distribution des authentiques (mesurée sur ces deux séries) ainsi que la distribution des imposteurs (mesurée en considérant les paires d'images de la série de 30 images de points distincts) sont représentées à la Figure I.11. Sur les mêmes images que celles utilises comme données aux paragraphes I.4.1 et I.4.2, nous obtenons un taux d'erreurs à part égales entre fausses acceptations et faux rejets (EER) de 2 (cf. [IFA10]).

I.5 Conclusion

Dans ce chapitre nous avons introduit la notion d'authentification autonome d'objets en combinant authentification cryptographique et authentification biométrique. Une authentification d'objet fait intervenir outre l'entité à vérifier, une entité vérifieur chargée de réaliser une vérification *sûre* (au sens cryptographique) d'empreinte, l'empreinte étant considérée ici publique tout comme sa fonction d'extraction à

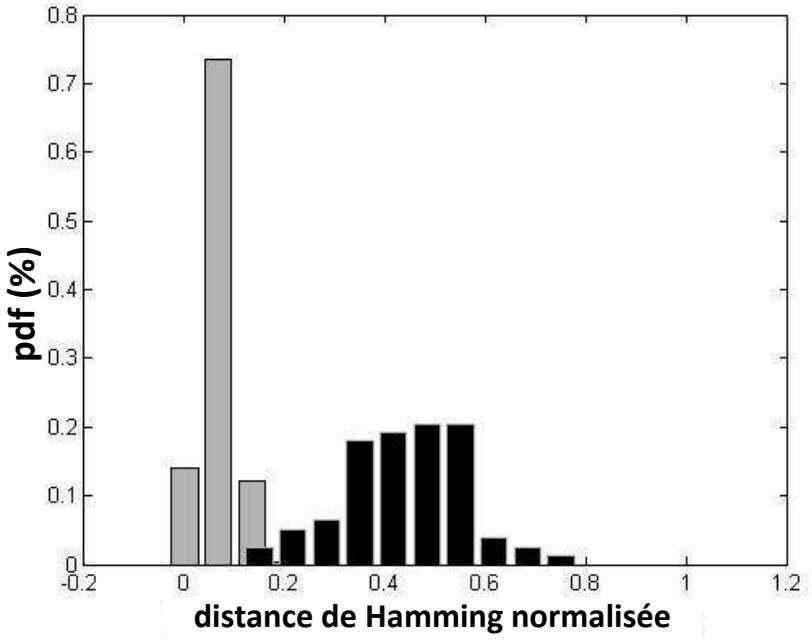

Figure I.11 *Distribution de la distance normalisée de Hamming relative aux authentiques (en clair) et aux imposteurs (en sombre), après convexification des formes binarisées de points et extraction de signatures radiales de 90 bits.*

– ○ –

partir de caractéristiques physiques associées à l'objet. En authentification autonome, l'objectif va être de limiter la participation d'un tiers de confiance tout en assurant une vérification hors-ligne du dispositif de lecture préalablement à la vérification d'objets.

La dernière section présente une forme de caractérisation de document par impression laser. Les développements effectués autour de cette méthode de marquage ont permis une illustration concrète et une amélioration au passage du processus d'extraction d'empreinte, ainsi qu'une imprégnation du domaine de la traçabilité sécurisée pour mieux définir les spécificités propres à une authentification d'objet physique.

Dans le chapitre suivant, nous partons d'une authentification locale pour voir comment la rendre autonome et dégager les notions associées.

Chapitre II

Authentification autonome d'objets

II.1 Introduction : certification de l'objet et vérification du dispositif d'authentification

A la section I.3.3 du chapitre I nous avons introduit la notion d'authentification autonome d'objet. On rappelle que pour ce faire, toute vérification d'objet nécessitera une authentification du dispositif matériel utilisé. Les interactions du dispositif et les exigences auxquelles il doit satisfaire pour assurer une telle vérification sont ici définies [IAFF10]. La description qui en est faite s'inspire du passeport biométrique [ICA06, HHJ+06] alors que nous associons un passeport appelé certificat, à chaque produit manufacturé. Notre apport réside dans la description d'un cadre d'authentification du dispositif matériel où la participation active de l'agent vérifieur est accrue.

Dans ce chapitre, la notion d'authentification autonome d'objet est tout d'abord formalisée à travers les définitions des participants en interaction et des diverses fonctions en jeu. Nous abordons ensuite les hypothèses sur lesquelles doit se baser une traçabilité sécurisée autonome. Il s'agit de rappeler les exigences de fiabilité d'une technologie de caractérisation d'objets, et de définir les moyens d'accès aux empreintes de référence des objets enregistrés en mode autonome c'est à dire sans faire appel à un tiers distant. Nous introduisons les ressources d'un adversaire dont le but est de modifier de façon frauduleuse ou d'empêcher un processus d'authentification tel que nous l'étudions. Enfin nous décrivons les phases d'enregistrement et de vérification du processus de traçabilité sécurisée autonome.

II.2 Définitions principales pour une authentification autonome

Définition 1 (Objet à Authentifier : $\mathcal{OA}$)
L'objet à authentifier $\mathcal{OA}$ est un produit manufacturé pour lequel on suppose disposer

d'une fonction robuste permettant d'en extraire une empreinte unique et infalsifiable qui le caractérise.

Pour les spécificités d'une authentification autonome, il est important que tout $\mathcal{OA}$ possède une zone de stockage d'informations. Le stockage peut prendre la forme usuelle d'un Code barres 1D, 2D (Datamatrix, QR code) ou d'une puce RFID. Les formes et paramètres de la zone de stockage dépendront en premier lieu de la quantité d'informations et de la durée de vie du produit. Toutes les informations qui seront conservées sur ou avec l'$\mathcal{OA}$ sont supposées publiques, c'est à dire lisibles sans restriction. En pratique, la plupart des produits manufacturés dispose d'une étiquette, composée généralement d'un Code 1D ou 2D mais souvent également d'une puce RFID, contenant des informations telles que le numéro de série, date de production, ... utiles à l'identification du produit.

Définition 2 (Empreintes physiques :$\mathfrak{E}_{phy}$)
L'empreinte physique d'un $\mathcal{OA}$ est la signature d'une caractéristique physique sur l'objet. Elle est obtenue à partir de la structure ou du marquage de l'objet. Elle s'apparente au signal brut de sortie d'un capteur examinant l'objet. De ce fait, l'empreinte physique est sujette au bruit d'acquisition. Dans [ZWK03, IFA10], l'empreinte physique d'un document imprimé est une image de points imprimés sur l'objet, extraite au microscope.

Pour une technologie donnée de caractérisation d'objet, nous représentons par $\mathfrak{E}_{phy}$ l'ensemble des empreintes physiques.

Définition 3 (Empreintes Numériques $\mathfrak{E}_{num}$ et la fonction d'extraction $\mathcal{E}(\cdot)$)
Nous définissons l'empreinte numérique comme étant une suite numérique extraite d'une empreinte physique de l'ensemble $\mathfrak{E}_{phy}$. Dans [ZWK03, IFA10], l'empreinte numérique est une suite de rayons de la forme issue de la binarisation d'une empreinte physique, en l'occurrence une image d'un point imprimé sur le document. La binarisation d'images d'un même point pourra donner des formes légèrement différentes et donc des empreintes numériques différentes.

A noter qu'il est important de distinguer empreinte physique et empreinte numérique car dans un processus d'authentification automatique, la machine impliquée dans le processus de décision traite l'empreinte numérique.

Nous appelons *fonction d'extraction* $\mathcal{E}(\cdot)$ (équation II.1), toute fonction permettant d'extraire une empreinte numérique à partir d'une empreinte physique. Ainsi la fonction $\mathcal{E}(\cdot)$ est une fonction de condensation à sens unique qui résume la caractéristique d'un objet en une suite numérique. Dire que la fonction $\mathcal{E}(\cdot)$ est à sens unique signifie qu'il est difficile de reconstituer l'empreinte physique à partir de l'empreinte numérique. Cependant, la fonction $\mathcal{E}(\cdot)$ n'est pas une fonction de calcul d'empreinte au sens cryptographique du terme telle une fonction de hachage, en ce sens que, pour deux empreintes physiques proches, extraites d'un même objet, la fonction $\mathcal{E}(\cdot)$ donne

deux empreintes numériques similaires, cela à pour but de réduire l'influence du bruit d'acquisition. Ce comportement n'est pas celui d'une fonction cryptographique de hachage qui se caractérise par *l'effet* dit *d'avalanche*[1].

$$\begin{array}{rccc} \mathcal{E}: & \mathfrak{E}_{phy} & \longrightarrow & \mathfrak{E}_{num} \\ & EP & \longmapsto & EN = \mathcal{E}(EP). \end{array} \tag{II.1}$$

où $\mathfrak{E}_{num}$ représente l'ensemble des valeurs que peut renvoyer la fonction $\mathcal{E}(\cdot)$.,

Définition 4 (*FAR* et *FRR*)
Comme en biométrie [JRP04], la performance d'un système de traçabilité sécurisée basé sur une empreinte d'objet c'est à dire sa capacité à extraire une empreinte fiable, peut être évaluée aux moyens de deux paramètres statistiques : le taux de fausses acceptions, *FAR* (*False Acceptation Rate*), et le taux de faux rejets,*FRR* (*False Reject Rate*).

Le *FAR* est la probabilité que deux objets différents aient des empreintes numériques proches. Le *FAR* représente ainsi la capacité à distinguer deux objets différents appartenant à une même famille.

Le *FRR* mesure la probabilité d'obtenir des empreintes numériques éloignées pour un même objet. Il indique la capacité du système à identifier un objet donnée et à tolérer les variations des empreintes physiques, y compris celles induites par les conditions d'usage telles que les dégradations liées à la manipulation des objets ou leur vieillissement.

Définition 5 (Fonction d'appariement $\Phi(\cdot,\cdot)$)
Nous définissons par $\Phi(\cdot,\cdot)$ (équation II.2) la fonction booléenne qui compare deux empreintes numériques. $\Phi(\cdot,\cdot)$ modélise la fonction qui au bout du processus, permet de décider si deux empreintes numériques proviennent d'empreintes physiques similaires. La fonction Φ intègre le niveau de tolérance acceptable, fixé par les paramètres *FAR* et *FRR* .

Pour toute paire d'empreintes numériques EN_1, $EN_2 \in \mathfrak{E}_{num}$, $\Phi(EN_1, EN_2) = 1$ signifie que pour toute paire d'empreintes physiques EP_1, $EP_2 \in \mathfrak{E}_{phy}$ telle que $EN_1 = \mathcal{E}(EP_1)$ et $EN_2 = \mathcal{E}(EP_2)$ alors EP_1 et EP_2 sont similaires et proviennent surement du même objet. La probabilité minimale que EP_1 et EP_2 ne proviennent du même objet est définie par *FAR*. Inversement, lorsque $\Phi(EN_1, EN_2) = 0$, alors EP_1 et EP_2 sont complètement différentes avec une probabilité d'erreur inférieure à *FRR*.

$$\begin{array}{rccl} \Phi: & \mathfrak{E}_{num} \times \mathfrak{E}_{num} & \longrightarrow & \{0,1\} \\ & (EN_1, EN_2) & \longmapsto & \Phi(EN_1, EN_2) = \left\{ \begin{array}{ll} 1 \Leftrightarrow & Pr[EP_1 \approx EP_2] \geq 1 - FAR \\ 0 \Leftrightarrow & Pr[EP_1 \approx EP_2] \leq FRR \end{array} \right. \end{array} \tag{II.2}$$

1. L'effet avalanche se caractérise par le fait qu'en perturbant un seul bit en entrée, on obtient une sortie totalement différente, soit environ 1 bit sur deux de changé.

Définition 6 (Le fabriquant d'objets : *Trusted Manufacturer* $\mathcal{TM}$)
Nous définissons par $\mathcal{TM}$ l'autorité à laquelle appartiennent les objets à authentifier $\mathcal{OA}$, c'est à dire le tiers garant de l'authenticité des $\mathcal{OA}$. Le $\mathcal{TM}$ fabrique ainsi les objets originaux. Nous supposons qu'il est chargé d'effectuer le processus d'enregistrement de l'authentification de ces objets.

Nous supposons que le $\mathcal{TM}$ est fiable et sûr, c'est à dire qu'il effectue toutes les opérations nécessaires à l'enregistrement sans erreur et sans être affecté par une tentative d'attaque. Par analogie aux différents processus d'authentification introduits à la section I.3 du chapitre I, nous considérons que le $\mathcal{TM}$ joue aussi le rôle du laboratoire d'authentification (LA).

Définition 7 (Dispositif d'authentification autonome $\mathcal{DAA}$)

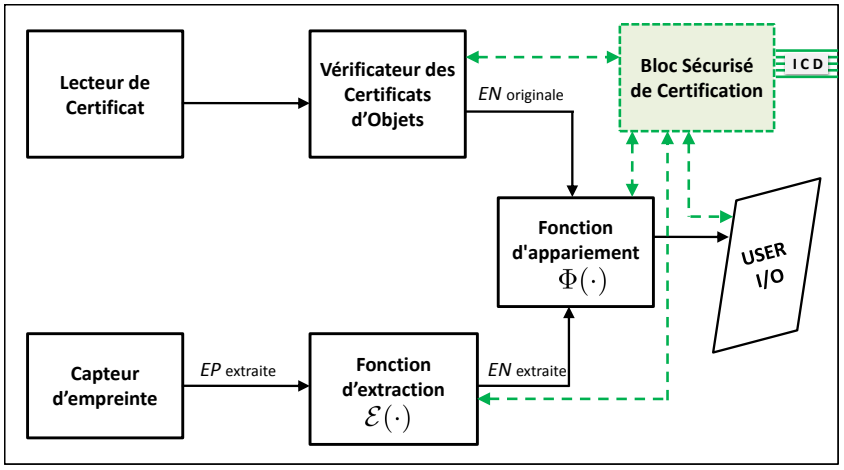

Figure II.1 *Architecture du Dispositif d'Authentification Autonome ($\mathcal{DAA}$)*
***ICD** : Interface pour les Connexions à Distance*

– o –

Pour définir la traçabilité sécurisée autonome, nous rassemblons les fonctions de lecture et de comparaison d'empreintes dans un seul dispositif nommé $\mathcal{DAA}$. Un $\mathcal{DAA}$ est donc un appareil électronique qui permet d'effectuer la vérification d'authenticité des $\mathcal{OA}$, en réalisant l'acquisition de l'empreinte physique d'un $\mathcal{OA}$ puis l'extraction de l'empreinte numérique correspondante. Nous avons défini les objets à authentifier (Définition 1) avec une zone porteuse d'informations (comme une étiquette ou une puce RFID). Le $\mathcal{DAA}$ permettra donc de lire le contenu de cette zone de stockage, dans le cas où les données de cette zone forment le certificat de l'objet. Le $\mathcal{DAA}$ doit disposer des fonctions et des clés permettant de vérifier un certificat.

Ces considérations conduisent à une architecture de $\mathcal{DAA}$ telle que représentée

sur la Figure II.1. Un capteur d'empreinte physique et un lecteur de certificat constituent les blocs matériels qui permettent à un $\mathcal{DAA}$ d'acquérir une empreinte physique et de lire un certificat sur l'$\mathcal{OA}$, respectivement. Le lecteur de certificat est le lecteur de la zone de stockage des informations qui peut être un lecteur de Code 1D ou 2D, ou un récepteur de puce RFID. Ces blocs sont liés respectivement à un bloc implantant la fonction d'extraction d'empreinte $\mathcal{E}(\cdot)$ et à un bloc de vérification de signature électronique. Le bloc de vérification de certificat électronique doit avoir les clés publiques nécessaires pour vérifier les certificats des $\mathcal{OA}$ (la clé publique du $\mathcal{TM}$). Le vérifieur de certificat et le bloc extracteur d'empreinte numérique sont connectés à un bloc implantant la fonction d'appariement $\Phi(\cdot,\cdot)$, qui comparera les empreintes numériques.

Le $\mathcal{DAA}$ contient également un bloc sécurisé, permettant la certification des autres blocs du $\mathcal{DAA}$. Ce bloc de certification contient l'identité du $\mathcal{DAA}$ qui est publiquement connu et une clé secrète partagée entre le $\mathcal{DAA}$ et l'autorité d'authentification du dispositif $\mathcal{DAA}$ (appelée le $\mathcal{CA}$ définie ci-dessous). Ce bloc doit gérer un protocole de communication avec le $\mathcal{CA}$ lors de certification des autres blocs du $\mathcal{DAA}$, et aussi surveiller le processus d'authentification des $\mathcal{OA}$.

Définition 8 (Autorité de certification $\mathcal{CA}$)
Nous définissons par le $\mathcal{CA}$ l'autorité de maintenance et de certification des dispositifs d'authentification autonome d'objets. Nous lui donnons ainsi la responsabilité de pouvoir vérifier la fiabilité et l'intégrité des $\mathcal{DAA}$.

Définition 9 (Agent vérifieur : $\mathcal{V}$)
Le rôle de l'agent vérifieur est de s'assurer de l'authenticité d'un $\mathcal{OA}$ en utilisant un $\mathcal{DAA}$ sans déplacement chez le $\mathcal{TM}$ et sans communication à une base de données distante. Il doit pouvoir vérifier la fiabilité d'un appareil $\mathcal{DAA}$ avant de l'utiliser, avec cette fois-ci une possibilité de communication externe vers l'autorité de certification $\mathcal{CA}$ des $\mathcal{DAA}$.

Dans ce rôle l'agent vérifieur est donc une personne dont le but est de disposer d'un dispositif fiable pour authentifier des objets $\mathcal{OA}$.

II.3 Hypothèses du protocole

Afin de poser le principe d'authentification autonome, il est important de décrire les hypothèses sur lesquelles sont basées la traçabilité sécurisée, et de définir de nouvelles hypothèses pour une traçabilité sécurisée autonome. La première des hypothèses est l'unicité de l'empreinte caractérisant l'objet. La seconde, moins couramment explicitée, exprime que la force de l'authentification repose sur les performances de la technologie de caractérisation et non sur le secret des empreintes qui sont publiquement accessibles. La troisième hypothèse est de considérer la fonction d'extraction $\mathcal{E}(\cdot)$(*cf.* Définition 3) comme une fonction à sens unique. La quatrième et dernière hypothèse est une propriété

logique qui à l'usage, permet à un agent vérifieur, de prévenir des dégradations sur le processus d'authentification.

Hypothèse 1 (*Unicité de l'empreinte*)
Deux objets $\mathcal{OA}$ conçus dans les mêmes conditions avec les mêmes matières premières ont toujours des empreintes physiques différentes.

Autrement dit, la technologie de caractérisation des objets doit avoir un *FAR* assez faible. Cette hypothèse assure que les objets ne sont pas reproductibles, et garantit que les objets non référencés par un $\mathcal{TM}$ peuvent être détectés.
Une autre façon d'exprimer cette hypothèse est de dire que, si les objets manufacturés d'une même famille ont des empreintes physiques similaires, alors cette famille aura une même empreinte numérique, possèdera un même identifiant et sera représentée par un seul objet à authentifier $\mathcal{OA}$. L'authentification dans ce cas permet de vérifier qu'un objet appartient à une famille donné et ne permet pas de distinguer intrinsèquement deux objets de la famille.

Hypothèse 2 (*Accès public à la méthode de lecture d'empreintes*)
Les méthodes et paramètres de prise d'empreintes des objets $\mathcal{OA}$ sont publiquement connues.

Cela signifie que la méthode d'acquisition de l'empreinte physique d'un objet et sa fonction d'extraction $\mathcal{E}(\cdot)$ sont publiques. Cela veut dire aussi que tout un chacun peut accéder à la technologie de caractérisation des $\mathcal{OA}$ et à ses performances (comme le *FAR* et le *FRR*), et est en mesure quelque soit l'$\mathcal{OA}$ d'obtenir son empreinte numérique .

Nous excluons ainsi de la traçabilité sécurisée autonome, toute technologie de marquage d'objets relevant de la *stéganographie*, c'est à dire consistant à "cacher de l'information" sur un objet pour en détecter ultérieurement sa présence. Ici, il nous faut contrôler la présence de l'information caractérisant un objet (son empreinte physique) et aussi s'assurer qu'elle est suffisamment proche de celle de référence, extraite par le fabricant $\mathcal{TM}$ de l'objet. La confidentialité des empreintes numériques de référence n'est donc pas nécessaire, seul importe de savoir si elles proviennent effectivement d'une autorité de confiance comme le $\mathcal{TM}$.

Hypothèse 3 (*Caractérisation à sens unique*)
Il est difficile de reconstituer une empreinte physique à partir d'une empreinte numérique connue et il est difficile de fabriquer un objet ayant une empreinte physique donnée.

Cette hypothèse permet à un processus d'authentification d'éviter les $\mathcal{OA}$ clonés, c'est à dire les objets produits en copiant un objet original référencé par un $\mathcal{TM}$ (leurres).

Hypothèse 4 (*Acquisition non contrôlable*)
L'acquisition de l'empreinte physique n'est pas contrôlable.

Cette hypothèse est sans doute la plus contraignante. Elle signifie que la fiabilité du capteur d'empreinte physique ne peut être ni vérifiée ni attaquée.

II.4 Modèle d'adversaire

II.4.1 Prise de contrôle de l'extraction

Comme nous l'avons souligné à la section I.3 du chapitre I, tout système d'authentification automatique constitue un intermédiaire potentiellement attaquable. Cela est particulièrement vrai en traçabilité sécurisée, dès lors que la vérification doit s'effectuer à travers un dispositif autre que l'oeil d'un expert.

Or, dans la majorité des technologies de caractérisation d'objets manufacturés, a fortiori dans un laboratoire d'authentification, un dispositif électronique est utilisé, au moins au niveau de l'acquisition de l'empreinte physique.

D'après l'hypothèse 4, une attaque au niveau de la prise d'empreinte signifie contrôler la fonction d'extraction $\mathcal{E}(\cdot)$. Autrement dit, si le capteur d'empreinte physique est un appareil photo numérique (APN), comme dans [ZWK03, IFA10], il ne serait pas possible pour le vérifieur de vérifier que l'appareil prend correctement les images ou pour l'attaquant de faire une prise d'images qui ne correspond pas au champ de vue. Cependant, un agent vérifieur $\mathcal{V}$ doit être capable de savoir si la fonction d'extraction $\mathcal{E}(\cdot)$ associée à l'APN extrait effectivement une empreinte valide à partir de l'empreinte physique acquise.

II.4.2 Violation de l'anonymat du vérifiant

D'après l'hypothèse 2 assurant le caractère publique de la méthode d'extraction d'empreinte d'un $\mathcal{OA}$; toute personne honnête (qui n'est pas forcément un agent vérifieur) disposant de moyens nécessaires comme un $\mathcal{DAA}$, peut vérifier honnêtement l'authenticité d'un objet donné.

C'est en fait là que réside une grande différence entre un processus de traçabilité sécurisée et un processus de biométrie. Pour une question de gestion de vie privée, et aussi pour le respect de la législation quelquefois, il est important dans un processus d'authentification biométrique de pouvoir assurer l'anonymat de l'empreinte à travers des agents assermentés et/ou des dispositifs sécurisés. A contrario pour un produit manufacturé tel qu'un médicament, il peut être utile qu'un tiers tel qu'un consommateur soit en mesure de vérifier son authenticité.
En revanche, dans un tel processus, divulguer conjointement l'identité du tiers et l'empreinte de l'objet pourrait être préjudiciable. Cette attaque est celle de l'anonymat de l'identité biométrique.

II.4.3 Déni de service

La principale faiblesse qu'il faut considérer, est la possibilité pour un adversaire de réaliser une *attaque par déni de service (DoS)* contre le processus d'authentification. En effet, pour empêcher un agent $\mathcal{V}$ de mener à bien un processus d'authentification, un adversaire peut, soit dégrader les objets $\mathcal{OA}$, soit empêcher les processus de vérification de la fiabilité du dispositif $\mathcal{DAA}$ ou tout simplement empêcher toute utilisation du $\mathcal{DAA}$ par l'agent $\mathcal{V}$.

D'autre part, pour vérifier l'authenticité d'un objet, il est nécessaire d'avoir accès à une empreinte de référence lui correspondant. Une attaque par déni de service lors d'une authentification à distance réalisée en dehors d'un laboratoire d'expertise, peut consister à suspendre l'accès aux empreintes de référence des objets. Dans la problématique d'authentification autonome, nous supposons que chaque objet porte son empreinte de référence, ce qui facilite une attaque par déni de service. Il suffit en effet à un attaquant de détruire l'empreinte de référence d'un objet $\mathcal{OA}$ donné.

II.5 Processus d'authentification d'objets

Le processus d'authentification que nous développons comporte deux phases conformes à celles de [ZWK03] : d'une part une phase d'enregistrement avec certification des objets, d'autre part une phase ultérieure de vérification de l'authenticité des objets. La phase d'enregistrement consistera à fabriquer des certificats cryptographiques, pour chacun des objets à authentifier. Un certificat d'$\mathcal{OA}$ comportera entre autre l'empreinte numérique de référence de cet objet. La phase de vérification consistera à vérifier en premier lieu le certificat de l'objet pour garantir que l'empreinte qu'il contient est la référence, puis à extraire une empreinte sur l'objet afin de la comparer à celle de référence.

II.5.1 Phase d'enregistrement et de certification des objets

Cette phase de certification (Figure II.2) doit s'effectuer en lieu sûr, chez une autorité de confiance : le $\mathcal{TM}$. La certification d'un objet débute immédiatement après la fabrication. Le $\mathcal{TM}$ extrait une empreinte physique de référence EP_r de l'objet (1), puis calcule une empreinte numérique EN_r de EP_r (2) en utilisant la fonction d'extraction $\mathcal{E}(\cdot)$:

$$EN_r = \mathcal{E}(EP_r). \tag{II.3}$$

EN_r représente l'empreinte numérique de référence de l'objet. Pour pouvoir authentifier l'objet, toute autre capture d'empreinte physique doit produire une empreinte numérique proche au sens de la fonction d'appariement $\Phi(\cdot,\cdot)$ de EN_r.

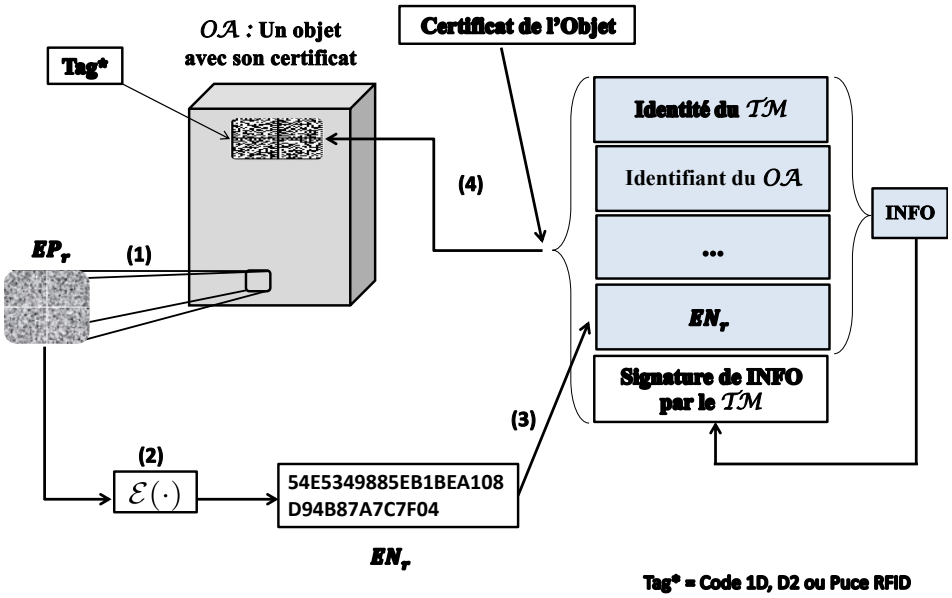

Figure II.2 *Processus d'enregistrement et de certification d'un objet*

– ○ –

La phase (3) consiste à créer un certificat pour l'objet, ce certificat contient principalement l'empreinte de référence EN_r de l'objet, et des informations d'identification de l'objet et de son fabriquant $\mathcal{TM}$ (selon l'hypothèse 1, chaque objet peut recevoir une identité unique). Selon la taille de stockage disponible et la nomenclature du $\mathcal{TM}$, le certificat peut contenir des précisions complémentaires telles que la date et le lieu production. Toutes ces données sont alors signées par le $\mathcal{TM}$, en utilisant un algorithme de signature électronique à partir d'une architecture de cryptographie à clé publique. L'empreinte de la signature est jointe en fin de certificat avant d'être écrit dans la zone de stockage de l'objet (4).

II.5.2 Phase de vérification des objets

La vérification de l'authenticité d'un objet $\mathcal{OA}$ (qui doit posséder un certificat) s'effectue en deux étapes. La première consiste à lire et vérifier le certificat. Si le certificat est valide, l'empreinte numérique qu'il contient est l'empreinte de référence de l'objet. La seconde étape, consiste à extraire une empreinte physique de l'objet et à calculer une empreinte numérique qui sera comparée à l'empreinte de référence en utilisant la fonction d'appariement $\Phi(\cdot,\cdot)$.

Cependant, la particularité de la traçabilité autonome est que toute la phase de vérification des objets doit s'effectuer à travers un $\mathcal{DAA}$. Et la première question que

doit se poser un agent vérifieur $\mathcal{V}$ concerne la fiabilité et l'intégrité du $\mathcal{DAA}$ lui-même. En effet, dans l'optique de garantir le résultat final du processus d'authentification, l'agent $\mathcal{V}$ doit pouvoir, avant toute phase de vérification d'objets, vérifier l'intégrité du $\mathcal{DAA}$ qu'il utilise.

Afin de vérifier un $\mathcal{DAA}$, nous proposons dans la suite de ce chapitre un protocole de communication distant entre un $\mathcal{DAA}$ et une $\mathcal{CA}$ destiné à vérifier la capacité du $\mathcal{DAA}$ à réaliser une authentification fiable d'une suite d'objets virtuels.

II.6 Protocoles de vérification d'intégrité du $\mathcal{DAA}$

Dans cette section, le processus de vérification d'un $\mathcal{DAA}$ s'effectue à travers son *Bloc Sécurisé de Certification*(BSC) (voir schéma de la Figure II.1). Il consiste à réaliser en premier lieu une authentification mutuelle entre le $\mathcal{DAA}$ et $\mathcal{CA}$ via un canal de télécommunication. Cette authentification établit une communication sécurisée à travers laquelle est vérifiée dans un second temps, la fiabilité et l'intégrité des fonctions du $\mathcal{DAA}$. Cette vérification d'intégrité du $\mathcal{DAA}$ consiste à réaliser une série d'authentification d'objets virtuels.

II.6.1 Authentification mutuelle du $\mathcal{DAA}$ et du $\mathcal{CA}$

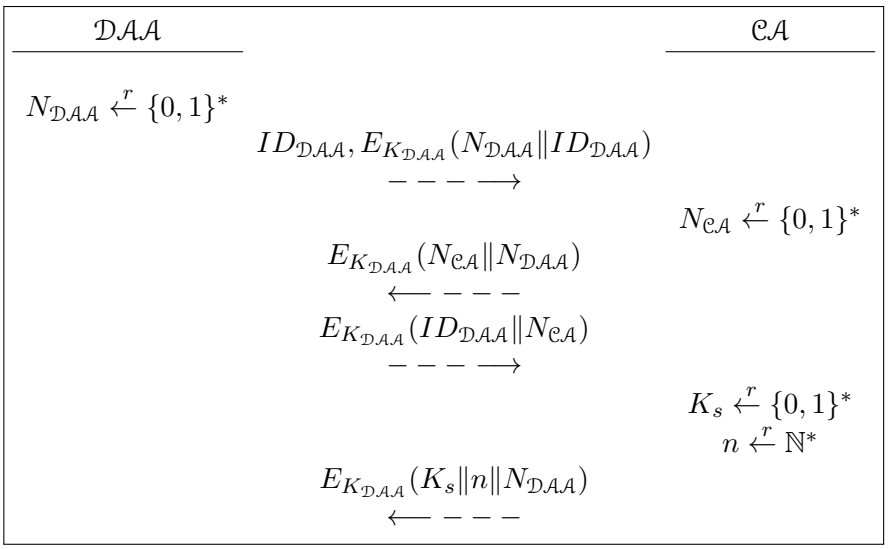

Figure II.3 *Authentification mutuelle entre le $\mathcal{DAA}$ et $\mathcal{CA}$*

– ∘ –

Un protocole tel que celui de la Figure II.3 permet au $\mathcal{CA}$ d'authentifier le $\mathcal{DAA}$ sur requête et au $\mathcal{DAA}$ d'avoir le certitude qu'il communique avec le $\mathcal{CA}$. L'authentification

que nous présentons est un protocole du type challenge-réponse entre $\mathcal{DAA}$ et $\mathcal{CA}$. La sécurité du processus d'authentification est garantie par une clé secrète $K_{\mathcal{DAA}}$ qui n'est partagé qu'entre le $\mathcal{CA}$ et le $\mathcal{DAA}$ (spécifiquement, le Bloc Sécurisé de Certification du $\mathcal{DAA}$).

La communication est initiée par le $\mathcal{DAA}$, des lors que l'agent vérifieur établit une liaison entre le $\mathcal{DAA}$ et le $\mathcal{CA}$. Le premier message est constitué d'un nombre $N_{\mathcal{DAA}}$ généré aléatoirement par le $\mathcal{DAA}$ pour chaque session (un "nonce") et de l'identifiant $ID_{\mathcal{DAA}}$ du $\mathcal{DAA}$. A partir des deux données $N_{\mathcal{DAA}}$ et $ID_{\mathcal{DAA}}$, le $\mathcal{DAA}$ forme un message, puis le chiffre avec la clé $K_{\mathcal{DAA}}$ avant de l'envoyer au $\mathcal{CA}$ avec son identifiant $ID_{\mathcal{DAA}}$.

En recevant le message provenant d'un $\mathcal{DAA}$, le $\mathcal{CA}$ lit de sa base de données la clé secrète $K_{\mathcal{DAA}}$ correspondant à l'identifiant $ID_{\mathcal{DAA}}$ reçu. Il déchiffre le message, puis vérifie la concordance de l'identifiant. Ensuite, le $\mathcal{CA}$ génère de même un nouveau nonce $N_{\mathcal{CA}}$. Il chiffre les nonces $N_{\mathcal{DAA}}$ et $N_{\mathcal{CA}}$ avec la clé secrète $K_{\mathcal{DAA}}$ et envoie le message chiffrée au $\mathcal{DAA}$.

En recevant un message du $\mathcal{CA}$, le $\mathcal{DAA}$ le déchiffre et vérifie son nonce $N_{\mathcal{DAA}}$ préalablement envoyé. Si la valeur de $N_{\mathcal{DAA}}$ reçue est correcte , alors du point de vue du $\mathcal{DAA}$ le $\mathcal{CA}$ est authentifié, c'est à dire que, le $\mathcal{DAA}$ est sûr que l'entité avec laquelle il communique, connait effectivement la clé secrète $K_{\mathcal{DAA}}$. Le $\mathcal{DAA}$ confirme alors son identité en renvoyant au $\mathcal{CA}$ un message chiffré de la nonce $N_{\mathcal{CA}}$ et de son identifiant.

Dans la dernière étape, le $\mathcal{CA}$ décrypte le message du $\mathcal{DAA}$. Si le message est correct, à savoir que le nonce $N_{\mathcal{CA}}$ est le même que celui généré lors de la session d'ouverture avec l'identifiant $ID_{\mathcal{DAA}}$, alors le $\mathcal{DAA}$ s'est aussi authentifié auprès du $\mathcal{CA}$. La phase de vérification de la fiabilité des composants du $\mathcal{DAA}$ peut démarrer.

Après ces échanges assurant la vérification de l'identité du $\mathcal{CA}$ et du $\mathcal{DAA}$. Le $\mathcal{CA}$ génère au hasard une clé de session K_s et un paramètre entier n (par exemple $n \in \{64, \cdots, 128\}$) puis les échanges avec le $\mathcal{DAA}$, en les chiffrant avec la clé $K_{\mathcal{DAA}}$. Nous utiliserons cette clé de session K_s (à la section II.6.2 ci-dessous) pour assurer la confidentialité des messages échangés lors de la vérification de l'intégrité du $\mathcal{DAA}$. L'entier n est le nombre maximal d'itérations que comptent effectuer le $\mathcal{CA}$ lors de cette vérification.

II.6.2 Vérification de la fiabilité du $\mathcal{DAA}$

Une fois que le $\mathcal{DAA}$ et le $\mathcal{CA}$ se sont authentifiés, le $\mathcal{CA}$ vérifie la capacité du $\mathcal{DAA}$ à effectuer correctement une authentification d'objets prédéfinis. Dans cette vérification, nous supposons que les fonctions de lecture du $\mathcal{DAA}$ qui sont le capteur d'empreintes physiques (cf hypothèse II.4.1)et le lecteur de certificat ne peuvent pas être corrompus. Cette étape de vérification concerne donc les fonctions du $\mathcal{DAA}$ telles le processus de vérifieur de certificat, la fonction d'extraction $\mathcal{E}(\cdot)$ et la fonction d'appariement $\Phi(\cdot, \cdot)$.

1: Choix de σ ; $i \leftarrow 1$; $index \leftarrow \sigma(i)$

2: $\mathcal{CA}$:
$$\begin{cases} EP \xleftarrow{r} DB_{EP} \\ [EN, b_{EN}] = \mathcal{F}_{EN}(EP) \\ [S, b_S] = \mathcal{F}_S(EN, index) \\ H_{\mathcal{CA}} = Hash(S \| b_{EN} \| b_S) \end{cases}$$

3: $\mathcal{CA} - - - \longrightarrow \mathcal{DAA}$: $E_{K_s}(EP \| EN \| index \| S)$

4: $\mathcal{DAA}$:
$$\begin{cases} \textbf{Si } index \leq n \textbf{ alors} \\ EN' \leftarrow \mathcal{E}(EP) \\ b'_{EN} \leftarrow \Phi(EN, EN') \\ b'_S \leftarrow check\_sign(EN \| index, S, K_{pb\_of\_TM}) \\ H_{\mathcal{DAA}} \leftarrow Hash(S \| b'_{EN} \| b'_S) \\ \textbf{sinon} \\ index = n+1 \Rightarrow Certification\ est\ CORRECTE \\ index = n+2 \Rightarrow Certification\ est\ FAUSSE \\ STOP \\ \textbf{fin si} \end{cases}$$

5: $\mathcal{DAA} - - - \longrightarrow \mathcal{CA} : E_{K_s}(H_{\mathcal{DAA}} \| index)$

6: $\mathcal{CA}$:
$$\begin{cases} \textbf{Si } H_{\mathcal{CA}} != H_{\mathcal{DAA}} \textbf{ alors } index \leftarrow n+2 \\ \textbf{sinon} \\ i \leftarrow i+1 \\ \textbf{if } i \leq n \textbf{ alors } index \leftarrow \sigma(i) \\ \textbf{sinon } index \leftarrow n+1 \\ \textbf{fin Si} \\ \textbf{fin Si} \\ Goto\ step\ 2: \end{cases}$$

Algorithme 1: *Protocole de vérification de la fiabilité du $\mathcal{DAA}$*

L'idée principale consiste à utiliser ces composantes pour réaliser une authentification d'un nombre pré-défini d'**Objets Virtuels (OV)**, envoyés directement au bloc de certification du $\mathcal{DAA}$. Pour cela, nous supposons que le $\mathcal{CA}$ à une base $\mathcal{DB}_{EP}$ d'empreintes physiques et la capacité de signer avec la clé du $\mathcal{TM}$. Le $\mathcal{CA}$ doit donc connaître la clé privée de signature de certificat utilisée par le $\mathcal{TM}$ pour la famille d'objets en question. Avec cette base de données, le $\mathcal{CA}$ crée aléatoirement un objet virtuel défini par une empreinte physique EP_{OV}, une empreinte numérique EN_{OV}, et une signature de l'empreinte numérique. L'objet virtuel ainsi crée peut être authentique, c'est à dire qu'une empreinte numérique valide (enregistrée) peut être extraite de l'empreinte physique EP_{OV} via la fonction $\mathcal{E}(\cdot)$) et que la signature de EP_{OV} a été faite à partir de la clé du $\mathcal{TM}$.

Indexation des objets virtuels
Dans le protocole de l' Algorithme 1, chaque objet virtuel est indexé par un nombre entier *index*. Cette indexation permet de distinguer les objets virtuels utilisés pendant une même session. Les valeurs que prend *index* varient aléatoirement entre 1 et $n+2$. Une valeur de *index* supérieure à n, notifie au BSC du $\mathcal{DAA}$ la fin du protocole et la décision du $\mathcal{CA}$ concernant la fiabilité des composantes du $\mathcal{DAA}$.
Le but de l'indexation des objets virtuels est de les identifier les uns des autres. Les valeurs de *index* auraient aussi pu être des nombres aléatoires dans $\mathbf{N}$. Il faudrait alors prévoir des valeurs particulières d'indexation permettant au $\mathcal{CA}$ de notifier au $\mathcal{DAA}$ la fin du protocole sans avoir à changer la forme des messages.
Ici nous choisissons donc de construire les valeurs de *index* à partir du paramètre n. Les valeurs de *index* sont choisis aléatoirement entre 1 et n de sorte que *index* ne dépende pas de l'ordre d'envoi au $\mathcal{DAA}$. La fin des échanges est notifiée avec une valeur de $index > n$ où *index* a le même format que n.

Nous associons une permutation σ sur $\{1, 2, \cdots, n\}$ à la façon dont le $\mathcal{CA}$ choisi d'indexer ses objets virtuels : si $index = \sigma(i)$ alors le i-ème objet virtuel est indexé par $\sigma(i)$.
Ainsi, après avoir fixé la valeur du nombre maximal n d'objets virtuels à créer, la première étape du protocole pour le $\mathcal{CA}$ consiste à choisir aléatoirement la permutation σ puis la première valeur $\sigma(1)$ de *index*.

Require: EP
$EN_0 \leftarrow \mathcal{E}(EP)$
$EP_1 \xleftarrow{r} \{x \in \mathcal{DB}_{EP}$ *tel que* $\Phi(\mathcal{E}(x), EN_0) = 0\}$
$EN_1 \leftarrow \mathcal{E}(EP_1)$
$b_{EN} \xleftarrow{r} \{0, 1\}$
Si $b_{EN} = 1$ **alors** $EN \leftarrow EN_0$.
Si $b_{EN} = 0$ **alors** $EN \leftarrow EN_1$.
return $[EN, b_{EN}]$

Algorithme 2: *Fonction* $\mathcal{F}_{EN}$

Authentification des objets virtuels
Les étapes 2 à 6 du protocole (décrit en Algorithme 1) sont répétées jusqu'à la fin des échanges.

L'étape 2 est la création d'un objet virtuel par le $\mathcal{CA}$ ce qui signifie choisir aléatoirement une empreinte physique EP dans sa base de données $\mathcal{DB}_{EP}$ d'empreintes physiques (appartenant à plusieurs objets de la même catégorie), activer la fonction de sélection $\mathcal{F}_{EN}(\cdot)$ pour générer une empreinte numérique EN et un témoin b_{EN} indiquant la relation entre les deux empreintes puis de la fonction $\mathcal{F}_S(\cdot)$ (Algorithme 3) pour générer la signature S de l'empreinte numérique EN et d'un index, avec un témoin b_S indiquant la validité de la signature. Avec les témoins b_{EN} et b_S, le $\mathcal{CA}$ crée un *checksum*

Require: $EN, index$
$K^0_{pr} \leftarrow$ clé privée de $\mathcal{TM}$
$K^1_{pv} \xleftarrow{r} \{x \in$ ensemble des clé privées différentes de$K^0_{pv}\}$
$b_S \xleftarrow{r} \{0,1\}$
Si $b_S = 1$ **alors** $K_{pv} \leftarrow K^0_{pv}$.
Si $b_S = 0$ **alors** $K_{pv} \leftarrow K^1_{pv}$.
$S = signature(K_{pv}, EN\|index)$
S est la signature de $EN\|index$ en utilisant K_{pv}.
return $[S, b_S]$

Algorithme 3: *Fonction* $\mathcal{F}_S$

d'authenticité $H_{\mathcal{CA}} = Hash(S\|b_{EN}\|b_S)$.

A l'étape 3, le $\mathcal{CA}$ chiffre le message $EP\|EN\|S$ avec la clé de session K_s et envoie le message chiffré au $\mathcal{DAA}$.

Du côte du $\mathcal{DAA}$, seul le bloc BSC connait la clé de session et le paramètre n. En déchiffrant le message du $\mathcal{CA}$, le $\mathcal{DAA}$ exécute l'étape 4 qui a pour objectif de vérifier l'authenticité de l'objet virtuel que décrivent les données déchiffrées.
Si $index \in \{1, 2, \cdots, n\}$, c'est à dire que $index = \sigma(i)$ pour un $i \leq n$, alors l'authentification de l'objet virtuel s'effectue comme suit :

- en utilisant la composant d'extraction $\mathcal{E}(\cdot)$, le $\mathcal{DAA}$ calcule une empreinte numérique EN' de l'EP reçu,
- puis le $\mathcal{DAA}$ calcule b'_{EN}, le résultat que retourne la fonction d'appariement $\Phi(\cdot,\cdot)$ avec EN et EN' comme entrées,
- le $\mathcal{DAA}$ vérifie également la signature S reçu de $DF\|index$ avec le bloc de vérification de certificats d'objets qui détient la clé publique utile, notée $K_{pb\_of\_TM}$. La fonction de vérification de signature et le résultat de cette vérification sont respectivement notés $check\_sign()$ et b'_S.
- Le $\mathcal{DAA}$ crée (dans son BSC), à partir de b'_{EN} et b'_S, un témoin d'authentification en calculant un $Hash$ $H_{\mathcal{DAA}}$ de l'ensemble S, b'_{EN} et b'_S.

$index > n$ notifie la fin du processus. Selon la convention choisie, $index = n + 1$ signifie que le $\mathcal{CA}$ valide la faculté des composantes du $\mathcal{DAA}$ à réaliser l'authentification d'objet. La composante de certification arrête alors la communication avec le $\mathcal{CA}$ et prend une configuration normale permettant l'utilisation de l'appareil par un agent $\mathcal{V}$ pour l'authentification des objets. $index = n + 2$ signifie que le $\mathcal{CA}$ juge que le $\mathcal{DAA}$ n'est pas fiable et que la BSC doit bloquer toute utilisation du $\mathcal{DAA}$ après avoir arrêté la communication.

A l'étape 5, le $\mathcal{DAA}$ envoie de façon confidentielle au $\mathcal{CA}$ un message composé du *checksum* $H_{\mathcal{DAA}}$ et $index$.

A l'étape 6, après avoir déchiffré le message reçu du $\mathcal{DAA}$, le $\mathcal{CA}$ compare le *checksum* $H_{\mathcal{DAA}}$ à celui calculé à l'étape 2 : $H_{\mathcal{CA}}$. Si les valeurs sont les mêmes et que le nombre maximal d'essais n'est pas atteint ($i < n$), le $\mathcal{CA}$ continue le protocole à l'étape 2 avec $index = \ sigma(i+1)$, ou avec $index = n+1$ pour signaler la fin du protocole. Si par contre il rejette $H_{\mathcal{DAA}}$, il notifie son choix au $\mathcal{DAA}$ avec $index = n+2$.

II.6.3 Sécurité du protocole

Authentification mutuelle entre le $\mathcal{DAA}$ et $\mathcal{CA}$
Le premier objectif du protocole d'authentification mutuelle (Figure II.3) est d'assurer au $\mathcal{CA}$ qu'il communique bien avec le $\mathcal{DAA}$ identifié par $ID_{\mathcal{DAA}}$, et réciproquement, en prouvant qu'il partage la même clé secrète $K_{\mathcal{DAA}}$. Cette authentification est basée sur le fait que seuls le $\mathcal{CA}$ et le $\mathcal{DAA}$ partagent cette clé. Le deuxième objectif du protocole est de partager une clé de session K_s et un paramètre secret n.

Dans le but de valider que les objectifs sont atteints grâce au protocole de la Figure II.3, considérons un attaquant actif [DY81] c'est à dire susceptible d'écouter, enregistrer, analyser, manipuler et introduire les messages échangées sur la ligne de communication entre le $\mathcal{CA}$ et le $\mathcal{DAA}$. En revanche, on suppose que l'attaquant ne peut pas obtenir d'information à partir d'un message chiffré sans avoir la clé confidentielle (hypothèse de *chiffrement parfait*). La validation est faite en utilisant ProVerif [BS11, BP05], outil de vérification automatique de protocoles cryptographiques. L'exécution de l'implémentation donnée à la Figure II.4 indique que le test des propriétés à vérifier est réussi.

Protocole de vérification de fiabilité du $\mathcal{DAA}$
L'objectif du protocole de vérification de fiabilité est de déterminer la capacité du $\mathcal{DAA}$ à authentifier des objets, en testant ses composantes. Ce protocole est basée sur l'hypothèse que le bloc BSC est sûr et fiable quelque soit l'état du $\mathcal{DAA}$. Comme le BSC gère l'utilisation du $\mathcal{DAA}$, toutes altérations sur le BSC doit rendre le $\mathcal{DAA}$ inopérant c'est à dire bloquer l'utilisation du $\mathcal{DAA}$ par un agent vérifieur.

L'objectif principal de ce protocole est de détecter une altération ou une modification des fonctions internes du $\mathcal{DAA}$ telle que par exemple l'installation d'un bloc "intelligent" entre le bloc de vérification des certificats d'objets et le BSC. Une modification frauduleuse de fonctions internes d'authentification ou de transformation du dispositif en un dispositif délivrant un "Authentique" systématique en sortie, sera détecter par le $\mathcal{CA}$ qui le notifiera au BSC. En effet, seul un calcul effectif des témoins b'_{EN} et b'_S par les composantes valides permettra au $\mathcal{CA}$ de valider de façon certaine tous les *checksum* $H_{\mathcal{DAA}}$ provenant du $\mathcal{DAA}$.

Par contre, l'ajout d'un bloc intelligent entre le BSC et les fonctions du $\mathcal{DAA}$, programmé pour répondre dynamiquement en fonction de l'utilisation du $\mathcal{DAA}$, ne peut

```
free CanalCommunication .

fun host/1.
private reduc getkey(host(x)) = x.

(* Shared key cryptography *)
fun encrypt/2.
reduc decrypt(encrypt(x,y),y) = x.

(* Secrecy assumptions *)
not KDAA.

private free secretDAA.
query attacker:secretDAA.

let processDAA =
new NDAA;
out(CanalCommunication, (host(KDAA),encrypt((NDAA,host(KDAA)), KDAA)));
in(CanalCommunication, m1_fromCA);
    let (nCA, =NDAA) = decrypt(m1_fromCA, KDAA) in
out(CanalCommunication, encrypt((host(KDAA),nCA), KDAA));
in(CanalCommunication, m2_fromCA);
let (Ks, n, =NDAA) = decrypt(m2_fromCA, KDAA) in
out(CanalCommunication, encrypt(secretDAA, Ks));
out(CanalCommunication, encrypt(secretDAA, n)).

let processCA =
in(CanalCommunication, (idDAA,m1_fromDAA));
let K = getkey(idDAA) in
let (=idDAA, nDAA) = decrypt(m1_fromDAA,K) in
new NCA;
out(CanalCommunication, (encrypt((NCA,nDAA),K)));
in(CanalCommunication, m2_fromDAA);
let (=idDAA, =NCA) = decrypt(m2_fromDAA, K) in
new Ksession;
new n;
out(CanalCommunication, encrypt(((Ksession,n),nDAA),KDAA)).

process new KDAA;
 let hostDAA = host(KDAA) in
  out(CanalCommunication, hostDAA);
((!processDAA) | (!processCA))
```

Figure II.4 *Script ProVerif du protocole d'authentification mutuelle en le $\mathcal{CA}$ et le $\mathcal{DAA}$.*

– ○ –

être détecté par le protocole de vérification de fiabilité que nous venons de présenter. Un tel composant permettrait d'authentifier des objets virtuels en court-circuitant les fonctions du $\mathcal{DAA}$. Cependant, nous pensons qu'ajouter a posteriori un tel composant à un système électronique défini sans altération détectable du matériel, est difficile. Cette vulnérabilité ne sera pas considérée ici.

Mis à part le risque d'un tel composant intelligent cablé, capable de contrôler en interne le $\mathcal{DAA}$, en externe (au $\mathcal{DAA}$ entre le $\mathcal{CA}$ et le $\mathcal{DAA}$), toute attaque autre qu'un déni de service (par renvoi de faux messages au $\mathcal{CA}$ et au $\mathcal{DAA}$) ne peut permettre à un attaquant de faire valider auprès du $\mathcal{CA}$, la fiabilité d'un $\mathcal{DAA}$ altéré. Pour un observateur extérieur au $\mathcal{DAA}$, qui n'a pas connaissance de la clé K_s, les messages circulant entre le BSC du $\mathcal{DAA}$ et le $\mathcal{CA}$ resteront indéchiffrables.

Quant à la clé K_s, elle transite au $\mathcal{DAA}$ à la fin de la phase d'authentification mutuelle ($\mathcal{DAA} - \mathcal{CA}$). Elle permet d'assurer le confidentialité des messages échangés entre le $\mathcal{DAA}$ et le $\mathcal{CA}$ lors du contrôle de la fiabilité du $\mathcal{DAA}$ (Algorithme1). Même si dans la présentation faite du protocole, la clé de session K_s reste inchangée, le $\mathcal{CA}$ et le $\mathcal{DAA}$ peuvent tout en changer conformément à gestion prédéfinie des clés. Ce qui importe ici est que tous les messages échangés entre le $\mathcal{DAA}$ et le $\mathcal{CA}$ le soient de façon confidentielle.

II.7 Conclusion

Nous montrons dans ce chapitre qu'une authentification autonome ne peut se réaliser par un agent humain que si cet humain a auparavant confiance en les outils / dispositifs impliqués dans l'authentification. Le processus de traçabilité que nous venons de décrire est une solution intermédiaire vers la traçabilité sécurisée et autonome. Nous le situons entre un processus d'authentification à distance et un processus d'authentification autonome. L'agent vérifieur $\mathcal{V}$ établit avant toute vérification d'objet un canal de communication entre le $\mathcal{DAA}$ et $\mathcal{CA}$ pour vérifier la fiabilité du $\mathcal{DAA}$. **L'avantage de cette procédure vis à vis d'une authentification à distance, est qu'avec une seule connexion distante $\mathcal{DAA} - \mathcal{CA}$, l'agent $\mathcal{V}$ peut procéder à l'authentification d'une série d'objets manufacturés sans connexions distantes.**

Le challenge est maintenant de rendre moins nécessaires les communications auprès du $\mathcal{CA}$ en vue de réaliser une vérification. Il s'agirait qu'un agent vérifieur humain puisse lui-même vérifier la fiabilité et l'intégrité de son dispositif $\mathcal{DAA}$. De façon général, cela revient à ce qu'un être-humain puisse lui-même détecter toute falsification ou non conformité d'un matériel électronique opérant de manière autonome. Cette problématique est le sujet principal de la deuxième partie.

Deuxième partie

Authentification autonome d'un système embarqué

Chapitre III

Etat de l'Art sur l'authentification de systèmes embarqués

III.1 Introduction : pourquoi authentifier un système embarqué ? Notre approche.

Un système embarqué est un système électronique spécialisé dans une tâche bien précise. Il peut évoluer de façon autonome, ou au sein d'un ensemble plus vaste pouvant comporter d'autres systèmes électroniques ou mécaniques. De nos jours, les systèmes embarqués font intégralement partie de notre quotidien à travers l'usage par exemple de la téléphonie mobile ou de la domotique mais aussi à travers la démarche du citoyen, lors d'un vote électronique, ou du consommateur, avec la lecture de données sur les produits manufacturés, par exemple.

La diversité d'usage des systèmes embarqués induit plusieurs contraintes pour les concepteurs et les utilisateurs. Traditionnellement, ces contraintes concernent principalement la sûreté de fonctionnement, l'encombrement et la consommation. Cependant la gestion de la sécurité devient crucial aujourd'hui. La question de l'authentification se révèle une problématique incontournable pour le concepteur et nécessaire pour l'utilisateur final qui doit avoir confiance dans le bon fonctionnement du système.

Un système embarqué comporte plusieurs composants électroniques. La sécurité d'un tel système repose donc d'abord sur celle de ses composants qui peuvent être vérifiés individuellement par l'utilisateur. Une vérification plus complète est réalisée lorsque l'architecture du système comporte un composant de plus haut niveau dédié à la sécurité globale appelé ici **composant d'authentification**, chargé d'authentifier les autres composants du système embarqué, l'utilisateur ayant pour tâche de vérifier l'authenticité du composant d'authentification. C'est cette approche que nous suivrons pour sécuriser un système embarqué.

Nous nous sommes focalisés sur les composants dits configurables (FPGAs) ou programmables (microcôntroleurs) que l'on trouve généralement dans les systèmes embarqués. Le comportement de ces composants dépend principalement des données qui servent à les configurer, respectivement à les programmer. Dans la suite, nous utiliserons le terme de **composant configurable** pour désigner indistinctement les composants programmables et les composants configurables.

Nous consacrerons donc ce chapitre à présenter l'authentification de tels systèmes embarqués à travers la littérature. Nous mettrons tout au long de ce chapitre, l'accent sur une intervention active de l'utilisateur humain dans cette authentification. Par le principe des jetons d'authentification et de l'authentification mutuelle entre machines, nous expliquerons comment cette problématique d'authentification Machine-Homme est abordée dans la littérature. Ce chapitre se terminera par une section sur les PUFs que nous avons introduit au chapitre I comme élément de caractérisation (ou empreinte caractéristique) d'un circuit électronique.

III.2 Sécurité d'un composant configurable

Un système embarqué utilise généralement un composant tel qu'un microcontrôleur ou un FPGA (Field-programmable Gate Array). Dans ce cas, la fonction que décrit le *firmware* du microcontrôleur ou du FPGA, est en partie ou entièrement configurable dans le matériel. Cette configuration dépend de données stockées dans une mémoire non volatile (NVM : Non Volatile Memory)- i.e. une mémoire morte (ROM) ou une mémoire FLASH. Pour un microcontrôleur d'un système embarqué, le contenu de la NVM sert en effet à configurer la RAM qui y est intégrée, déterminant ainsi son comportement. De même, l'architecture d'un FPGA dépend intégralement d'un flux numérique appelé *bitstream* qui est stocké dans une NVM. Dans ces deux cas, à la mise sous tension, la RAM de ce composant (FPGA/microcontroleur) est chargée par un contenu provenant d'une mémoire non volatile (NVM).

Pour assurer sécurité et fiabilité d'un tel système, la protection des données de la NVM est un point essentiel à étudier, au moins lorsqu'on considère - ce qui est le cas ici - que l'implémentation d'un protocole et de fonctions cryptographiques dans un composant électronique n'induit pas d'autres faiblesses de sécurité telles des fuites d'informations (lors de l'exécution des algorithmes par le composant), des vulnérabilités aux attaques par *canaux cachés* ou par injections de fautes. Ces genres d'attaques sont en général possibles sur tout système électronique.

Nous pouvons distinguer deux familles de vulnérabilités des données de configuration d'un composant (FPGA ou microcontroleur). La première concerne leur **confidentialité** et la seconde, leur intégrité répondant ainsi à la question : *est-ce-que le composant est conforme à sa spécification d'origine ?* Cela nécessite une **authentification** efficace de ces données.

III.2.1 Confidentialité du contenu

La première des vulnérabilités, due essentiellement à la faculté de configuration des composants que nous considérons (les FPGA ou microcontrôleurs configurables) est la faculté de copier/dupliquer les données de configuration du composant. Pour les FPGA, différents attaques, telles que le clonage de la SRAM du FPGA, l'ingénierie inverse du bitstream sont décrites dans [WGP04, Dri09]. L'objectif d'un attaquant, outre de chercher à extraire du circuit des informations secrètes, serait de chercher à cloner les données de la NVM, ou une partie de ces données à des fins non autorisées par l'entité qui détient la propriété intellectuelle sur ces données.

Des méthodes permettent ainsi, de chiffrer le bitstream d'un FPGA afin de protéger la propriété intellectuelle (IP=Intellectual Property) du développeur. Des publications comme [Aus95, Kea01, BGB06] décrivent les différents approches suivies dans la littérature pour lutter contre de telles vulnérabilités. Les données de configuration restent alors chiffrées par une clé secrété K_{ENC}, que seuls le module **Contrôleur de Configuration (CC)** du composant (microcontroleur ou FPGA) et le développeur du système (et d'autres autorités de confiance selon le protocole proposé) connaissent.

En pratique, pour des raisons évidentes de survie économique, la gestion de la propriété intellectuelle qui est liée ici à la confidentialité des données de configuration, est souvent traitée en priorité lorsqu'elle n'est pas la seule problématique à traiter. Comme nous venons de le voir, la littérature décrit différents protocoles proposant des solutions acceptables quant à la confidentialité du contenu de configuration d'un composant. Aussi nous ne traiterons pas de cette question dans la suite. Sauf mention du contraire, nous supposerons que les données de la NVM sont publiques.

III.2.2 Authentification du contenu par le composant

Dans la littérature concernant les FPGA, la nécessité de l'authentification du bitstream de FPGA fut introduit [PG06, Dri07] afin de prévenir les FPGA contre les modifications du bitstream et d'empêcher le FPGA d'être configuré par un bitstream non conforme.

L'idée principale d'une authentification du contenu de la NVM, consiste à calculer dans un premier temps (c'est à dire lors de la phase d'enregistrement du processus d'authentification) un code **MAC** (Message Authentication Code) de ce contenu. Ce MAC représente l'empreinte caractéristique du contenu de la NVM qui servira à vérifier l'authenticité du contenu. Ce MAC est obtenu à partir d'une clé secrète K_{MAC} propre au composant. Seule le composant et les entités qui possèdent cette clé K_{MAC} peuvent vérifier l'authenticité du contenu de la NVM à partir du code MAC. Cette empreinte MAC est ensuite stocké dans la mémoire NVM avec les données qu'il caractérise. Il s'agit alors pour un composant de vérifier ce MAC pendant les processus de configuration, c'est à dire de recalculer à partir du secret K_{MAC} un nouveau MAC,

afin de comparer celui-ci avec le premier MAC (calculé lors de la phase d'enregistrement).

Nous présentons dans les deux sections ci-dessous, comment un composant authentifie le contenu de la NVM avant de le charger. Nous décrivons en premier lieu le principe des protocoles qui se basent sur un calcul simple de MAC du contenu de configuration. Parce que ces protocoles sont vulnérables aux attaques par rejeu, nous décrivons dans un second temps un principe d'authentification permettant d'y remédier.

III.2.2.1 Authentification par calcul simple de MAC

Dans cette section, nous parlons de l'authentification directe et simple à travers un code MAC qui consiste à calculer et à vérifier, à chaque configuration du composant, le code MAC des données à charger. Ce calcul de MAC et sa vérification doivent être faits par le **contrôleur de configuration** (CC) du composant microcontrôleur ou FPGA. Si cette vérification échoue (c'est à dire si les deux MACs diffèrent), la configuration du composant doit avorter. Ce type de protocole d'authentification, suppose que le CC ait une zone de stockage sécurisée, car il contient la clé secrète K_{MAC} de calcul du MAC. De même le CC doit avoir toutes les fonctions de calcul et de comparaison de MACs. De plus le CC doit permettre dans le protocole, le chargement des données de la mémoire NVM vers la mémoire RAM intégrée dans le composant.

Dans la littérature concernant les FPGAs, ce contrôleur de configuration (CC) est appelé Code/Logique de Configuration (*Configuration Logic*) dans [?] ou Mécanisme Sécurisé de Mise-à-jour (*Secure Update Mechanism*) dans [BET08]. Cette appellation diffère selon les auteurs, car chacun d'eux associe ce module à des fonctions différentes. Cependant, en faisant abstraction des autres fonctions, leur rôle principale est la vérification du bitstream avant configuration du FPGA.

Les dernières circuits Actel [Act10] et la série 6 de Xilinx (Spartan6 et Virtex6) [Xil10] incluent un mécanisme cryptographique pour garantir l'authenticité de leur bitstream de configuration. Concrètement, les circuits Actel implantent un bloc AES servant au contrôle d'intégrité du bistream (le calcul de MAC est fait à travers l'algorithme de chiffrement AES) : un code MAC accompagne le bitstream lors du stockage, et le CC vérifie le MAC avant toute configuration du circuit.

Les composants de la Série 6 des FPGA Xilinx incluent également un module de vérification d'authenticité. Dans les circuits de la famille Virtex-6 [Xil10], le calcul de MAC de ce module est basé sur une fonction HMAC dépendant de la fonction de hachage SHA-1. Lors de la configuration, le fonction HMAC/SHA256 dans le FPGA calcule le MAC puis le compare au MAC présent dans la NVM. Si les deux MAC ne correspondent pas, le CC désactive l'interface de configuration, et bloque tout accès au FPGA.
De même, pour le microcontrôleur Maxim Zatara ZA9L1 [Max09] (un microcontrôleur ARM sécurisé de 32-bits), l'authentification du code de configuration de la RAM, se

fait à partir d'un HMAC (calculé selon SHA256).

III.2.2.2 Authentification et gestion des mises à jour

Un protocole d'authentification qui ne vérifie que le code MAC du contenu de la configuration, ne répond pas complètement à l'authentification d'un composant susceptible d'être reconfigurée et mise à jour. En effet, dans le principe d'authentification précédemment présenté, le composant **vérifieur** n'a aucun élément pour différencier deux contenus. Ce principe permet seulement de vérifier l'intégrité du contenu à travers le MAC, mais ne permet aucune *identification* du contenu de la NVM. Ce manque d'identification du contenu de la NVM par le composant, peut conduire le composant à accepter un rejeu d'une ancienne version des données de la NVM. C'est à dire, qu'une ancienne version (avec son code MAC) authentique, mais admettant des failles de sécurité, pourrait être utilisée pour configurer un composant qui constituerait une *attaque par rejeu*.

Pour pallier cette faiblesse, il convient de conférer au composant la capacité d'identifier dans le temps la version de données autorisée à le configurer. Choisir comme élément d'identification, le numéro de version (ou un compteur de version) est une solution simple et efficace, comme le propose Badrigans *et al.* dans [BET08]. Le protocole décrit par Badrigans *et al.* propose de stocker dans le *CC* (qu'il nomme *SEM : Secure Update Mechanism*), en plus de la clé secrète K_{MAC}, le numéro de version correspondant à la dernière mise à jour de données authentiques. Contrairement à la clé K_{MAC}, ce numéro de version n'est pas supposé être secret, mais il y doit être stocké de façon sure. L'empreinte MAC calculée lors des phases d'enregistrement et de vérification intègre, en plus des données de configuration, le numéro de version servant à l'identifier.

III.2.3 Authentification à distance du composant et de son contenu

L'utilisateur final d'un système embarqué qui intègre un circuit FPGA définit l'**entité finale** (un être humain ou un dispositif électronique). Il utilise les fonctions implantées dans le circuit par le développeur avec une spécification donnée. Cet utilisateur final est formellement reconnu par Drimer [Dri09] (appelé *System Owner*) dans son modèle d'usage de FPGA.

La première question que peut se poser un utilisateur final est : comment identifier un système dans un ensemble de systèmes embarqués semblables ? En réduisant la question aux composants (les composants d'authentification de ces systèmes embarqués), la formulation devient : comment l'utilisateur final peut-il identifier un composant (FPGA ou microcontrôleur) parmi un ensemble de composant de la même famille ? La question qui suit immédiatement est : comment savoir si la configuration d'un composant tel composant est conforme à sa spécification d'origine ?

Dans tous les mécanismes d'authentification ([BET08, Xil10, Act10]) présentés à la section III.2.2 précédente, l'utilisateur final d'un composant est **passif**. L'authentification du composant, dont dépend l'authentification du système embarqué, n'est jouée qu'au niveau du composant, autrement dit le composant s'**auto-authentifie**. Cependant, un adversaire malveillant peut alors modifier dans le système embarqué, à la fois le composant (FPGA ou microcontrôleur) de sécurité et le contenu de la NVM qui doit le configurer. L'auto-authentification d'un tel composant n'est pas suffisante pour assurer l'authentification du système embarqué. C'est ce que nous appelons une **attaque par substitution** de composants.

Une première solution pour l'utilisateur est qu'il fasse appel à un **tiers de confiance distant**. Dans ce cas, l'utilisateur connecte à chaque phase de vérification, son système embarqué à une autorité de confiance externe pour réaliser simultanément l'authentification du composant et celle de son contenu de configuration. Ainsi dans la littérature, les protocoles d'authentification en ligne comme dans [Dri07, BET08] auprès d'une autorité de confiance peuvent permettre à un utilisateur final d'authentifier un composant et son contenu.

Par contre, dans le cadre d'une utilisation off-line, par exemple lorsqu'une configuration a lieu à la mise sous tension du composant, l'utilisateur final d'un composant n'a pas la possibilité d'authentifier lui même ce composant et son contenu de configuration. Dans l'état de l'art actuel (FPGA) l'utilisateur final n'a aucun rôle dans l'authentification du composant et son contenu de configuration.
Cependant il existe dans la littérature des protocoles d'authentification d'une machine par un utilisateur humain, où l'utilisateur à un rôle actif. Nous consacrerons la section suivante III.3 à introduire le principe de ces protocoles.

III.3 Authentification d'une machine par un Humain

De nos jours, diverses tâches automatiques doivent être assurées en off-line (ou avoir une autonomie en mode off-line). Ce contexte d'utilisation, peut être requis par la législation comme dans le cas des machines de vote électronique, ou en raison d'une indisponibilité de moyens techniques dans certains domaines comme l'authentification autonome de produits [IAFF10]. Dans ces conditions, les machines doivent être authentifiées avant chaque session de travail afin de pouvoir détecter toute altération du système, une attaque par substitution ou une corruption système (par un *cheval de Troie* par exemple). Attention, il ne s'agit pas ici de gérer l'accès à une machine électronique en authentifiant un utilisateur, mais réellement de permettre à un utilisateur humain de détecter toute modification interne de son système, par rapport à un état du système, défini a priori comme authentique. Cela suppose que la décision concernant l'intégrité de la machine doit être validée par l'utilisateur final et non par le système lui-même. Pour cela, il faut que lors de la phase de vérification, l'utilisateur soit actif et puisse avoir un comportement imprévisible.

L'intervention active d'un humain dans une authentification fait généralement usage d'un mot de passe ou d'un code PIN (*Personal Identification Number*). Il est donc important d'indiquer ici, qu'un système d'authentification par mot de passe ou par PIN n'est pas satisfaisant dans les cas qui nous intéressent, car lors de l'authentification le rôle de l'utilisateur se résume à un comportement assez prévisible, qui est la saisie d'un même PIN ou mot de passe. Selon Chabaud *et al.* [CF06], un tel processus authentification par mot de passe est un *déverrouillage* plutôt qu'une authentification.

Dans les sections ci-dessous, nous décrivons deux principes que l'on trouve dans la littérature et qui apportent une réponse partielle à cette problématique d'authentification d'une machine par un humain.

III.3.1 Jetons d'authentification

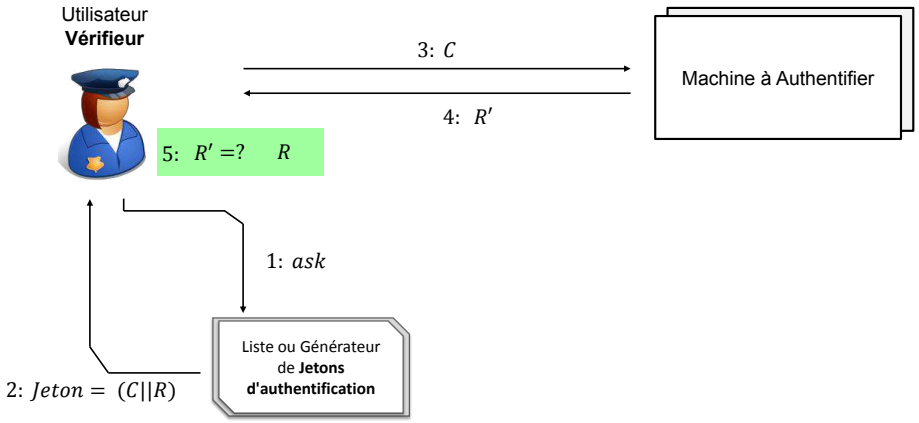

Figure III.1 *Authentification d'une machine à travers un générateur ou une liste de jetons (des couples challenges/réponses).*

— ○ —

Le principe des jetons d'authentification permet d'assurer une *authentification forte*, très souvent d'un utilisateur auprès d'une machine ou d'un système informatique. Les applications les plus courantes de ce principe se trouvent dans les protocoles de paiement et d'authentification bancaire. Ainsi plusieurs protocoles et brevets [Bur07] sont développés par les sociétés de paiement électronique et les entreprises de sécurité comme RSA Security, VeriSign etc. Dans [GKD09], le principe de jetons d'authentification apporte une première solution à une participation d'un utilisateur humain à l'authentification d'une machine de vote.

Une authentification entre un homme et une machine à travers un PIN (ou un

mot de passe) est vulnérable à une attaque par copie et par rejeu. En effet, un humain ne peut communiquer avec une machine qu'à travers une interface d'entrées/sorties. Les informations qui y circulent sont aisément copiables de l'intérieur de la machine ou même de l'extérieur car accessibles aux regards indiscrets. D'autre part, une authentification par PIN ne peut être efficace contre une machine qui simule le comportement d'une machine authentique, tel par exemple une machine, forme de « YesCard », qui répond systématiquement **OUI** quelque soit le mot saisi pour mot de passe.

Pour prévenir le risque d'une « YesCard », une première solution, est que l'utilisateur puisse prévoir de faux PIN qui seront choisis aléatoirement. Lors de la phase d'authentification ces faux PIN serviront à masquer la vrai vérification d'authenticité, et une réponse OUI à un faux PIN témoignera d'une altération de la machine. Cependant, cette solution n'est toujours pas complètement satisfaisante, car tant que la réponse d'une machine à un PIN sera OUI ou NON (ou toutes réponses préalablement fixées), et que le vrai PIN restera invariant pour plusieurs phases de vérification, un adversaire pourra aisément déduire le vrai PIN en copiant les échanges de ces différentes phases de vérification.

Pour prévenir cette copie de PIN, il faut soit en changer après chaque utilisation (c'est à dire après chaque phase de vérification), soit prévoir un grand nombre de PIN et leurs réponses correspondantes. Un PIN et sa réponse forment un couple challenge/réponse appelé **jeton d'authentification** ou jeton cryptographique. En terme d'authentification, cela implique pour l'utilisateur d'avoir une liste de jetons d'authentification, ou une méthode pour générer à la demande un jeton, plutôt qu'un code PIN ou un mot de passe. Le principe de cette authentification (Figure III.1) est qu'en se servant d'un jeton $(C\|R)$, l'utilisateur stimule la machine à s'authentifier en fournissant une réponse R' au challenge C du jeton. L'utilisateur valide l'authenticité de la machine en comparant la réponse R' à la référence R du jeton.

En pratique, comme il est évident qu'un utilisateur ne peut mémoriser qu'un petit nombre de jetons, un système embarqué (donc une machine) est mise à la disposition de l'utilisateur pour la génération ou la mémorisation des jetons d'authentification de la machine. De tels générateurs sont basés sur différents méthodes et primitives cryptographiques, en allant de l'implantation d'un simple compteur, ou d'un système de signature électronique, jusqu'à l'utilisation de PUFs (*cf.* section III.4.2.1).

Nous venons ainsi de voir en quoi, les jetons d'authentification apportent une réponse à une authentification machine - homme. Cependant, ce principe d'authentification fait intervenir, comme tiers, une machine qui génère ou qui stocke des jetons. Cela transfert donc la question de la vérification d'authenticité sur une machine à jetons. La section ci-dessous apporte une solution à cette dernière question.

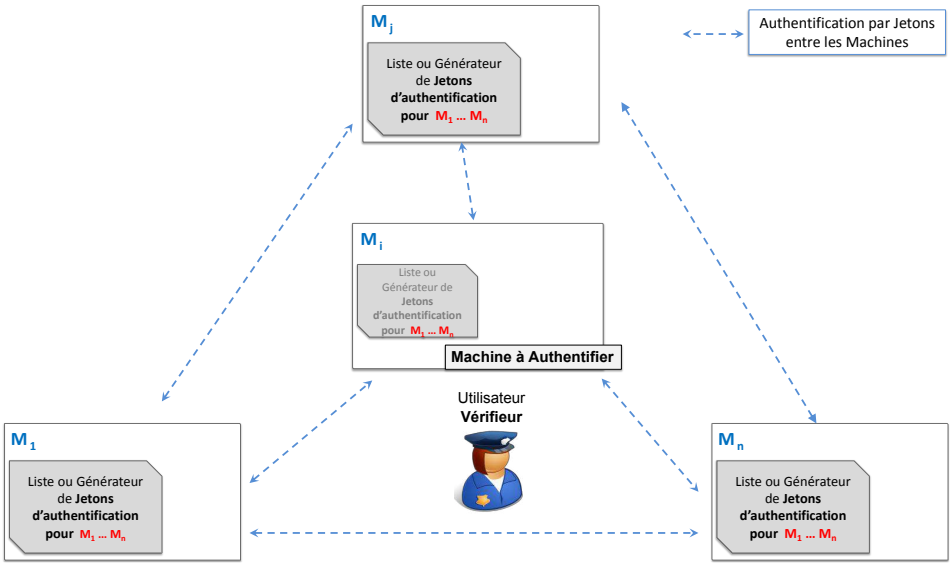

Figure III.2 *Authentification à travers un ensemble de machines : pour authentifier une machine M_i, l'utilisateur (**Vérifieur** de M_i) peut se servir d'autres machines M_j ($j \neq i$) comme fournisseur de jetons d'authentification.*

— o —

III.3.2 Ensemble de machines

L'idée est la suivante : pour diminuer le risque de corruption de la machine à jetons (de la section III.3.1 précédente), il faut donner la possibilité à plusieurs autre machines de générer des jetons d'authentification.
C'est l'idée présentée par Gallo *et al.* dans [GKD09] pour l'authentification off-line de leur machine de vote électronique. Gallo *et al.* décrivent un schéma basé sur un module matériel d'identification permettant l'authentification des machines de vote. Ainsi, chaque machine de vote a un module unique d'authentification (nommé CID : Cryptographic IDentity), qui assure donc l'identification et permet de vérifier l'intégrité de la machine. Cette vérification d'intégrité se fait à travers un groupe de machines de la même famille possédant tous des CID. Ces CID sont configurés à leur fabrication (c'est à dire à la configuration initiale des composants CID) avec les clés et les données nécessaires pour générer des jetons d'authentification, capables de vérifier tous les autres CID.

Dans ce schéma, pour authentifier une machine de vote, il faut d'autres machines de la même famille. L'utilisateur vérifieur d'une machine doit l'authentifier par toute une série de machines. Pour savoir si chacune de ces machines est authentique, il peut les authentifier à leur tour en se servant des autres. Dans ce schéma, l'utilisateur joue un rôle plus actif, car, c'est à lui de vérifier visuellement les réponses aux challenges des jetons, donc de décider si une machine est au final authentique ou pas. Les machines ne sont là que pour délivrer les jetons. En plus de répondre à la problématique d'une

authentification d'une machine par l'homme, ce schéma de Gallo *et al.* permet à un utilisateur d'authentifier plusieurs machines en une seule séance de vérification : car l'authentification d'une machine entraine et exige l'authentification de plusieurs d'entre elles.

Cet avantage est aussi l'inconvénient principal de ce schéma, car pour que l'authentification d'une machine soit acceptable, il faut en avoir plusieurs autres. De plus, une machine ne peut authentifier une autre que si elle était préalablement configurée pour le faire.

Nous consacrerons le chapitre IV à décrire nos travaux sur une intervention encore plus active de l'utilisateur humain dans le protocole d'authentification. Nous proposerons deux niveaux de vérification : dans le premier, l'utilisateur aura besoin d'un système de calcul [IAF12], comme dans les deux cas illustrés aux Figures III.1 et III.2 ci-dessus, mais avec plus de liberté pour l'utilisateur, et un second protocole où, nous proposerons un système de vérification basé sur une série de permutations choisies par l'utilisateur, mais dont il soit intrinsèquement capable d'effectuer les opérations nécessaires lors de la vérification d'authenticité.

III.4 Éléments sur les PUFs

Pour un composant, un PUF est à la fois une fonction électronique et une empreinte physique (l'"ADN" du composant). C'est à ce dernier titre que nous évoquons les PUFs sans qu'il soit question de présenter dans cette section, une technologie spécifique de PUF, ni de décrire les phénomènes électroniques sur lesquels elle peut se fonder. La littérature abonde de publications en ce sens [PRTG02, GCVDD02, GKST07, SD07, KGM+08]. Nous développerons dans cette section l'intérêt des PUFs en cryptographie moderne, en particulier leur intérêt pour l'authentification de circuits électroniques.

III.4.1 Définition d'un PUF

Un **PUF**(*Physical Unclonable Function*) définit une fonction physiquement non reproductible. Un PUF est une fonction qui dépend physiquement du circuit électronique qu'il caractérise. Il représente ainsi une empreinte physique de ce circuit. Il représente l'information stable provenant d'un phénomène physique qui est aléatoirement imprévisible et inimitable d'un circuit à l'autre : c'est l'aléa de fabrication.

Comme fonction, un PUF établit une relation entre une donnée de stimulation appelée **challenge** C et sa **Réponse** R. La paire (C, R) définit un challenge-réponse (*Challenge-Response Pair* ou CRP). Un circuit donné est caractérisé par son PUF, lui même représenté par un ensemble de paires CRP qu'il a générées. Ainsi, une technologie donnée de PUF doit satisfaire aux propriétés suivantes :

1. toute modification d'un PUF ou du composant qu'il caractérise doit avoir un impact non prévisible sur la table des CRPs qui représente la PUF,

2. pour deux challenges différents C_i, C_j, la réponse R_i du challenge C_i, ne fournit aucune information sur R_j provenant de C_j,
3. étant donnée un challenge C_i, il est improbable d'avoir la réponse R_i lui correspondant, sans posséder le circuit et sa fonction PUF qui a permis de le générer.

La première propriété définit la capacité de distinguer deux circuits à travers leurs PUFs respectifs. La validité de cette propriété garantit la non reproductibilité d'une fonction PUF et permet ainsi de considérer un PUF comme un élément de vérification de l'identité d'un circuit. La seconde et la troisième propriétés assurent la possibilité d'utiliser un PUF comme fonction à sens unique. Ces deux dernières propriétés garantissent ainsi la capacité d'utiliser un PUF comme source de clés.

Ces propriétés permettent de définir concrètement ce que l'on attend d'un PUF idéal. Cependant pour une technologie de PUF donnée, réaliser toutes ces conditions ne va pas de soi. L'écart entre une technologie et un comportement idéal de PUF se mesure par deux probabilités [SD07] exprimant respectivement la **variation inter-chip** et la **variation intra-chip**.

III.4.1.1 Variation inter-chip

Pour une technologie de PUF, la variation inter-chip est la probabilité que deux PUFs de circuits différents aient des tables de CRPs différentes. Pour une technologie de PUF idéale, l'inter-chip doit être 50%, pour valider la première propriété ci-dessus. Autrement dit, pour deux circuits A et B ayant une même technologie de PUF, si R_A et R_B représentent respectivement la réponse du PUF de A et celle du PUF de B à un même chalenge C, alors la distance de Hamming entre R_A et R_B doit être 50% c'est à dire que 50% des bits diffèrent entre R_A et R_B.

Une technologie de PUF se doit d'avoir une variation inter-chip proche de 50%. Une valeur trop faible indique une faible capacité à authentifier à partir du PUF et rend problématique la gestion de clés qui en sont issues (car cela augmente la probabilité d'avoir des clés identiques pour des circuits différents).

III.4.1.2 Variation intra-chip

Un PUF est une fonction physique, en cela il est sensible à l'environnement et aux conditions d'utilisation. Selon la technologie du PUF, les conditions d'utilisation du circuit, comme la température et la tension d'alimentation, ou encore sa vétusté peuvent induire des variations de comportement. Ces variations conduisent un PUF de circuit à donner des réponses sensiblement différentes pour un même challenge. La probabilité intra-chip caractérise le pourcentage de bits différents entre deux réponses R_1 et R_2 du PUF à un même chalenge C. La variation intra-chip idéale d'un PUF est bien sûr de 0% (une distance de Hamming de zéro).

Généralement, la variation intra-chip est indépendante de la variation inter-chip, c'est-à-dire que, quelque soit la valeur de la variation intra-chip, pour un même challenge, deux réponses du PUF d'un même circuit resteront assez semblables par rapport à toutes réponses obtenues par d'autres PUFs de circuits différents. On peut en déduire qu'une variation intra-chip plus ou moins élevée, n'a pas d'inconvénients sur l'utilisation des PUFs pour l'identification et l'authentification de circuits. Par contre, pour une utilisation cryptographique, comme la génération de clés, il est important de maîtriser les variations intra-chip et même de le corriger.

III.4.2 Utilisation de PUFs

III.4.2.1 Authentification de circuits

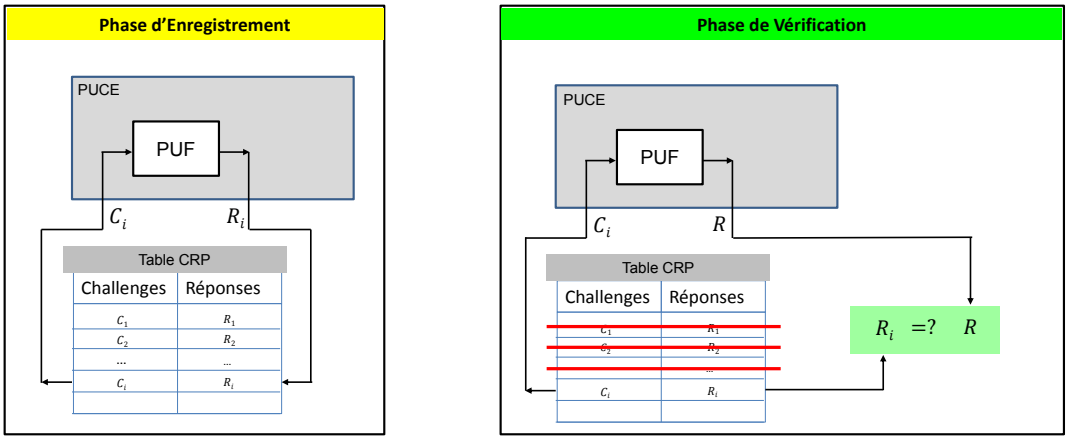

Figure III.3 *Authentification d'un composant électronique à partir d'une fonction PUF*

— o —

Au paragraphe I.2.2.5 du chapitre I nous avons évoqué comme exemple d'authentification d'objets, l'authentification de systèmes électroniques à partir de PUFs. En effet, le caractère intrinsèque qui lie un PUF au composant sur lequel il est implanté, permet de considérer un PUF comme un élément de vérification de l'authenticité d'un composant et le cas échéant du système embarqué comprenant le composant.

La phase d'enregistrement (*cf.* III.3) d'un tel processus d'authentification consiste à établir la table CRP de correspondance entre les challenges choisis et les réponses fournies par le PUF. Cette table définit ainsi l'empreinte de référence du composant. La phase de vérification s'effectue en envoyant un challenge au PUF et en comparant sa réponse à la réponse correspondant au challenge dans la table CRP. Pour un sécurisation du processus d'authentification, il est important de ne pas réutiliser un challenge. Cela nécessite de calculer à l'enregistrement, une table CRP assez importante, afin de pouvoir réaliser un nombre suffisant de challenges/réponses.

A la variation intra-chip près, la réponse du PUF doit être la même. Dans un contexte d'authentification, pour une technologie de PUF donnée, le *FRR* et le *FAR* sont définies en fonction des variations intra-chip et inter-chip. Une variation intra-chip faible induit un *FRR* proche de zéro, donc une plus grande fiabilité du processus d'authentification et la possibilité d'avoir une table CRP plus dense.

III.4.2.2 Identification de circuit

A la section III.4.3 nous discuterons du stockage de clés, ou de façon générale, du stockage d'information dans un composant électronique. Dans un système embarqué, les éléments d'identification d'un composant doivent être disponibles près ou dans ce composant. Un PUF peut tenir un tel rôle. En effet, pour l'identification d'un composant (ou d'un système embarqué comportant ce composant), un challenge et sa réponse peuvent être réservées comme élément d'identification du composant. Dans ce cadre, au lieu de stocker une donnée comme identifiant d'un composant, il suffit de désigner un challenge standard dont la réponse sera considérée comme l'identifiant du composant. Ainsi, pour créer un identifiant à un composant possédant un PUF il suffit de calculer et de stocker sa réponse à un challenge standard. Cependant, une bonne variation inter-chip est nécessaire pour assurer une identification dans une même famille de composants.

III.4.2.3 Génération de clé

Les conditions de la définition (section III.4.1) d'un PUF idéal, assurent une l'imprévisibilité du résultat fourni par la fonction PUF. De telle sorte qu'on peut affirmer qu'avec un PUF idéal, la réponse de R à un challenge C donné, peut prendre équiprobablement une valeur dans toutes les réponses possibles. Autrement dit, pour une technologie idéale de PUF, connaissant seulement un challenge C, sans avoir le PUF, déduire la réponse R de n-bits du PUF correspondant à C, équivaut à faire un choix au hasard avec une probabilité de $1/2^n$. Ainsi si les réponses d'un PUF idéal sont sur n bits, nous pouvons considérer que le PUF permet à partir d'un challenge de générer aléatoirement n bits. Chaque réponse peut donc être utilisée comme une clé.

En pratique, la robustesse d'une technologie de PUF se caractérise par sa variation inter-chip et intra-chip. Comme nous l'avons souligné, une technologie possédant une mauvaise variation inter-chip, ne peut servir à générer une clé. En effet dans ce cas, en partant d'un même challenge, on peut tirer de l'information de la réponse du PUF d'un composant, à partir de la réponse du PUF d'un autre composant de sa famille. Dans l'utilisation d'un PUF en vue de générer une clé cryptographique, on suppose qu'étant donné un composant, il est en pratique difficile de produire un composant-clone possédant un PUF capable de fournir la même table CRP avec des ressources polynomiales. Les ressources polynomiales [Gas03**?**] définissent le temps, le coup financier, les éléments matériels qu'il faudrait pour réaliser une telle copie. Ainsi avec la variation inter-chip et une estimation des ressources qui permettrait de copier

un PUF, on peut estimer les risques que l'on prend à utiliser un PUF pour la génération de clés cryptographiques.

Une variation intra-chip non nulle signifie que les réponses du PUF peuvent être bruitées. Or du fait qu'une clé cryptographique ne peut admettre une modification, ne serait-ce que d'un seul bit, en vue d'utiliser une réponse de PUF comme clé ou source de clé, on effectue une correction de la variation intra-chip [ŠTO05, SD07] via un Code Correcteur d'Erreur (Error Coding Code ou ECC) ou on exige des conditions particulières de fonctionnement : une température donnée (ce qui n'est pas simple) ou une tension d'alimentation donnée.
Pour des utilisations de PUFs autorisant une variation intra-chip non nulle, on peut se limiter à fixer des conditions optimums d'utilisation du PUF. Cela peut suffire à stabiliser la variation intra-chip afin d'avoir un résultat acceptable, comme dans le cadre d'une utilisation de PUF en tant qu'élément d'identification.

PUF et Code Correcteur d'Erreur

Pour une technologie donnée de PUF, il incombe d'avoir une bonne estimation du pourcentage du bruit maximal que peut induire la variation intra-chip sur les réponses. En général une étude du phénomène physique qui induit le PUF, associée à une étude statistique sur un échantillon de PUF peut permettre d'avoir une bonne estimation. En connaissant la probabilité intra-chip, on en déduit un nombre maximum de bits différents que peut contenir une réponse d'un PUF, par rapport à une réponse antérieure pour un même challenge. Il suffit alors d'utiliser un code correcteur d'erreur qui peut corriger le bruit d'une réponse du PUF pour retrouver la réponse attendue. Suh *et al.* [SD07] associe ainsi un code BCH au PUF pour assurer une telle correction.

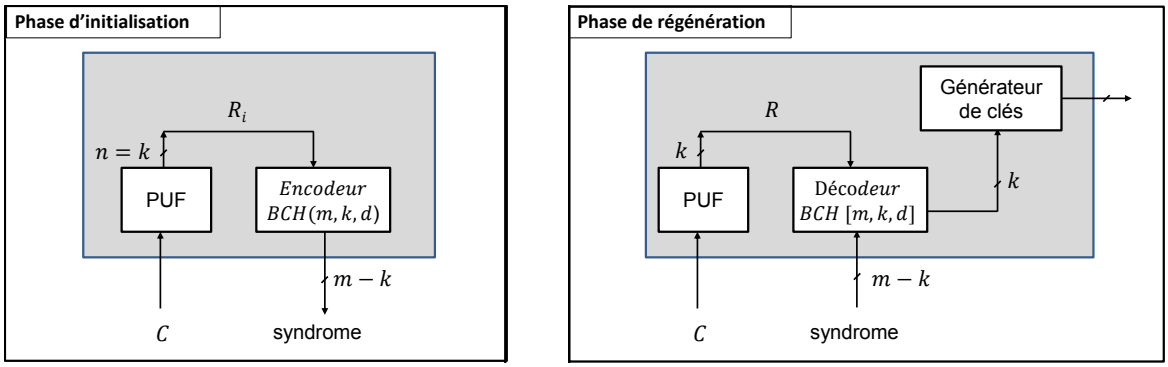

Figure III.4 *Génération de clés avec un PUF et un ECC (code correcteur d'erreur) de type* $BCH(m,k,d)$

– ○ –

Un code BCH de paramètres (m, k, d) [DRT07] est un code correcteur capable de corriger au plus $(d-1)/2$ erreurs à partir de mots de code (un message plus la redondance) de m bits contenant k bits d'information, d étant la distance de Hamming minimale entre deux mots de code. Il s'agit ainsi, pour une technologie de PUF fournissant des réponses sur n bits et dont la variation intra-chip n'induira pas plus de $(d-1)/2$ bits d'erreurs, de lui associer un ECC définit par $BCH(m, k = n, d)$.

Pour créer une clé à partir d'un challenge C, d'un PUF et d'un ECC, Suh *et al.* décrivent une méthode basée sur deux phases : une phase d'initialisation puis une phase de régénération (*cf.* Figure III.4).
Lors de la phase d'initialisation, le PUF fournit une première réponse R_i enregistrée comme réponse originelle au challenge C après encodage à l'aide de $BCH(m, k, d)$: un mot de code composé des bits de R_i et de $m - k$ **bits de redondance**. Suh *et al.* appellent ces bits de redondance le *syndrome*. Cette redondance permet de caractériser l'erreur éventuelle induite par la variation intra-chip du PUF lors d'une vérification. La phase de régénération décrit le décodage par le syndrome qui permet de corriger les réponses du PUF à partir du challenge C.

Le couple PUF+ECC permet ainsi d'obtenir une réponse toujours identique quelque soit les conditions d'utilisation du composant. Les bits d'information de cette réponse peuvent être utilisée directement comme clé cryptographique, comme entrée d'une fonction de hachage ou comme graine d'un générateur pseudo-aléatoire servant à générer une clé cryptographique.

Selon ce principe, le challenge et le syndrome sont supposés publiques (c'est à dire supposés non secrets), et à partir d'eux une clé unique est générée dans le composant. Du fait du caractère aléatoire du PUF, un challenge ne fournit aucune information sur la réponse qu'il génère, donc aucune information sur la clé produite à partir de cette réponse. Cependant, le syndrome (ou bits de redondance correspondant) représente clairement une fuite d'information sur la réponse du PUF. Suh *et al.* affirment que cette perte d'information est représentée par les $m - k$ bits du syndrome. Cependant, il faudrait estimer plus rigoureusement, l'information que fournissent ces bits de redondance sur le réponse.

Sans oublier que dans un tel cas (PUF+ECC), dans une recherche exhaustive de la réponse R sur n bits (correspondant à un challenge C et un syndrome), on peut se contenter de rechercher dans un ensemble de taille plus petite que 2^n, en utilisant la capacité du ECC à corriger jusqu'à $(d-1)/2$ bits d'erreurs : il suffit dans la recherche de R de parcourir exhaustivement $n-(d-1)/2$ bits, puis les compléter par $(d-1)/2$ bits.

III.4.3 Stockage de clés

La gestion de la sécurité dans un système embarqué induit la nécessité de pouvoir y stocker, de façon sûre, notamment confidentielle, et fiable des données comme les clés et éléments d'identification utiles. Comme nous l'avons vu pour la confidentialité du contenu de configuration (section III.2.1), une clé secrète K_{ENC} doit être nécessairement disponible pour déchiffrer le contenu de la NVM avant chaque configuration. De même, l'authentification du système auprès d'un tiers ou d'un utilisateur, et l'authentification interne du contenu de configuration du système dépendent d'informations secrètes stockées en interne dans le système. La protection de la propriété intellectuelle du système et à la protection des données de l'utilisateur final supposent la possibilité que le système électronique dispose d'informations stockées de façon sure, confidentielle au sein du système embarqué, à l'intérieur ou à proximité du composant.

On distingue alors principalement deux types de stockage de clés ou d'informations importantes dans un circuit électronique : le stockage **volatile** et le stockage **non-volatile**.

En mode volatile, les clés sont stockées dans une mémoire de type SRAM à faible consommation, alimentée par une batterie extérieure. Ce mode de stockage efface les clés dès que la SRAM n'est plus alimentée, cela procure un grand niveau de sécurité des périphériques contre les attaques invasives et semi-invasives (des attaques physiques destructrices du matériel). Le problème avec un tel choix, est que la disponibilité et la vie des clés dépendent de la durée de vie de la batterie car par principe, les clés de la SRAM seront perdues dès que la batterie sera vide ou débranchée. La vie d'un tel composant (et du système embarqué qui en dépend) est donc limitée à celle de la batterie.

Pour conserver les clés de façon durable, on utilisera une mémoire non-volatile telle qu'une mémoire PROM, EPROM, EEPROM ou FLASH. Il faut donc inclure une mémoire non volatile dans le composant. Cependant, selon S. Drimer [Dri09], l'intégration d'une mémoire non-volatile dans un circuit FPGA est un procédé nouveau qui induit une nouvelle étape dans la réalisation, ayant un impact sur le prix, le rendement et même la fiabilité du circuit.

Une idée de stockage ou de génération locale de clés serait d'implanter un PUF dans le composant en lieu et place de la mémoire de stockage de clé. Cette idée décrite dans [GGM85**?** , ŠTO05, SD07] constitue l'objectif moteur, qui à permis le développement des technologies de PUF. Au paragraphe III.4.2.3 nous avons montré comment il est possible de générer une clé cryptographique à partir d'une fonction PUF combiné à un code correcteur d'erreur et d'une fonction hachage comme décrit dans [SD07]. Le challenge et le syndrome peuvent être considérés comme des données publiques, et dans ce cas, être aussi stockés dans une mémoire non volatile, sans exigence de confidentialité.

III.5 Conclusion

Dans ce chapitre, nous nous sommes restreints aux systèmes électroniques embarqués dont l'authentification résulte de celle d'un composant configurable à partir d'une mémoire non-volatile (NVM).

L'authentification d'un tel système dépend principalement de trois acteurs qui sont : *le composant d'authentification*, le *contenu de la NVM* servant à la configuration ou à la programmation du composant, et enfin *l'utilisateur final*. Cela nous a conduit à décrire deux niveaux d'authentification : d'une part l'authentification du contenu de la NVM par le composant et d'autre part l'authentification externe du composant et de son contenu de configuration par l'utilisateur final.

Concernant la première authentification, celle du contenu de la NVM par le composant, les protocoles d'aujourd'hui se contentent d'un calcul de MAC soit seulement du contenu de configuration, soit du contenu de configuration et d'une identification de ce contenu. Seul un protocole qui considère une telle identification, permet de prendre en compte une gestion des mises à jour. Il ne suffit donc pas de calculer une empreinte MAC du seul contenu de la NVM pour assurer une authentification fiable du contenu d'un composant configurable. Il est indispensable de prévoir et de mettre également à jour un identifiant du contenu de la NVM (un compteur ou un numéro de version) à l'intérieur du composant. Nous avons montré que les protocoles assurant une authentification interne du contenu d'un composant par lui même, ne permettent pas de réellement authentifier le composant, a fortiori le système embarqué qui l'intègre, car un tel protocole ne résiste pas à une **substitution du composant d'authentification** du système embarqué.

C'est pour cela que le contexte d'authentification assurant une authentification externe d'un composant d'un système embarqué par un utilisateur final (homme ou machine) est nécessaire. Les cas où l'utilisateur final est une machine ne nous concerne pas ici. Les protocoles décrits dans la littérature [Dri07, BET08] ne réservent à un utilisateur humain que la possibilité de réaliser une authentification à distance, en déléguant l'authentification à un tiers de confiance distant. Ainsi, dans les conditions d'une authentification off-line, aucun protocole ne permet à un utilisateur humain de vérifier réellement l'authenticité d'un composant et de son contenu, et encore moins celle d'un système embarqué basé sur ce composant.

Nous avons présenté à la section III.3 de ce chapitre, le principe des schémas d'authentification de machines par un humain. Nous avons ainsi décrit le principe d'authentification basé sur une machine génératrice de jetons d'authentification, palliant la mémoire humaine, pour que l'utilisateur puisse avoir un choix important de jetons différents à chaque phase de vérification d'authenticité. Cette solution pose le problème de l'authentification de la machine génératrice de jetons par l'utilisateur. Pour cette authentification, nous avons exposé un principe de [GKD09] dont l'idée est

que l'utilisateur parte d'un ensemble de machines, dont chacune peut générer des jetons pour authentifier l'autre. Ce schéma présente cependant des défauts : l'utilisateur n'est pas libre dans le choix de machines tiers, les machines tiers utilisées sont nécessairement de la même famille et configurées ensemble, afin d'authentifier une machine de son choix.

C'est dans ce cadre qu'intervient la question d'une **d'authentification autonome** (ou encore, partiellement autonome) entre un vérifieur humain et le composant d'authentification du système embarqué. Ce cas est traité dans le chapitre suivant, en apportant plus de liberté à l'utilisateur dans le choix d'une éventuelle machine tiers. L'idée est de donner à l'utilisateur la liberté de choisir n'importe quel outil de calculs, pour pallier ses faibles capacités de mémorisation et de calcul.
Nous proposons aussi un protocole d'authentification que nous avons bâti sur les capacités de calcul et de mémorisation que peut avoir un humain, ou qu'un humain peut développer après une phase d'entraînement.

Chapitre IV

Authentification autonome d'un système embarqué par un Humain

IV.1 Introduction : vérification via le composant d'authentification

Nous considérons le modèle de système embarqué introduit au chapitre précédent, dont l'authentification passe par celle d'un composant dit d'authentification au fonctionnement lié à la configuration ou à la programmation d'une mémoire RAM intrinsèque (un FPGA ou un microcontrôleur dans notre étude). Cette authentification repose sur l'hypothèse d'une clé secrète unique $Kmac$ à partir de laquelle le composant d'authentification calculera des MACs qui constitueront les empreintes du système à vérifier. La clé $Kmac$ est ainsi supposée secrète pour toute entité autre que le composant d'authentification.

Comme tout processus d'authentification (chapitre I, section I.2.1), celui que nous proposons ici se décompose en une phase d'enregistrement (ou d'initialisation) consistant à générer et stocker l'empreinte digitale caractérisant l'état du système électronique et une seconde phase, de vérification (ou d'audit) répétable à souhait, lors de laquelle une nouvelle valeur de l'empreinte est générée pour être comparée à l'empreinte stockée (en référence).

IV.2 Notre principe d'authentification par un Humain

IV.2.1 Phase d'enregistrement

Le principe (Figure IV.1) de notre méthode d'authentification est qu'à la phase d'initialisation, l'utilisateur partage une information secrète S avec le composant d'authentification du système embarqué, créant ainsi un lien entre le composant et

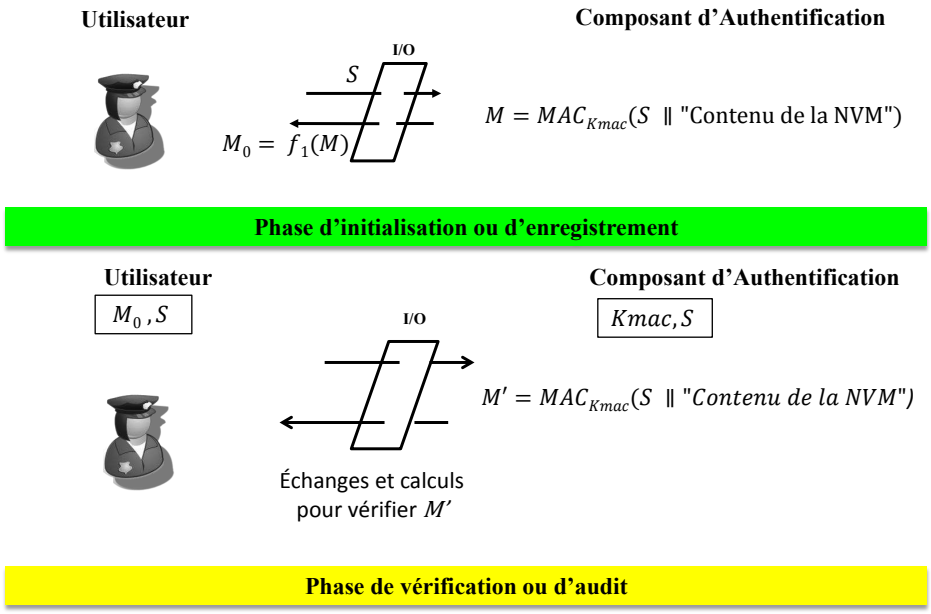

Figure IV.1 *Principe de l'authentification d'un système embarqué par un humain : La f_1 représente une fonction générique, nous la définissons en fonction des différents hypothèses sur les moyens de l'utilisateur à la phase de vérification.*

Les données publiques : $M_0 = M$.

Les données secrètes : $S, Kmac$.

Les données encadrées, représentent les connaissances de l'utilisateur ou du composant au début de la phase de vérification.

— ○ —

l'utilisateur.

Pour cela, il est important que cette phase d'initialisation se déroule dans un local sûr afin que le processus ne révèle aucune information sur le secret S de l'utilisateur. Cela signifie une exécution de la phase d'enregistrement dans un local protégé contre toute forme d'attaques, passives et actives, de telle sorte que seuls l'utilisateur et le composant d'authentification sont supposés connaître S. D'autre part, la génération et la mémorisation du secret S ne doit laisser aucune trace exploitable par un adversaire. Lors de la phase d'enregistrement on suppose que le système embarqué est authentique (au moins lors d'un tout premier enregistrement) donc fiable et intègre. Le secret S n'est enregistré nulle part dans le système embarqué autre que dans la mémoire sécurisée qui lui est réservée dans le composant d'authentification. A l'usage, pour certifier la fiabilité et l'intégrité d'un système embarqué, l'utilisateur peut effectuer au préalable

une authentification en ligne ou faire une mise à jour du système, qui effectuera une certification auprès d'une autorité de certification distante (nous avons vu que différents protocoles sont proposés dans la littérature pour assurer une authentification ou une mise à jour en ligne d'une composant configurable, comme celui décrit dans [? BET08]).

Ainsi, le calcul de l'empreinte MAC de référence par le composant d'authentification doit inclure, en plus du contenu de configuration du composant, le secret S de l'utilisateur. Les calculs internes du système embarqué doivent être faits par le composant d'authentification, car lui seul connaît la clé secrète $Kmac$ et les informations sensibles comme le secret S. Ici, l'empreinte authentique M est un MAC du secret S et du contenu original de la NVM, généré à partir de la clé $Kmac$.

Cette phase d'enregistrement se termine par le renvoi depuis composant à destination de l'utilisateur de la donnée M_0 (avec $M_0 = f_1(M)$) qui définit le témoin de l'empreinte de référence M calculée par le composant. L'utilisateur qui se charge de stocker M_0 doit se prémunir de toute modification de sa valeur car elle constitue le témoin de l'empreinte de référence alors qu'elle représente à la fois, la clé secrète $Kmac$ du composant, le contenu de configuration du composant et la secret S de l'utilisateur. Cependant il n'est pas nécessaire d'imposer le secret sur l'empreinte M_0. La fonction f_1 représente ici, la relation qui existe entre le MAC M calculé par le composant et l'empreinte témoin M_0 en possession de l'utilisateur de M. Les opérations à réaliser par l'utilisateur lors de la phase de vérification (de l'authenticité du système) dépendront de la spécification et de la complexité de la fonction f_1.

IV.2.2 Phase de vérification

Pour effectivement conférer à l'humain une part plus active dans l'authentification du système embarqué, nous établissons un lien qui le relie au système. Ce lien est établit en partageant un secret S (Figure IV.1). La vérification de l'authenticité dépend donc de la capacité de l'humain à vérifier le secret sans pour autant divulguer sa valeur. Or une vérification peut se dérouler dans un environnement hostile où par exemple, une partie du circuit a pu être corrompue afin d'enregistrer les entrées/sorties.

Pour une vérification, l'utilisateur à la connaissance du témoin M_0 et du secret S. Le processus de vérification permettra à l'utilisateur de vérifier à partir de ses connaissances, l'authenticité du composant d'authentification, en vérifiant simultanément, le contenu de configuration du composant (provenant de la NVM) et la présence de $Kmac$ et S au niveau du composant d'authentification. Plus précisément, après stimulation du système, le composant d'authentification calcule une nouvelle empreinte M', puis un protocole interactif permet à l'utilisateur de vérifier le nouveau MAC calculé par le composant. Ce protocole implique des échanges entre l'utilisateur et le système embarqué ainsi que des calculs pour vérifier M'.

Le protocole de vérification de M' va dépendre de la nature de la fonction f_1.

Nous définirons f_1 en fonction des hypothèses sur les moyens de calcul et de mémorisation dont un utilisateur vérifieur disposerait lors d'une phase de vérification. Ainsi, nous proposons deux protocoles différents pour authentifier de façon autonome un système embarqué, que nous comparerons aux méthodes présentées à la section III.3 du chapitre III.

IV.3 Authentification autonome avec utilisation d'un outil de calculs externe

Après avoir posé notre principe d'authentification d'un système embarqué par un humain, il nous faut proposer les fonctions et le protocole. Côté utilisateur, la principale difficulté que nous devrons surmonter est l'incapacité d'un humain à effectuer des calculs cryptographiques dans un délai raisonnable. En effet, si les calculs de hachage et de chiffrement moderne sont à la portée des machines (disposant des clés), ils ne sont pas vraiment réalisables par un humain en un temps court. Aussi considérons qu'à chaque phase de vérification, l'utilisateur dispose d'un outil de calcul. Mais contrairement aux machines génératrices de jetons que nous avons évoquées au chapitre III, cet outil de calcul sera indépendant du système embarqué à authentifier : il ne doit y avoir aucune interaction entre l'outil de calcul et le système à authentifier.

Nous proposons deux protocoles de mise en œuvre de ce principe. Un premier protocole consiste à utiliser des fonctions cryptographiques symétriques, qui sont généralement implantées dans les architectures matérielles pour assurer l'authentification du contenu de configuration - comme dans Actel Fusion [Act10], Virtex6 [Xil10]. Cela signifie que la vérification de l'utilisateur sera basée sur un calcul externe de MAC en s'aidant d'un outil implantant le même algorithme de calcul de MAC que celui implanté dans le système à authentifier. Le second protocole est une adaptation du principe d'identification de Fiat-Shamir [FS87] selon un protocole de Zero-Knowledge (protocole à divulgation nulle de connaissances) entre le système et l'utilisateur humain. Dans ce dernier cas l'outil de calcul de l'utilisateur pourrait être une calculatrice capable d'effectuer des opérations d'arithmétique modulaire.

IV.3.1 Authentification par calcul externe de MAC

IV.3.1.1 Phase d'enregistrement

Par rapport à la phase d'enregistrement IV.2.1 (*cf.* Figure IV.1) décrite à la section précédente, nous supprimons le bloc fonctionnel correspondant à la fonction f_1 en considérant f_1 comme la fonction identité : le système électronique transmet directement à l'utilisateur le MAC original comme témoin.

$$f_1(M) = M = M_0. \tag{IV.1}$$

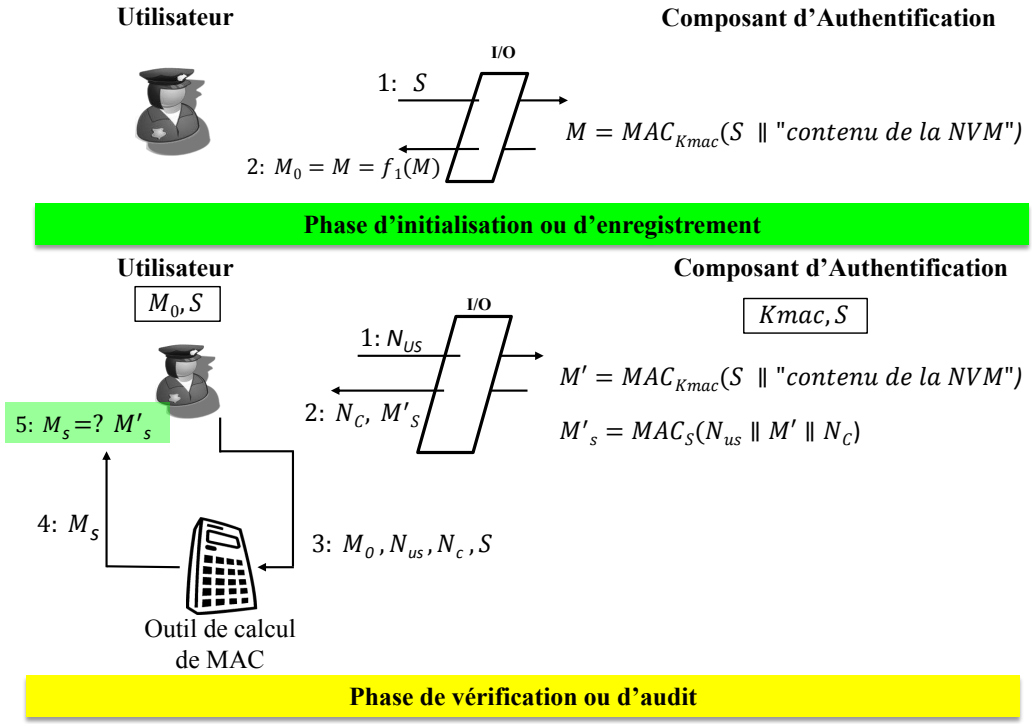

Figure IV.2 *Protocole d'authentification autonome avec un outil de calcul externe de MAC*

Les données publiques : $M_0 = M$.

Les données secrètes : $S, Kmac$.

Les données encadrées, représentent les connaissances de l'utilisateur ou du composant au début de la phase de vérification.

— ○ —

Sous l'hypothèse que le système est authentique, cette phase d'enregistrement se déroule alors comme suit (Figure : IV.2) :

1. *Utilisateur :* l'utilisateur sélectionne au hasard un secret S, le secret S doit seulement avoir la même longueur en bits que la clé $Kmac$ de calcul de MAC. S est transmis au composant d'authentification du système, à travers une interface réservée à cette authentification.

2. *Composant d'authentification* : dès réception du secret S de l'utilisateur, le composant d'authentification stocke S dans sa zone mémoire sécurisée, il lit le contenu de la NVM et calcule un code MAC (équation IV.2) du contenu de la NVM et du secret S à l'aide de la clé $Kmac$. Le système retourne la valeur calculée de MAC à l'utilisateur.

$$M_0 = MAC_{Kmac}(S \| \text{"contenu de la NVM"}) \qquad \text{(IV.2)}$$

IV.3.1.2 Phase de vérification

A ce stade, l'utilisateur et le système peuvent se trouver dans n'importe quel environnement. Nous supposons que l'utilisateur connaît le secret S et possède l'empreinte authentique M_0 (équation IV.2) calculée lors de phase d'enregistrement. Il s'agit alors pour l'utilisateur de vérifier simultanément, les données secrètes connues par la composant d'authentification du système et d'authentifier le contenu de configuration de la RAM de ce composant. Ici, nous négligeons le risque de déni de service (ou de faux rejet), c'est à dire une modification délibérée ou accidentelle des données stockées dans le système. C'est à dire qu'un rejet de l'authenticité du système ne permettra pas à l'utilisateur de savoir la cause du rejet.

Nous faisons l'hypothèse que l'utilisateur dispose d'un outil de calculs indépendant du système, pour réaliser des calculs de MAC. Cet outil de calculs peut être toute entité capable de calculer une empreinte MAC dans un délai raisonnable, par exemple un ordinateur personnel, un *smartphone* (téléphone intelligent) ou une calculatrice prévue à cet effet. Il faut seulement que, la fonction de calcul de MAC de cet outil, soit la même que celle mise en œuvre dans le composant d'authentification.

Avec cette outil de calcul, la vérification par l'utilisateur se déroule comme décrit à la Figure IV.2.

1. *Utilisateur :* l'utilisateur démarre le processus de vérification en choisissant un nonce aléatoire N_{us}, puis l'envoie au composant d'authentification à travers l'interface d'entrée/sortie du système embarqué.

2. *Composant d'authentification :* à la réception du nonce N_{us}, le composant initie la lecture de la NVM et calcule une nouvelle empreinte MAC : $M^{'}$.
   Ensuite, le composant génère aléatoirement un nouveau nonce N_c puis calcule une empreinte MAC $M^{'}_S$ des nonces et du MAC $M^{'}$ utilisant cette fois-ci le secret S comme clé pour calculer le MAC. L'échange entre le composant et l'utilisateur se termine par l'envoi du nonce N_c et du code $M^{'}_S$ à l'utilisateur.

3. *Utilisateur :* l'utilisateur vérifie alors l'empreinte $M^{'}_S$, à l'aide de son outil de calcul de MAC. Pour cela, il introduit dans l'outil : la valeur MAC M_0, les deux nonces N_{us} et N_c puis son secret S qui sert ici de clé MAC.

4. *Outil de calcul de MAC* : cet outil calcule, puis renvoie un MAC M_S (équation IV.3).

$$M_S = MAC_S(N_{us} \| M_0 \| N_c) \tag{IV.3}$$

5. *Utilisateur :* pour finir, l'utilisateur valide l'authentification en comparant les MAC M_S et $M^{'}_S$.

IV.3.1.3 Analyse de la sécurité

Dans le protocole que nous venons de décrire, comme dans ceux qui sont proposés dans [? BET08, Act10, Xil10] concernant l'authentification des puces configurables, le processus d'authentification du composant repose sur la fiabilité et la sécurité d'une zone appelée **contrôleur de configuration**. Nous supposons donc que le composant est sûre et calcule ne valeur correcte du MAC à chaque configuration de la RAM. En effet, il est important qu'à la vérification, le MAC $M^{'}$ corresponde à la valeur calculée après lecture de la NVM, et non à une valeur pré-enregistrée. C'est en ce sens là que nous supposons fiable le contrôleur de configuration du composant.

En dehors de la valeur du MAC $M^{'}$, tous les messages échangés entre le composant et l'utilisateur sont paramétrés par des nonces aléatoires, qui sont forcement générés à chaque session de vérification. Ainsi comme les nonces N_{us} et N_c sont choisis aléatoirement, seule la connaissance du secret S permettra au composant de calculer un MAC $M_S^{'}$ valide.

Cependant, la calcul de M_S pour vérifier $M_S^{'}$, à partir de l'outil de calcul externe de MAC, conduit à la divulgation du secret S à cet outil. Cela nous ramène, à nous poser la question de la fiabilité et de l'intégrité de l'outil de calcul. Ici, cette question se traduit en deux points : la fonction MAC de l'outil est-elle la même que celle mise en œuvre dans le composant d'authentification et, est-ce que l'outil ne divulguerait pas des informations sur le secret S ? La réponse au premier point de cette question est la faculté que nous donnons dans ce protocole, à un utilisateur de changer facilement d'outil de calcul de MAC : tout logiciel de calcul de MAC peut être utiliser comme outil de calcul dans ce protocole. Par contre pour le second point, seule la possibilité de changer de secret S en effectuant une phase d'enregistrement, permet à un utilisateur de se prévenir contre la divulgation du secret par un quelconque outil de calcul de MAC.

Afin d'éviter à l'utilisateur de partager son secret avec un périphérique externe, nous proposons dans la section suivante un protocole sans divulgation de connaissance (*Zero-Knowledge*). Il s'agit d'adapter notre principe au protocole d'authentification Zero-Knowledge de Fiat-Shamir [FS87].

IV.3.2 Authentification par protocole à divulgation nulle

La faiblesse majeure du protocole que nous avons décrit à la section IV.3.1 précédente, est qu'à la vérification du témoin d'authenticité, l'utilisateur doit fournir son secret S à l'outil de calcul de MAC.

Une solution à cette faiblesse consiste à définir, toujours selon notre principe d'authentification (section IV.2), une phase de vérification à divulgation nulle entre l'utilisateur et le composant d'authentification. L'idée que nous développons ici est de

définir la fonction f_1 de calcul de témoin (*cf.* Figure IV.1) comme une fonction à sens unique dont la vérification peut se faire à travers une preuve à divulgation nulle.

Comme fonction à sens unique répondant à notre principe, nous avons l'élévation au **carré modulaire** (ou la racine carrée modulaire), la **factorisation** et aussi l'**exponentiation modulaire**. Des schémas de protocole zero-knowledge sont basés sur ces fonctions à sens unique comme le schéma de Fiat-Shamir (basé sur la racine carrée), le schéma de de Guillou-Quisquater [GQ88] (basé sur la factorisation de module RSA), le schéma de Schnorr [Sch91] (basé sur la multiplication modulaire).
Dans cette section, nous présentons un protocole où la fonction à sens unique est basée sur le carré modulaire [IAF12](protocole d'authentification de Fiat-Shamir).

IV.3.2.1 Le protocole d'authentification de Fiat-Shamir

Le protocole d'authentification et de signature d'A. Fiat et A. Shamir décrit dans [FS87] est basé sur le *problème difficile* de calcul de racines carrées modulaires lequel est aussi difficile que le *problème de la factorisation de grands entiers*) : les problèmes sont polynomialement équivalents. Ainsi étant donné un entier n (produit de nombre premiers bien choisis) il est, sans connaître sa factorisation, difficile de calculer la racine carrée modulo n d'un nombre entier quelconque. ([SV96, Dif88]).

Nous présentons une simplification du protocole d'authentification de Fiat-Shamir. Il s'agit de permettre à un **vérifieur** V de s'assurer de l'authenticité d'un **prouveur** P, sans que ce prouveur ait à partager une information secrète avec le vérifieur.

Les paramètres du protocole :
- Un **module public** n, un grand nombre entier défini par $n = p \cdot q$, où p et q sont deux nombres premiers. Dans ce protocole ni le prouveur, ni le vérifieur ne sont supposés connaître la factorisation de n.
- Deux entiers positifs publiques k et t.
- Une suite d'entiers représentant le **secret** à vérifier $(s_1, \cdots, s_k)$, tels que :

$$\forall \ \ 1 \leq i \leq k, s_i < n, \text{ et } pgcd(s_i, n) = 1.$$

- **Données publiques** $(y_1, \cdots, y_k)$ telles que

$$y_i = s_i^2 \pmod{n}, \forall \ \ 1 \leq i \leq k. \tag{IV.4}$$

Les hypothèses du protocole :
- Le module n, et la suite $(y_1, \cdots, y_k)$ sont supposés non secrets et connus de toutes les entités.
- Seul le **prouveur** P est censé connaître la suite secrète $(s_1, \cdots, s_k)$.

Le protocole : Sous les hypothèses décrites précédemment, il s'agit pour une entité qui

joue le rôle du prouveur P, de démontrer à une entité vérifieur V sa connaissance des données secrètes $(s_1, \cdots, s_k)$. Le protocole consiste à rejouer un nombre suffisant de fois les quatre étapes suivantes :

1. P génère un nombre aléatoire r, tel que $r \in [\![0, n[\![$, et envoie à V le carré c de r.

$$c = r^2 \pmod{n} \tag{IV.5}$$

2. Le vérifieur V choisit aléatoirement un vecteur binaire B, qu'il envoie à P

$$B = (b_1, \cdots, b_k) \in \{0,1\}^k. \tag{IV.6}$$

3. Dès réception du vecteur binaire $B = (b_1, \cdots, b_k)$, P calcule u, le produit du nombre aléatoire r et de l'inverse du produit des secrets s_i dont l'indice i correspond à un $b_i \neq 0$ (c'est à dire que s_i fait partie du produit si $b_i = 1$). Après ce calcul, P envoie u à V.

$$u = r \cdot \left(\prod_{i=1}^{k} s_i^{b_i} \right)^{-1} \pmod{n}. \tag{IV.7}$$

4. Enfin, V vérifie que c et c' sont égaux :

$$c' = u^2 \cdot \prod_{i=1}^{k} y_i^{b_i} \pmod{n}. \tag{IV.8}$$

Ces quatre étapes du protocole sont répétées avec différentes valeurs de r et B, jusqu'à atteindre le nombre maximum d'échanges prévus, qui est caractérisé par l'entier t.

Le lemme 1 ci-dessous nous montre qu'à travers ce protocole l'entité P peut montrer à V qu'elle connait les s_i sans pour autant que V n'obtienne d'information sur ces secrets, et V peut accepter cette preuve avec une probabilité de se tromper inférieure à $2^{-k.t}$.

Lemme 1 ([FS87])

1. *Si le prouveur P et le vérifieur V exécutent le protocole honnêtement alors l'entité V accepte la preuve que P connait le secret.*
2. *Si P ne connait pas les secrets s_i et ne peut calculer en un temps polynomial la racine carrée des produits $\left(\prod_{i=1}^{k} s_i^{b_i}\right)^{-1} \pmod{n}$, alors en suivant le protocole, l'entité V ne peut accepter la preuve de P qu'avec une probabilité de $2^{-k.t}$.*
3. *Le protocole ainsi défini est un protocole zero-knowledge.*

IV.3.2.2 Authentification à divulgation nulle d'une machine par un humain

Nous proposons dans cette section une adaptation du protocole Fiat-Shamir à notre principe d'authentification d'un système embarqué par un humain.

Pour cela, en plus de la fonction de calcul de MAC, il faudrait que le contrôleur de configuration du composant d'authentification intègre toutes les fonctions arithmétiques nécessaires à la réalisation du protocole de Fiat-Shamir. Le module n et le paramètre k doivent être prédéfinis, connus de l'utilisateur, et stockés en toute sûreté dans la mémoire sécurisée du contrôleur de configuration du composant. Ici, nous supposons que ni l'utilisateur, ni le composant d'authentification ne connaissent la factorisation du module n.

Phase d'enregistrement
L'idée est de définir des valeurs secrètes $(s_1, \cdots, s_k)$ à partir du contenu de la NVM, de la clé $Kmac$ et du secret S de l'utilisateur puis de considérer les $(y_1, \cdots, y_k)$ comme les empreintes de références. Nous définissons la phase d'initialisation comme suit :

1. *Utilisateur :* comme à la section IV.3.1 (avec le calcul externe de MAC), pour chaque processus d'enregistrement, l'utilisateur initie le processus en choisissant un secret aléatoire S. Ici, le secret S peut être n'importe quelle chaine de bits. L'utilisateur envoie S au composant.
2. *Composant d'authentification :* après avoir reçu le secret de l'utilisateur S, le composant d'authentification lit le contenu de la NVM, puis calcule pour tous les i de 1 à k, s_i et y_i comme suit :

$$s_i = MAC_{Kmac}(S \| \text{"contenu de la NVM"} \| i) < n \quad \text{(IV.9)}$$
$$y_i = s_i^2 \pmod{n}. \quad \text{(IV.10)}$$

   Si $pgcd(s_i, n) \neq 1$ [1] pour une valeur i alors le processus d'enregistrement est relancé : le composant demande à l'utilisateur une nouvelle valeur du secret S.
   Si $pgcd(s_i, n) = 1$, la puce enregistre le secret S dans sa mémoire sécurisée, puis retourne les données publiques $(y_1, \cdots, y_k)$ à l'utilisateur. La suite des (y_i) représente donc l'empreinte numérique originale du circuit.

Dans ce protocole le secret S, le module n et et la suite $(y_1, \cdots, y_k)$ constituent les informations qui permettront à l'utilisateur d'effectuer une phase de vérification. Comme toujours, l'utilisateur doit donc se prévenir de toute modification de ces données, pour réduire les fausses authentifications. Il est également important que la valeur de S reste secrète jusqu'à une nouvelle phase d'initialisation, c'est à dire jusqu'au choix d'un nouveau S. Ainsi, à travers les y_i nous avons un lien entre le secret S et la clé $Kmac$, et surtout entre le contenu de configuration de la puce et S.

1. Un cas où $pgcd(s_i, n) \neq 1$, ne doit en principe jamais se produire, car cela impliquerait qu'on ait trouvé une factorisation du module n

Remarque : Par analogie au schéma de notre principe (*cf.* Figure IV.1), la fonction f_1 est définie par les équations IV.9 et IV.10.

Phase de vérification
L'objectif de l'utilisateur est de vérifier si l'empreinte MAC du contenu présent dans la NVM (qui servira à configurer le composant d'authentification du système à authentifier), correspond au MAC calculé lors de la phase d'initialisation, qui a fourni les $(y_1, \cdots, y_k)$.

Ici, le **prouveur P est le composant d'authentification** (le contrôleur de la configuration de ce composant) et **le vérifieur V est l'utilisateur humain**. Cependant nous faisons intervenir un **outil de calcul**, pour épauler l'humain en effectuant pour son compte les opérations arithmétiques que doit réaliser le vérifieur de Fiat-Shamir. Le processus de vérification se déroule comme suit, à la mise sous tension lors de la configuration du composant d'authentification :

1. *Régénération des s_i par le composant d'authentification :* à la mise sous tension, le composant lit le contenu de la NVM et calcule les secrets $(s_1, \cdots, s_k)$ comme spécifié par l'équation IV.9, en utilisant sa clé $Kmac$ et le secret S de l'utilisateur qu'il a dans sa mémoire sécurisée.
2. P*rotocole interactif :* les échanges suivants représentent le protocole de vérification des s_i que vient de calculer le composant. Ces échanges sont ceux du protocole zero-knowledge de Fiat-Shamir que nous avons présentées à la section IV.3.2.1, où interviennent nos acteurs.
   a) *Composant d'authentification :* le contrôleur de configuration du composant génère un nombre aléatoire r avec $r < n$, puis calcule c le carré de r ($c = r^2 \pmod n$) qu'il envoie à l'utilisateur.
   b) *Utilisateur :* l'utilisateur choisit et envoie au composant un vecteur binaire B de longueur k ($B = (b_1, \cdots, b_k) \in \{0,1\}^k$).
   c) *Composant d'authentification :* avec la suite binaire B et les nouveaux s_i, le composant calcule u, puis l'envoie ($u = r \cdot \left(\prod_{i=1}^{k} s_i^{b_i}\right)^{-1} \pmod n$).
   d) *Utilisateur :* à la récupération de u, l'utilisateur introduit u dans son outil de calcul avec la vecteur binaire B et les paramètres de vérification que sont le module n et les y_i.
   e) *Outil de calculs :* calcule et affiche à l'utilisateur le produit c' :

$$c' = u^2 \cdot y_1^{b_1} \cdot y_2^{b_2} \cdots y_k^{b_k} \pmod n.$$

   f) *Utilisateur :* l'utilisateur constate alors si oui ou non, c et c' sont égaux.

   Les étapes 2a à 2f sont ainsi itérées un certain nombre de fois, ce nombre est défini par le paramètre t du schéma de Fiat-Shamir. Donc que la probabilité que l'utilisateur accepte l'authenticité d'une faux système est de $2^{-k.t}$.

Ainsi, l'utilisateur a besoin d'un outil de calculs externe pour effectuer les multiplications modulaires. Dans ce protocole, l'outil de calcul peut être n'importe quelle calculatrice qui peut effectuer une multiplication modulo n. A l'inverse du protocole reposant sur le calcul de MACs par l'utilisateur, ici aucune information secrète ne sera partagée avec l'outil de calcul. Contrairement au protocole proposé par Gallo *et al.* [GKD09], basé sur une machine génératrice de jetons d'authentification, l'utilisateur garde ici la liberté de se servir de n'importe quel outil de calcul(comme une simple calculatrice, ordinateur, smartphone, etc ...) permettant d'effectuer une multiplication modulaire. Dans la prochaine section nous proposons une nouvelle démarche qui permet à un utilisateur humain de vérifier l'authenticité d'un système embarqué sans l'aide d'un outil de calcul.

IV.4 Authentification autonome sans outil de calculs externe

En vertu du principe introduit en début de chapitre, l'utilisateur construit puis partage un secret avec le système. A partir de ce secret, une empreinte de référence des données d'authentification du système (obtenue à partir de l'unique clé $Kmac$ du composant d'authentification et du contenu de configuration du composant) est calculée par le système puis transmis à l'utilisateur à travers une fonction f_1 (*cf.* Figure IV.1). De ce principe nous avons tiré deux protocoles dans lesquels l'utilisateur a besoin d'un outil de calculs externe pour effectuer une vérification d'authenticité du système. Dans cette section, **notre objectif est de nous affranchir de la machine auxiliaire. Cela induit que toutes les opérations de vérification soient réalisables par un utilisateur humain**, en toute autonomie, sans pour autant introduire une quelconque faiblesse dans le processus d'authentification.

La solution que nous proposons ici (*cf.* Figure IV.3), utilise la fonction identité comme fonction f_1, ce qui veut dire que l'empreinte de référence du système qu'a l'utilisateur est le MAC $M = MAC_{Kmac}(secret \| \text{"}contenu\ de\ la\ NVM\text{"})$. Les fonctions f_2, f_3 de la phase de vérification sont construites à partir d'opérations **Xor bit-à-bit** et de **permutations**. Contrairement à la section précédente, nous spécifions une forme particulière pour le secret : **le secret de l'utilisateur est une permutation** dont la longueur sera fonction du niveau de sécurité exigé pour le protocole d'authentification.

Pour rappel, la clé secrète $Kmac$ dédiée aux calculs de MAC du composant d'authentification est toujours stockée dans sa mémoire sécurisée : $Kmac$ définit ainsi l'information secrète qui caractérise le composant d'authentification. Le secret de l'utilisateur, qui est ici une permutation, est stockée dans cette même mémoire sécurisée après une phase d'enregistrement.
La phase de vérification off-line est constituée des étapes suivantes : le composant d'authentification prouve à l'utilisateur qu'il connaît la permutation secrète et la clé

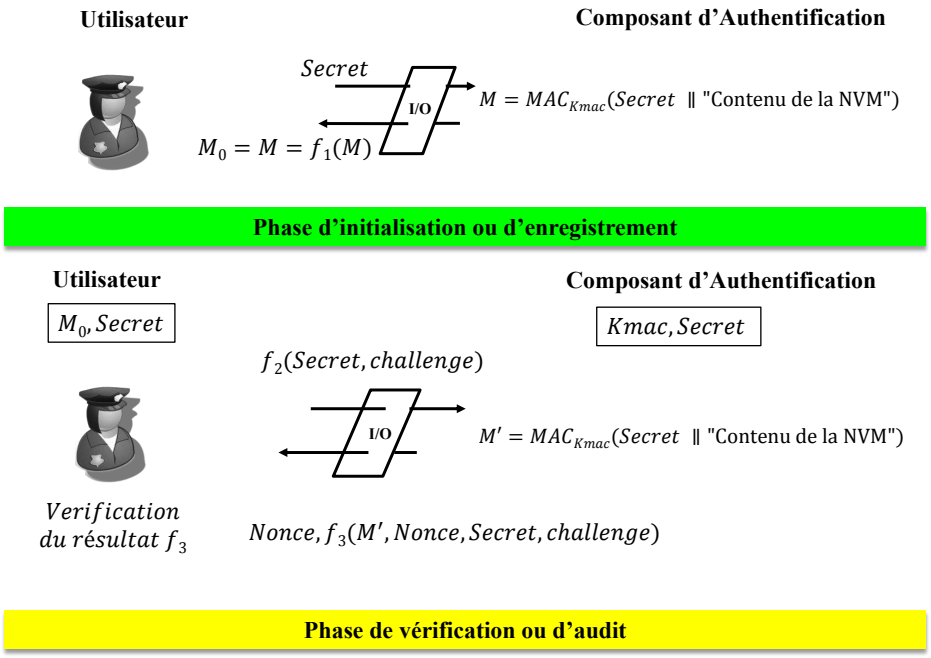

Figure IV.3 *Principe du protocole d'authentification autonome outil de calculs externe*

Les données publiques : $M_0 = M$.

Les données secrètes : $Secret, Kmac$.

Les données encadrées, représentent les connaissances de l'utilisateur ou du composant en début de phase de vérification.

— ○ —

$Kmac$, puis que son contenu de configuration provenant de la NVM correspond à celui de la phase d'enregistrement. Ici, comme décrit dans sur la Figure IV.3, nous proposons une vérification en deux messages. L'utilisateur envoie un premier message généré par la fonction f_2 avec pour paramètre sa permutation secrète et un challenge qu'il choisit au hasard. Le second message est la réponse du composant retournée par la fonction f_3 appliquée aux différentes données d'authentification et un nonce aléatoire (pour prévenir tout rejeu de ce message ou attaque à clair connu).

Avant de spécifier ce protocole par ses phases d'enregistrement (section IV.4.2.1) et de vérification (section IV.4.2.2), nous spécifions à la section suivante le vocabulaire qui nous permettra de définir les fonctions f_2 et f_3.

IV.4.1 Définitions et notations

Nous rappelons les significations des notations suivantes, que nous utilisons pour décrire notre protocole :

- $x = y$, signifie que la variable x prend la valeur de la variable y.
- $x \xleftarrow{r} X$, signifie que la variable x est choisie au hasard dans l'ensemble X.
- $\#X$ désigne le nombre d'éléments (le cardinal) d'un ensemble X.

Définition 10 (Permutation)
En général, une permutation sur un ensemble fini $\mathcal{S}$ est définie comme une fonction bijective de $\mathcal{S}$ sur lui-même.
Ici, nous ne considérons que les permutations sur les n premiers entiers soit $\mathcal{S} = \{1, \cdots, n\}$. Nous noterons $\mathfrak{S}_n$ l'ensemble des permutations sur $\mathcal{S}$.

Définition 11 (Le groupe de permutation $(\mathfrak{S}_n, \circ)$) *:*
Nous considérons ici, la structure de groupe sur les permutations de $\mathfrak{S}_n$ appelé **groupe symétrique**.

- La loi de composition est définie telle que toute permutation σ, σ' et $\sigma'' \in \mathfrak{S}_n$

$$\sigma = \sigma' \circ \sigma'' \Leftrightarrow \forall i \in \{1, \cdots, n\}, \sigma(i) = \sigma'(\sigma''(i)). \tag{IV.11}$$

- On notera i_n l'identité du groupe $(\mathfrak{S}_n, \circ)$ c'est à dire l'élément neutre de la loi de groupe.
- Pour une permutation σ de $\mathfrak{S}_n$, nous notons $\sigma^{-1} \in \mathfrak{S}_n$ sa permutation inverse.

Représentation des permutations :
Dans cette section, nous représentons une permutation $\sigma \in \mathfrak{S}_n$ comme une ligne constituée des différentes valeurs $\sigma(i)$, i allant de 1 à n :

$$\sigma = \sigma(1) \ \ \sigma(2) \ \ \cdots \ \ \sigma(n) \tag{IV.12}$$

Exemple 1 *:*

- Pour $n = 4$, $\sigma = 4\ 2\ 1\ 3 \in \mathfrak{S}_4$, représente la permutation définie par $\sigma(1) = 4, \sigma(2) = 2, \sigma(3) = 1$ et $\sigma(4) = 3$.
- De même $\sigma = 8\ 15\ 2\ 16\ 10\ 7\ 4\ 3\ 11\ 14\ 6\ 12\ 9\ 5\ 13\ 1$ représente une permutation de $\mathfrak{S}_{16}$ telle que $\sigma(1) = 8,\ \sigma(2) = 15,\ \cdots,\ \sigma(16) = 1$.

Définition 12 (Support d'une permutation)
Le support d'une permutation $\sigma \in \mathfrak{S}_n$ est l' ensemble noté $Supp(\sigma)$ des entiers k tel que $\sigma(k) \neq k$.

$$Supp(\sigma) = \{1 \leq k \leq n, \text{ telle que } \sigma(k) \neq k\} \subseteq \{1, \cdots, n\}$$

Par exemple $Supp(i_n) = \emptyset$.

Définition 13 (Alphabet Σ & Mot)
Pour la représentation de messages tels que des empreintes MAC ou des nonces, nous définissons la notion d'**alphabet** et la notion de **mot** qui en découle. Nous appelons **alphabet** un ensemble fini de symboles. Pour un alphabet Σ, et un entier positif non nul k, nous appelons **mot** de Σ^k toute suite de k éléments de Σ.

Définition 14 (La fonction $\mathcal{E}nc$:)
Nous définissons la fonction $\mathcal{E}nc$ permettant de formaliser l'application d'une permutation à un mot. Etant donné un alphabet Σ et un entier positif non nul n, la fonction $\mathcal{E}nc$ est définie de $\mathfrak{S}_n \times \Sigma^n$ vers Σ^n de telle sorte que le mot $\mathcal{E}nc(\sigma, M) \in \Sigma^n$ soit définie à partir du mot $M = (M_i)_{i=1,\cdots,n} \in \Sigma^n$ en changeant l'ordre des éléments M_i selon la permutation $\sigma \in \mathfrak{S}_n$ soit :

$$E = \mathcal{E}nc(\sigma,\ M) \Leftrightarrow E = M_{\sigma(1)}, \cdots, M_{\sigma(n)} \Leftrightarrow \forall i \in \{1, \cdots, n\}\ E_i\ =\ M_{\sigma(i)} \qquad \text{(IV.13)}$$

Ainsi,

$$\forall E, M \in \Sigma^n \ \text{ et } \ \sigma \in \mathfrak{S}_n, \text{on a } \ E = \mathcal{E}nc(\sigma,\ M) \Leftrightarrow M = \mathcal{E}nc(\sigma^{-1},\ E) \qquad \text{(IV.14)}$$

Exemple 2 *:*

1. Soient $\Sigma = \{a, b, c, d\}, n = 6$, et le mot $M = \quad M_1M_2M_3M_4M_5M_6 \quad = abbcab$

$$\begin{array}{rcl} \sigma_1 = 3\ 1\ 5\ 4\ 6\ 2, & \mathcal{E}nc(\sigma_1,\ M) & = M_3M_1M_5M_4M_6M_2 \ = baacbb \\ \sigma_2 = 6\ 1\ 4\ 3\ 2\ 5, & \mathcal{E}nc(\sigma_2,\ M) & = M_6M_1M_4M_3M_2M_5 \ = bacbba \end{array}$$

2. Soit l'alphabet $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$; et la permutation $\sigma = 8\ 9\ 11\ 12\ 5\ 1\ 15\ 2\ 10\ 6\ 16\ 13\ 4\ 3\ 7\ 14$, on a alors

$$\begin{array}{rcllll} M & = & BFF2 & D513 & 27DE & E382 \\ \mathcal{E}nc(\sigma,\ M) & = & 32DE & DB8F & 752E & 2F13 \end{array}$$

$$\begin{array}{rcllll} M' & = & E3B9 & 198B & DD5D & 24D2 \\ \mathcal{E}nc(\sigma,\ M') & = & BD5D & 1ED3 & D922 & 9B84 \end{array}$$

IV.4.2 Protocole d'authentification

Maintenant que nous avons introduit le vocabulaire nécessaire et rappelé la notion de groupe de permutations, nous pouvons détailler le protocole schématisé à la Figure IV.3. Nous supposons que la valeur de la longueur n du secret de l'utilisateur est préalablement fixée, et que toutes les opérations sur les permutations de $\mathfrak{S}_n$ sont implantées dans le contrôleur de configuration du composant d'authentification.

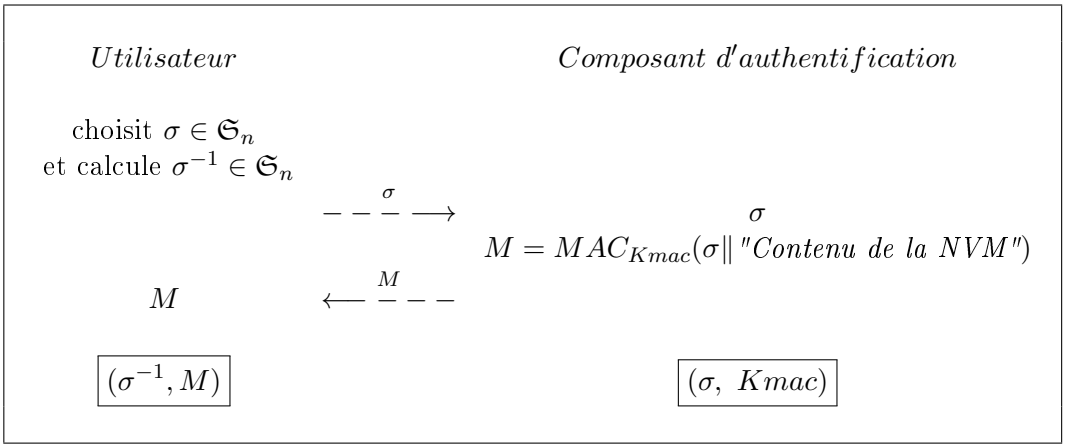

Figure IV.4 *Phase d'enregistrement du protocole d'authentification autonome sans outil de calculs externe*

Les données encadrées sont celles qu'enregistrent l'utilisateur et le composant d'authentification à la fin de cette phase.

– ∘ –

IV.4.2.1 Phase d'enregistrement

Chaque phase d'enregistrement doit se dérouler comme suit (Figure IV.4) :

1. l'utilisateur commence par choisir aléatoirement une permutation $\sigma \in \mathfrak{S}_n$. Lorsque σ est *non trivial*[2] c'est à dire lorsque σ est différente typiquement de l'identité i_n, alors l'utilisateur calcule la permutation inverse σ^{-1}. Le calcul de σ^{-1} permettra à l'utilisateur de gagner du temps sur les opérations de la phase de vérification. La permutation σ représente ici le secret de l'utilisateur, il le partage donc avec le composant d'authentification.

2. A la réception de la permutation σ, le composant la stocke comme prévu dans sa mémoire sécurisée. Puis il calcule l'empreinte de référence M à partir de cette permutation et du contenu de provenant de la NVM :

$$M \ = \ MAC_{Kmac}(\sigma \| "Contenu\ de\ la\ NVM") \tag{IV.15}$$

L'empreinte ainsi calculée, est retournée à l'utilisateur sans être stockée dans le composant. L'utilisateur enregistre l'empreinte authentique de façon sûr pour se prévenir de toute modification de sa valeur.

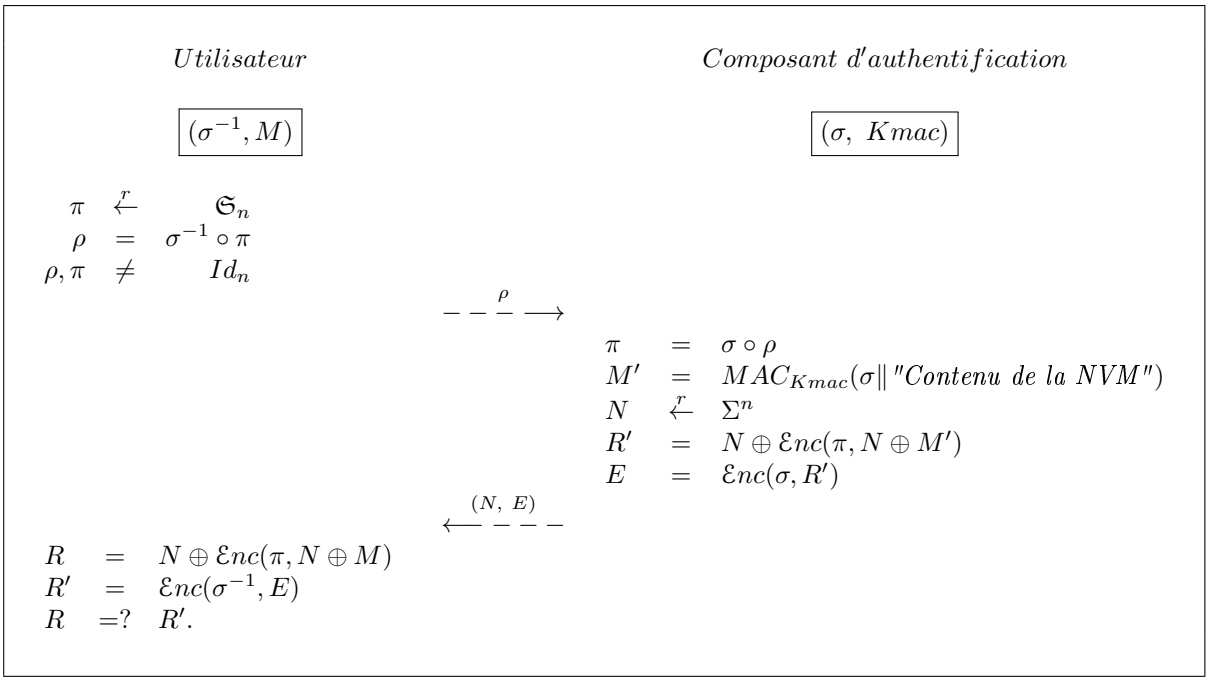

Figure IV.5 *Phase de vérification du protocole d'authentification autonome sans outil de calculs externe*

Les données publiques : M, ρ, E, N (les valeurs de ρ, E et N changent à chaque session)

Les données secrètes : $\pi, \sigma, \sigma^1, Kmac$.

Les données encadrées, représentent les connaissances de l'utilisateur ou du composant en début de phase de vérification.

$- \circ -$

IV.4.2.2 Phase de vérification

Dans cette configuration, nous supposons donc que l'utilisateur possède, à la fois la permutation σ^{-1} et l'empreinte MAC M. Il s'agit de vérifier si le composant d'authentification possède la permutation σ (le secret de l'utilisateur) et la clé secrète $Kmac$ de calcul de MAC, et de simultanément vérifier l'intégrité du contenu de configuration du composant provenant de la NVM. Le processus de vérification se définit comme suit (Figure IV.5) :

1. L'utilisateur envoie un challenge au composant d'authentification (voir fonction f_2 à la figure IV.3). Pour cela, l'utilisateur choisit aléatoirement une permutation $\pi \in \mathfrak{S}_n$ avant de calculer la permutation $\rho = \sigma^{-1} \circ \pi \in \mathfrak{S}_n$. Si l'une des permutations ρ ou π est égale à l'identité $i_n \in \mathfrak{S}_n$, l'utilisateur choisit une autre permutation

2. Nous reviendrons dans la section suivante pour définir l'ensemble des permutations σ acceptables vis à vis d'un niveau de sécurité donné.

π et calcule la nouvelle valeur de ρ, sinon l'utilisateur envoie au composant la permutation ρ.

2. Dès réception de la permutation ρ par le composant d'authentification :
   - le composant calcule un MAC M' du contenu configuration provenant de NVM et de la permutation secret σ :

$$M' = MAC_{Kmac}(\sigma \| "Contenu\ de\ la\ NVM") \tag{IV.16}$$

   - le composant calcule la permutation challenge π en utilisant σ, en effet $\pi = \sigma \circ \rho$. Il choisit aléatoirement un nonce $N \in \Sigma^n$ puis calcule le message de réponse E pour l'utilisateur :

$$E = \mathcal{E}nc(\sigma, N \oplus \mathcal{E}nc(\pi, N \oplus M')). \tag{IV.17}$$

   Enfin, le composant d'authentification renvoie à l'utilisateur la paire (N, E).
3. A la réception de la paire (N, E), l'utilisateur effectue les opérations suivantes afin de vérifier la valeur du MAC M' et le calcul de π par le composant :
   - calculer le xor : $N \oplus M$
   - appliquer la permutation π sur le mot $N \oplus M$: $\mathcal{E}nc(\pi, N \oplus M)$
   - calculer : $R = N \oplus \mathcal{E}nc(\pi, N \oplus M)$
   - appliquer la permutation σ^{-1} sur le mot E : $R' \leftarrow \mathcal{E}nc(\sigma^{-1}, E)$
   - et enfin comparer les mots R et R'.

En suivant cette procédure, il est évident (équation IV.14) que $R = R'$ si et seulement si le composant d'authentification du système embarqué utilise le secret σ et que $M = M'$, ce qui voudrait dire que la clé secrète $Kmac$, et le contenu provenant de la NVM sont conformes à celles de la phase d'enregistrement.

IV.4.3 Sécurité du protocole

Nous allons discuter dans cette section des possibles faiblesses du protocole que nous venons de décrire. Il s'agit de définir les meilleures valeurs pour les permutations σ et π afin d'assurer un niveau de sécurité élevé au processus d'authentification contre d'éventuels adversaires. Car, contrairement aux ressources limitées de l'utilisateur, l'attaquant dispose de ressources importantes comme une machine performante de calcul et de mémorisation.

IV.4.3.1 Choix de la permutation secrète

Nous savons que pour tout entier positif non nul n, l'ensemble $\mathfrak{S}_n$ contient $n!$ permutations ($\#\mathfrak{S}_n = n!$). Cependant, pour un minimum de sécurité, compte tenu de certaines propriétés du groupe des permutations de $(\mathfrak{S}_n, \circ)$, il est **préférable** de bien choisir la permutation secrète σ ainsi que le challenge π de l'utilisateur.

Il est avantageux pour un utilisateur d'effectuer plusieurs phases de vérification après une seule phase d'enregistrement, cela implique d'utiliser plusieurs fois la même permutation secrète σ. Donc, un attaquant qui aurait enregistrer des sessions de vérification avec le même secret peut en déduire de l'information sur σ.
Par exemple, si la permutation secrète σ commute avec une série de challenge $(\pi_i)_{(1,\cdots,r)}$, choisie par l'utilisateur au cours des différentes sessions de la phase de vérification (Figure IV.5) alors un attaquant peut écrire différentes combinaisons sur les éléments de la série $(\rho_i)_{(1,\cdots,r)}$, où $\rho_i = \sigma^{-1} \circ \pi_i$. Ces combinaisons peuvent lui fournir de l'information sur la permutation σ. Pour cela l'utilisateur devra choisir les permutations σ et π pour qu'elles aient moins de chance de commuter entre elles.

Le support d'une permutation (*cf.* définition 12) est une notion adéquate pour étudier la commutativité dans $(\mathfrak{S}_n, \circ)$. En effet deux permutations à supports disjoints commutent toujours (l'équation (IV.18)) :

$$\forall \sigma_1, \sigma_2 \in \mathfrak{S}_n,\ Supp(\sigma_1) \cap Supp(\sigma_2) \ = \ \emptyset \Rightarrow \sigma_1 \circ \sigma_2 = \sigma_2 \circ \sigma_1 \tag{IV.18}$$

Par ailleurs, plus le support de la permutation σ est petit, plus le nombre de permutations qui peuvent commuter avec σ est important. **Il convient donc d'éviter pour le choix de σ et π, les permutations qui ont un "petit" support.**

Pour le challenge π, il nous faut aller plus loin et complètement éviter les permutations à point fixe. En effet, s'il s'avérait pour un attaquant qu'il existe $i \in \{1, \cdots, n\}$ tel que $\pi(i) = i$, alors sans connaître π, il peut en déduire que $R'_i = M'_i = M_i$ (les mots R' et M' du protocole sont définis à la section IV.4.2.2). De plus, comme la valeur de l'empreinte de référence M est censée ne pas être secrète, l'égalité $R'_i = M_i$ peut donner de l'information sur la permutation secrète σ. En effet, en connaissant E, il est possible pour un attaquant d'avoir de l'information sur $\sigma(i)$ en se servant de l'égalité $E = \mathcal{E}nc(\sigma, R')$. Cette égalité donne par définition $E_i = R'_{\sigma(i)}$, ce qui restreint les valeurs possibles de $\sigma(i)$ à l'ensemble $\{k, \text{tel que } E_k = R'_{\sigma(i)}\}$.
Pour éviter ce risque, il convient que les permutations π soient toujours de support maximal, c'est à dire sans point fixe ($\forall i \in \{1, \cdots, n\}, \pi(i) \neq i$ est équivalent à $Supp(\pi) = \{1, \cdots, n\}$).

Pour un n donné, nous savons calculer le nombre de permutations de $\mathfrak{S}_n$ qui sont sans point fixe. Si on note $FP_0(n)$ l'ensemble des permutations de $\mathfrak{S}_n$ sans point fixe, le nombre d'éléments de $FP_0(n)$ est donné par l'équation IV.19 ci-dessous. Cette formule est en effet un résultat classique de l'analyse combinatoire [Com74].

$$\#FP_0(n) = n! \left(\sum_{i=2}^{n} \frac{(-1)^i}{i!} \right) \ \approx \ \frac{n!}{e} \tag{IV.19}$$

où e désigne le nombre d'Euler $e \approx 2.71828$.

Nous déduisons de la formule de $\#FP_0(n)$, le nombre $\#FP_1(n)$ des permutations de $\mathfrak{S}_n$ ayant un seul point fixe. En effet, les permutations de $FP_1(n)$ peuvent toutes être construites à partir de celles de $FP_0(n-1)$ avec n choix pour le point fixe, d'où :

$$\#FP_1(n) = n \cdot \#FP_0(n-1) \approx \frac{n!}{e}. \tag{IV.20}$$

En somme, **le secret σ doit être une permutation ayant un support suffisamment grand. Nous considérons pour cela que σ possède au plus un point fixe, c'est à dire que** $\sigma \in FP_0(n) \cup FP_1(n)$, cette ensemble représentant plus de deux tiers des permutations de $\mathfrak{S}_n$. Par contre, **pour la permutation π, l'utilisateur ne doit choisir que des permutations sans point fixe** ($\sigma \in FP_0(n)$). Plus d'un tiers des éléments de $\mathfrak{S}_n$ sont ainsi éligibles pour le choix de π. En prenant en compte les permutations avec au plus un point fixe (zéro ou un point fixe), il y a plus de deux tiers de $\mathfrak{S}_n$ pour choisir son secret σ.

Dans les prochaines sections, nous désignons par $\mathcal{K}(\mathfrak{S}_n) \subset \mathfrak{S}_n$ le sous-ensemble des permutations réservées (par l'utilisateur ou par le concepteur du protocole) à σ et π, soit un choix parmi :

$$\#\mathcal{K}(\mathfrak{S}_n) \approx \frac{n!}{e}. \tag{IV.21}$$

IV.4.3.2 Les attaques génériques

La fonction de permutation $\mathcal{E}nc$ (équation IV.13) peut également être considérée comme une primitive de chiffrement symétrique (équation IV.14), définie sur un mot de n éléments de l'alphabet Σ et une permutation secrète choisie dans $\mathcal{K}(\mathfrak{S}_n)$. Nous définissons son déchiffrement comme suit :
Si $C = (C_1, \cdots, C_n) = \mathcal{E}nc(\sigma, M) \in \Sigma^n$, alors :

$$\mathcal{D}ec(\sigma, C) = \mathcal{E}nc(\sigma^{-1}, C) = (C_{\sigma^{-1}(1)}, \cdots, C_{\sigma^{-1}(n)}) \in \Sigma^n \tag{IV.22}$$

Discutons ci-dessous de la sécurité de notre protocole en fonction de la fonction $\mathcal{E}nc()$.
Attaque à texte clair connu :
Comme tout système de chiffrement linéaire, la fonction $\mathcal{E}nc$ ne résiste pas à une attaque à texte clair connu. Cela signifie que, dès qu'un attaquant possède deux mots M et $C \in \Sigma^n$, tels que $C = \mathcal{E}nc(\sigma, M)$, il peut en déduire la permutation σ. En effet, par définition de $\mathcal{E}nc$, pour tout i (de 1 alors n), on a $\sigma(i) \in \{k, \text{tel que } M_k = C_i\}$. Or pour un i donné, le nombre d'éléments de l'ensemble $\{k, \text{tel que } M_k = C_i\}$ dépend fortement de la redondance de M_i dans $M \in \Sigma^n$ (c'est à dire du nombre d'occurrences de M_i). Par exemple, si les symboles M_i composants le mot $M = (M_i)$ sont différents, alors tous les k, tels que $M_k = C_i$ ont un seul élément qui définit $\sigma(i)$. Cela veut dire que plus le nombre d'éléments de l'alphabet Σ est élevé (et que n est petit), moins la redondance dans les mots $M \in \Sigma^n$ est assurée.

Ainsi, nous supposons que la fonction $\mathcal{E}nc$ est vulnérable à une attaque à clair connu. De ce fait, dans l'utilisation que nous faisons de la fonction $\mathcal{E}nc$ dans le protocole d'authentification (*cf.* IV.5), aucun message connu, ne transite entre l'utilisateur et le système à authentifier. En aucune façon un observateur différent de l'utilisateur et du composant d'authentification ne peut avoir un couple (message,chiffré-du-message), la fonction de Xor par un nonce aléatoire nous aide en cela. Ainsi, la faiblesse de la fonction $\mathcal{E}nc$ ne joue pas dans le protocole que nous avons décrit.

Attaque à texte chiffré connu :
Dans la phase de vérification du protocole (Figure IV.5), seuls le mot $E \in \Sigma^n$ et la permutation ρ sont produits à partir du secret σ et transitent à travers l'interface d'entrée/sortie du système.

La question ici, est combien de sessions de vérification un utilisateur peut réaliser après une seule phase d'enregistrement, sans qu'un attaquant ne puisse déduire de l'information sur la permutation secrète $\sigma \in \mathcal{K}(\mathfrak{S}_n)$à partir des différentes valeurs de ρ et E qu'il aurait enregistrées. Cette question revient à déterminer le nombre de permutations ρ et de couples (N, E) qui seraient suffisants, pour qu'un attaquant détermine le secret σ, sans utiliser une recherche exhaustive.

Tout d'abord, concernant la paire (N, E) (où $E = \mathcal{E}nc(\sigma, R')$, $R' = N \oplus \mathcal{E}nc(\pi, N \oplus M')$), la valeur de R' ne peut être déduite uniquement de la paire (N, E), parce que la valeur N a été choisie aléatoirement par le composant d'authentification et que la permutation π est inconnue pour tout observateur ne possédant pas σ.

Avec cette condition, le nombre de rondes qu'il faudrait à un attaquant pour tirer de l'information sur σ à travers Enc est défini par **la distance d'unicité** r de Enc sur les n-blocs de Σ^n La distance d'unicité ([MVOV96] Chapitre 7) est donnée par :

$$r = \frac{H(\mathcal{K}(\mathfrak{S}_n))}{D}, \tag{IV.23}$$

où $H(\mathcal{K}(\mathfrak{S}_n))$ est l'entropie de l'espace des clés $\mathcal{K}(\mathfrak{S}_n)$ de la fonction de chiffrement $\mathcal{E}nc$, et D définit la redondance des éléments de Σ.

Toutefois dans la description de ce protocole, le mot à chiffrer correspond toujours à un Xor entre des nonces aléatoires et un autre mot, de ce fait la redondance usuelle de la fonction $\mathcal{E}nc$ est nulle ou très faible. Cela implique une grande valeur pour le nombre de ronde défini par r. Cependant, il est difficile de déterminer la valeur exacte de la redondance D. Pour cela, nous recommandons à l'utilisateur de **définir un nombre maximum r_{max} de sessions de vérification qui peuvent se faire avec un même secret σ.**

Concernant la permutation ρ, nous savons qu'à toute permutation $\rho \in \mathfrak{S}_n$ correspond une permutation π choisie au hasard dans $\mathcal{K}(\mathfrak{S}_n)$ par l'utilisateur ($\rho = \sigma^{-1} \circ \pi$).

Ainsi pour chaque ρ nous avons $\sigma = \pi \circ \rho^{-1}$ pour un unique π choisi au hasard dans $\mathcal{K}(\mathfrak{S}_n)$. Donc, pour déterminer σ en fonction d'une série de ρ, un attaquant devrait trouver une relation sur la série correspondante de π. Or par construction, les permutations π sont toutes choisies aléatoirement à des temps différents donc elles sont indépendantes les unes des autres. Cela, nous amène à conclure que, si l'utilisateur choisit correctement (c'est à dire aléatoirement et indépendamment) les permutations π, alors il est difficile de déterminer σ en partant de différents ρ.

Attaque par force brute :
En partant des messages que peut avoir un attaquant qui espionne une phase de vérification d'un système embarqué, la question que l'on peut se poser, est la possibilité pour un attaquant de déterminer le secret σ juste en fonction de ses connaissances. Parmi les données en la possession d'un attaquant, il y a les messages ρ, N, E qui sont échangés entre l'utilisateur et le système à travers l'interface d'entrée/sortie. En plus de ces messages, l'attaquant peut connaître l'empreinte de référence M, qui n'est pas supposée être secrète.

Avec ces données, une attaque par force brute ou exhaustive sur ρ, consiste à chercher exhaustivement la valeur de $\sigma \in \mathcal{K}(\mathfrak{S}_n)$, jusqu'à ce que les messages $\mathcal{E}nc(\sigma, N \oplus \mathcal{E}nc(\sigma \circ \rho, N \oplus M))$ et E soient égaux. De cette façon, il suffit à l'attaquant de faire, au maximum $\frac{n!}{e}$ essais pour déterminer la vrai valeur de σ.

Pour rendre inopérante cette attaque, la valeur n doit être choisie suffisamment grande pour qu'une recherche exhaustive soit insoluble entre deux phases d'initialisation (c'est à dire, durant la vie du secret σ). Il faudrait alors **définir une période maximale T_{max} entre deux phases d'initialisation**. La valeur de T_{max} dépendra ainsi de la valeur de n et du degré de sécurité souhaité par l'utilisateur.

IV.4.3.3 Choix de la valeur de n

Selon la section précédente IV.4.3.1, l'ensemble $\mathcal{K}(\mathfrak{S}_n)$ des permutations possibles pour le secret σ et le challenge π, dépend du support des permutations (du nombre de points fixes), mais surtout de la taille n. L'ensemble $\mathcal{K}(\mathfrak{S}_n)$ est constitué d'au moins le tiers des éléments de $\mathfrak{S}_n$: $\#\mathcal{K}(\mathfrak{S}_n) \approx \frac{n!}{e}$ soit dans l'approximation de Stirling du factorielle ($n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$,) :

$$\#\mathcal{K}(\mathfrak{S}_n) \approx \frac{\sqrt{2\pi n}}{e} \cdot \left(\frac{n}{e}\right)^n \approx 2^{(n+\frac{1}{2})\log_2(n) - n/\log(2)} \tag{IV.24}$$

Le tableau IV.1, montre les différentes valeurs de n, pour un alphabet Σ à 16 ou 256 éléments (qui correspond respectivement à $m = 4$ et $m = 8$ sur la tableau IV.1 avec $m = \log_2(\#\Sigma)$) et un nombre de bits différents pour les empreintes MAC. Le tableau montre que pour une fonction de calcul de MAC donnée, et pour un alphabet Σ, on doit augmenter la longueur de permutation pour accroître le niveau de sécurité du

$m = 4$	Le nombre de bits du MAC	64	128	160	192	256	512
	n	16	32	40	48	64	128
	$\log_2(\#\mathcal{K}(\mathfrak{S}_n))$	42	116	157	201	294	714
$m = 8$	Le nombre de bits du MAC	64	128	160	192	256	512
	n	8	16	20	24	32	64
	$\log_2(\#\mathcal{K}(\mathfrak{S}_n))$	13	42	59	77	116	294

Tableau IV.1 *Une estimation du nombre des permutations de $\mathcal{K}(\mathfrak{S}_n)$ en fonction de n, de la longueur des empreintes MAC et du nombre d'éléments de l'alphabet Σ ($\#\Sigma \approx 2^m$), selon l'équation (IV.24) :* $\log_2(\#\mathcal{K}(\mathfrak{S}_n)) = \lfloor (n + 1/2)\log_2(n) - n/\log(2) \rfloor$.

– ∘ –

protocole. Ceci implique d'augmenter le cardinal de $\mathcal{K}(\mathfrak{S}_n) \subset \mathfrak{S}_n$.

Cependant, notre défi est que le protocole que nous proposons ici soit aisément réalisable par un utilisateur humain. Ainsi nous avons le choix entre faciliter les opérations de l'utilisateur avec des permutations courtes ou rendre calculatoirement longue une attaque exhaustive en choisissant des permutations suffisamment longues. Nous discutons ci-dessous (dans la section IV.4.3.4) de la difficulté de calcul que représente pour un utilisateur les opérations à faire lors d'une vérification.

Un compromis serait de définir les paramètres r_{max} (le nombre maximum de sessions pour un même secret σ) et la période T_{max} (le période de validité) tels que, les permutations soient de taille "intermédiaire" et qu'il soit en pratique impossible de faire une attaque exhaustive pendant la période T_{max}.

IV.4.3.4 Complexité calculatoire pour l'utilisateur

Nous venons de montrer l'importance du choix de n : plus n est grand, plus une attaque exhaustive est difficile. Cependant, notre objectif premier est que ce protocole soit entièrement réalisable par un être humain. Nous allons essayer de mesurer la difficulté à executer les calculs que peut rencontrer un utilisateur en participant à ce protocole.

Pour cela, le choix de l'alphabet est vaste, il faut en effet qu'un utilisateur puisse manipuler aisément les données afin d'effectuer efficacement ses opérations de vérification. Dans ce protocole, il nous est facile de supposer une relation entre le nombre de bits de l'empreinte MAC, l'alphabet Σ et la taille n des permutations. En effet, si nous considérons que l'empreinte MAC M est un mot de Σ^n, alors le nombre de bits du MAC est forcément un multiple de n et vaut $n \cdot m$ (où $\#\Sigma = 2^m$).

Cela veut dire que pour une fonction de calcul de MAC fixée, on est libre de choisir n plus grand ou plus petit pour un alphabet plus ou moins important. Par

exemple, si la fonction MAC est basée sur SHA256 (ou l'AES256), on peut choisir les permutations de $\mathfrak{S}_{64}$ avec une subdivision en hexadécimal (c'est à dire que Σ est l'ensemble des chiffres hexadécimaux) ou des permutations de $\mathfrak{S}_{32}$ avec une subdivision en octet (Σ est l'ensemble des octets).

Le protocole que nous proposons exige de l'utilisateur, lors de la **phase d'enregistrement** (*cf.* Figure IV.4), de choisir aléatoirement une permutation, puis de calculer l'inverse de cette permutation. Ces opérations effectuées une seule fois, en lieu sûr, n'influence pas la difficulté calculatoire des opérations réalisées par l'utilisateur en phase de vérification. Cependant **la mémorisation d'un permutation** de n caractères différents et ordonnés, peut paraître difficile pour un utilisateur après une phase d'enregistrement.

Lors de chaque **phase de vérification** off-line (*cf.* Figure IV.5), l'utilisateur vérifieur doit nécessairement effectuer les opérations suivantes :

- le choix de la permutation π puis le calcul par composition de $\rho = \sigma^{-1} \circ \pi$,
- 2 opérations Xor sur deux mots de Σ^n (c'est à dire le calcul de $N \oplus M$ et de R) soient $2n$ lectures dans une table de Xor remplie de lettres de l'alphabet Σ,
- 2 appels de la fonction $Enc(\ )$ avec la permutation π, puis avec σ^{-1}, soit $2n$ réécritures de lettres de Σ,
- 1 comparaison de deux mots R et R'.

Sans minimiser le choix de la permutation π et la comparaison des deux mots R et R', nous pouvons dire qu'à chaque vérification off-line, un utilisateur effectue 3 opérations avec permutations et 2 opérations Xor de mots de longueur n (Figure IV.6). En considérant que chacune des ces opérations se décompose en n manipulations des éléments de l'alphabet Σ, **la difficulté globale d'une phase de vérification pour l'utilisateur est de l'ordre de 5n manipulations sur les éléments de l'alphabet** Σ. Cela montre que cette complexité dépend aussi de la taille n des permutations. De cela on peut dire que, plus n est grand, plus le protocole est difficile pour l'utilisateur humain, cependant cette difficulté est linéaire par rapport à n.

Il paraît claire qu'effectuer de tête ces opérations n'est pas facilement réalisable pour un être humain. Par contre, en considérant que l'utilisateur possède un support papier ou une ardoise (une ardoise physique et non une ardoise numérique) ainsi qu'une table de Xor comme celle du tableau IV.2 (en général une table Xor hexadécimal suffit pour les alphabets usuels que nous considérons, pour $m = 4$ ou $m = 8$), nous pensons que réaliser ce protocole de façon autonome est à sa porté moyennant un entraînement et une pratique régulière des calculs de Xor et permutations de longueur n.

IV.5 Conclusion

Nous venons de présenter dans ce chapitre nos travaux sur l'authentification autonome d'un système embarqué par un humain. Nous avons ainsi décrit trois protocoles,

$\oplus$	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	0	3	2	5	4	7	6	9	8	B	A	D	C	F	E
2	2	3	0	1	6	7	4	5	A	B	8	9	E	F	C	D
3	3	2	1	0	7	6	5	4	B	A	9	8	F	E	D	C
4	4	5	6	7	0	1	2	3	C	D	E	F	8	9	A	B
5	5	4	7	6	1	0	3	2	D	C	F	E	9	8	B	A
6	6	7	4	5	2	3	0	1	E	F	C	D	A	B	8	9
7	7	6	5	4	3	2	1	0	F	E	D	C	B	A	9	8
8	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7
9	9	8	B	A	D	C	F	E	1	0	3	2	5	4	7	6
A	A	B	8	9	E	F	C	D	2	3	0	1	6	7	4	5
B	B	A	9	8	F	E	D	C	3	2	1	0	7	6	5	4
C	C	D	E	F	8	9	A	B	4	5	6	7	0	1	2	3
D	D	C	F	E	9	8	B	A	5	4	7	6	1	0	3	2
E	E	F	C	D	A	B	8	9	6	7	4	5	2	3	0	1
F	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0

Tableau IV.2 *Table Hexadécimal de Xor bit à bit*

– ∘ –

tous fondés sur l'introduction d'une information secrète de l'utilisateur dans le calcul de l'empreinte de référence du système. Cette information représente le secret qui permettra une authentification du système par l'utilisateur. Selon notre principe, ce secret assure également, de par son unicité, le rôle d'élément d'identification à la fois du composant d'authentification du système et de son contenu de configuration.

A partir du processus de vérification, nous avons proposé trois protocoles d'authentification où l'humain est effectivement plus actif que dans les rôles qui lui sont attribués dans la littérature aujourd'hui. Nous avons ainsi présenté deux niveaux de participation (active) de l'utilisateur. Dans le premier, l'utilisateur humain a toujours à sa disposition un outil de calculs suppléant sa faible performance calculatoire, dans le second, l'outil de calculs est supprimé en simplifiant les fonctions cryptographiques impliquées, de façon à permettre une vérification 100 % humaine.

Nos deux premiers protocoles sont ainsi caractérisés par un outil tiers de calcul, mais ici cet outil n'est nullement lié ni au système à authentifier. Par cela, nous avons donné la possibilité à l'utilisateur de choisir et de changer d'outils à souhait. Cette liberté de choix de l'utilisateur, est l'avantage principal de nos deux protocoles par rapport aux protocoles basés sur des machines génératrices ou délivreuses de jetons d'authentification (*cf.* section III.3 du chapitre III).

En se servant de la propriété de diffusion qu'apporte les permutations et les propriétés de confusion qu'offre la fonction Xor avec un paramètre aléatoire, nous avons développé un protocole original, par lequel un humain peut authentifier directement et

en toute indépendance (sans tiers de confiance) un système embarqué.

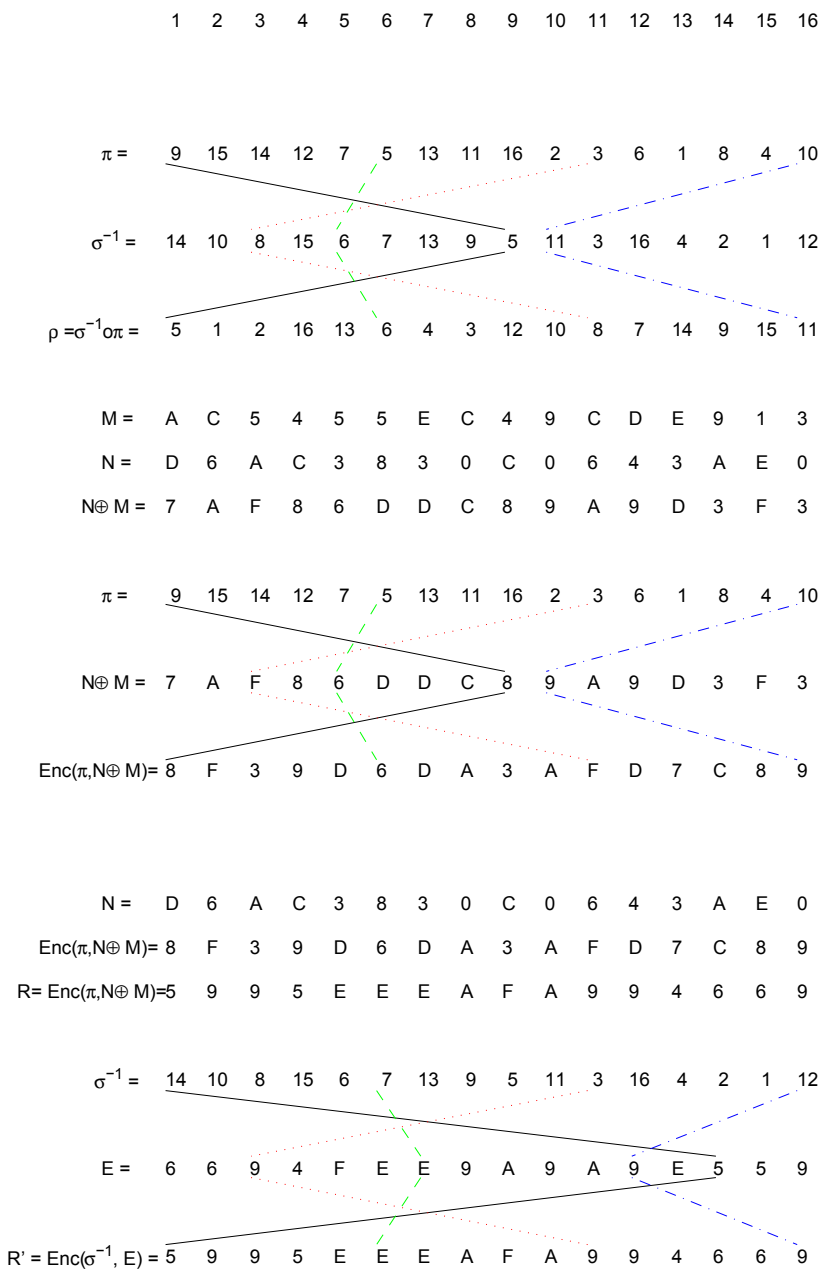

Figure IV.6 *Les opérations de l'utilisateur lors d'un phase de vérification du protocole. Pour cet exemple, nous avons des empreintes MAC sur* 64 *bits, avec* $n = 16$ *et* Σ *les chiffres hexadécimaux.*

Conclusion

Dans ces travaux, nous avons posé un cadre à l'authentification d'objets à travers ses phases d'enregistrement d'empreintes caractéristiques, et de vérification autonome au moyen d'un système embarqué. Une telle authentification présuppose donc de disposer d'une méthode de caractérisation. Comme point de départ, nous avons développé l'exemple des "Print signatures", représentatif d'une authentification locale, faisant usage d'un dispositif matériel dédié à la double vérification de l'empreinte et de la signature électronique imprimé sur le document, sans accès à une base de données.

Ce type de dispositifs devient un tiers nécessaire qu'il s'agit d'authentifier au préalable. Une approche, à l'image du premier protocole que nous avons présenté, consiste à s'appuyer sur un tiers distant. Mais dans le cas d'une vérification d'objets sans télécommunication, l'authentification du dispositif matériel sera supervisée par l'agent vérifieur, un humain à qui revient au final la décision quant à l'authenticité de l'objet contrôlé. L'objectif poursuivi ici a été de faire jouer à l'humain un rôle plus actif que ce qu'il n'est aujourd'hui, en allant vers une authentification autonome d'objets c'est à dire une vérification préalable du dispositif permettant la vérification locale d'une série d'objets.

Pour cela, nous avons considéré pour dispositif, un système embarqué dont l'authentification peut être réduite à celle d'un composant configurable ou programmable (respectivement un FPGA ou un microcontroleur) que nous avons appelé composant d'authentification. Nous avons ainsi développé trois protocoles à partir de cette architecture. Les deux premiers supposent l'utilisation d'un outil pour pallier les capacités de mémorisation et de calcul d'un humain. Cet outil de calculs indépendant n'interagit pas directement avec le système à authentifier. C'est là un premier pas vers l'autonomie en authentification d'objets.

Le troisième protocole proposé vise une autonomie complète. Ce protocole est basé sur des concepts cryptographiques exécutables par un humain en un temps raisonnable. En terme de calcul, il implique des permutations et calculs de Xor, en terme de mémorisation, celle d'une permutation (clé) secrète à partager avec le système. Nous avons estimé le niveau de sécurité de ce protocole en spécifiant le choix des paramètres et évalué sa difficulté de mise en oeuvre par un humain.

Les protocoles que nous avons développés ici permettent effectivement de vérifier l'authenticité d'un système embarqué. Ils apportent une réponse à notre problématique de départ en traçabilité sécurisée. Ils pourraient aussi contribuer à la vérification auto-

nome de machines, dans des contextes tels que le vote électronique ou les transactions bancaires particulières, où l'humain doit s'assurer seul de leur bon fonctionnement.

Table des figures

Liste des tableaux

Publications de l'auteur

Conférences internationales

[IFA10] A. Idrissa, T. Fournel, and A. Aubert.
Secure embedded verification of print signatures. In *Proceedings of 8th International Workshop on Information Optics (WIO'09) - Journal of Physics : Conference Series*, page 012036. IOP Publishing, 2010.

[IAFF10] A. Idrissa, A. Aubert, T. Fournel, and V. Fischer.
Secure protocols for serverless remote product authentication. In *Proceedings of the 5th Workshop on Embedded Systems Security*, pages 11 :1–7. ACM, 2010.

[IAF12] A. Idrissa, A. Aubert, and T. Fournel.
Computer-assisted machine-to-human protocols for authentication of a ram-based embedded system. In *Proceedings of Mobile Multimedia/Image Processing, Security, and Applications 2012, SPIE 2012*, pages 84060U–1. SPIE, 2012.

[CPIF12] D. Coltuc, M. Petrovici, A. Idrissa, T. Fournel
Print signatures after convex shaping for reliable document authentication
To be published *in the proceedings of the EUSIPCO-2012*

Colloques sans actes :

- **Journées du projet Semba, Annecy, France, 2009**
  Abdourhamane Idrissa, Thierry Fournel, Alain Aubert
  Poster : Traçabilité sécurisée embarquée

- **Journées scientifiques du cluster ISLE, LYON, 2009**
  Abdourhamane Idrissa, Thierry Fournel, Alain Aubert
  Poster & Présentation oral : Traçabilité sécurisée embarquée

- **Journée de la recherche de l'école doctorale SIS 488, Saint-Etienne, France, 2010**
  Abdourhamane Idrissa, Thierry Fournel, Alain Aubert
  Poster : Protocole d'authentification d'objets à distance sans serveur

[IAFF10'] **Secure Protocols for Serverless Remote Product Authentication**
Abdourhamane Idrissa, Alain Aubert, Thierry Fournel, Viktor Fischer.
8-th International Workshop on Cryptographic Architectures Embedded in Reconfigurable Devices (CryptArchi'2010), Gif-sur-Yvette, France, 2010

Bibliographie

[Act10] Actel Corp. Fusion and extended temperature fusion fpga fabric user's guide. July 2010.

[Aus95] K. Austin. Data security arrangements for semiconductor programmable devices, February 1995. US Patent 5,388,157.

[Ban12] European Central Bank. Biannual information on euro banknote counterfeiting, January 2012.

[BC08] J. Bringer and H. Chabanne. An authentication protocol with encrypted biometric data. *Progress in Cryptology–AFRICACRYPT 2008*, 5023 :109–124, 2008.

[BCI+07] J. Bringer, H. Chabanne, M. Izabachène, D. Pointcheval, Q. Tang, and S. Zimmer. An application of the goldwasser-micali cryptosystem to biometric authentication. In *Proceedings of the 12th Australasian conference on Information security and privacy*, pages 96–106. Springer-Verlag, 2007.

[BDK+05] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith. Secure remote authentication using biometric data. *Advances in Cryptology–EUROCRYPT 2005*, 3494 :147–163, 2005.

[BET08] B. Badrignans, R. Elbaz, and L. Torres. Secure fpga configuration architecture preventing system downgrade. In *Proceedings of FPL 2008, International Conference on Field Programmable Logic and Applications*, pages 317–322. IEEE, 2008.

[BGB06] L. Bossuet, G. Gogniat, and W. Burleson. Dynamically configurable security for sram fpga bitstreams. *International Journal of Embedded Systems*, 2 :73–85, 2006.

[BP05] B. Blanchet and A. Podelski. Verification of cryptographic protocols : Tagging enforces termination. *Theoretical Computer Science*, 333 :67–90, 2005.

[BS11] B. Blanchet and B. Smyth. Proverif 1.85 : Automatic cryptographic protocol verifier, user manual and tutorial, March 2011.

[Bur07] W. Burns. Hardware token self enrollment process, November 2007. US Patent 7,302,703.

[CF06] Grumelard O. Chabaud F. Authentication : A model of human-machine authentication. *A common European language to identify security levels of authentication methods ENISA*, 2006.

[Com74] L. Comtet. *Advanced Combinatorics : The art of finite and infinite expansions.* D. Reidel Publishing Company, 1974.

[CSF+08] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet x. 509 public key infrastructure certificate and certificate revocation list (crl) profile. *RFC5280 (Proposed Standard)*, May 2008.

[DFM98] G.I. Davida, Y. Frankel, and B.J. Matt. On enabling secure applications through off-line biometric identification. In *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, pages 148–157. IEEE, 1998.

[Dif88] W. Diffie. The first ten years of public-key cryptography. *Proceedings of the IEEE*, 76 :560–577, 1988.

[Dri07] S. Drimer. Authentication of fpga bitstreams : Why and how. *Reconfigurable Computing : Architectures, Tools and Applications*, pages 73–84, 2007.

[Dri09] S. Drimer. Security for volatile FPGAs. Technical Report UCAM-CL-TR-763, University of Cambridge, Computer Laboratory, November 2009.

[DRT07] J.G. Dumas, J.L. Roch, and E. Tannier. *Théorie des codes : Compression, cryptage, correction.* Sciences sup. Dunod, 2007.

[DY81] D. Dolev and A. C. Yao. On the security of public key protocols. *Annual IEEE Symposium on Foundations of Computer Science*, 0 :350–357, 1981.

[FS87] A. Fiat and A. Shamir. How to prove yourself : practical solutions to identification and signature problems. In *Proceedings on Advances in CryptologyCrypto'86*, pages 186–194. Springer, 1987.

[Gas03] B.L.P. Gassend. *Physical random functions.* PhD thesis, Massachusetts Institute of Technology, 2003.

[GCVDD02] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas. Silicon physical random functions. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 148–160. ACM, 2002.

[GGM85] O. Goldreich, S. Goldwasser, and S. Micali. On the cryptographic applications of random functions. In *Proceedings on Advances in Cryptology '85*, pages 276–288. Springer, 1985.

[GKD09] R. Gallo, H. Kawakami, and R. Dahab. On device identity establishment and verification. In *Proceedings of the 6th European conference on Public key infrastructures, services and applications*, pages 130–145. Springer-Verlag, 2009.

[GKST07] J. Guajardo, S. Kumar, G.J. Schrijen, and P. Tuyls. Fpga intrinsic pufs and their use for ip protection. *Cryptographic Hardware and Embedded Systems-CHES 2007*, pages 63–80, 2007.

[GQ88] L. Guillou and J.J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In *Proceedings on Advances in Cryptology-EUROCRYPT '88*, pages 123–128. Springer, 1988.

[HHJ+06] J.H. Hoepman, E. Hubbers, B. Jacobs, M. Oostdijk, and R. Schreur. Crossing borders : Security and privacy issues of the european e-passport. *Advances in Information and Computer Security*, pages 152–167, 2006.

[IAF12] A. Idrissa, A. Aubert, and T. Fournel. Computer-assisted machine-to-human protocols for authentication of a ram-based embedded system. In *Proceedings of Mobile Multimedia/Image Processing, Security, and Applications 2012, SPIE 2012*, pages 84060U–1. SPIE, 2012.

[IAFF10] A. Idrissa, A. Aubert, T. Fournel, and V. Fischer. Secure protocols for serverless remote product authentication. In *Proceedings of the 5th Workshop on Embedded Systems Security*, pages 11 :1–7. ACM, 2010.

[ICA06] ICAO. 9303-machine readable travel documents-part 1-2. Technical report, Technical report, International Civil Aviation Organization, 2006.

[IFA10] A. Idrissa, T. Fournel, and A. Aubert. Secure embedded verification of print signatures. In *Proceedings of 8th International Workshop on Information Optics (WIO'09) - Journal of Physics : Conference Series*, page 012036. IOP Publishing, 2010.

[JRP04] AK Jain, A. Ross, and S. Prabhakar. An introduction to biometric recognition. *IEEE Transactions on circuits and systems for video technology*, 14 :4–20, 2004.

[Kea01] T. Kean. Secure configuration of field programmable gate arrays. In *Proceedings of Field-Programmable Logic and Applications*, pages 142–151. Springer, 2001.

[KGM+08] S.S. Kumar, J. Guajardo, R. Maes, G.J. Schrijen, and P. Tuyls. The butterfly puf protecting ip on every fpga. In *Proceedings of the 2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 67–70. IEEE, 2008.

[LGF09] J. Lancrenon, R. Gillard, and T. Fournel. Remote object authentication : confidence model, cryptosystem and protocol. In *Proceedings of SPIE 2009, the International Society for Optical Engineering*, page 73440J. SPIE, 2009.

[LGF11] J. Lancrenon, R. Gillard, and T. Fournel. Remote object authentication against counterfeiting using elliptic curves. In *Proceedings of SPIE 2011, the International Society for Optical Engineering*, page 80630B. SPIE, 2011.

[LLG+05] D. Lim, J.W. Lee, B. Gassend, G.E. Suh, M. Van Dijk, and S. Devadas. Extracting secret keys from integrated circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13 :1200–1205, 2005.

[Max09] Maxim Integrated Products, Inc. *ZA9L1 : Zatara High-Performance, Secure, 32-Bit ARM Microcontroller*. maxim-ic, March 2009.

[MRLW01] F. Monrose, M.K. Reiter, Q. Li, and S. Wetzel. Cryptographic key generation from voice. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, pages 202–213. IEEE, 2001.

[MVOV96] A.J. Menezes, P.C. Van Oorschot, and S.A. Vanstone. *Handbook of applied cryptography*. CRC Press, Inc., 1996.

[PG06] M.M. Parelkar and K. Gaj. Implementation of eax mode of operation for fpga bitstream encryption and authentication. In *Proceedings of the 2005 IEEE International Conference on Field-Programmable Technology*, pages 335–336. IEEE, 2006.

[PRTG02] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld. Physical one-way functions. *Science*, 297 :2026–2030, 2002.

[PSA11] S.B. Pollard, S.J. Simske, and G.B. Adams. Model based print signature profile extraction for forensic analysis of individual text glyphs. In *Proceedings of the 2010 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2011.

[SA10] S.J. Simske and G. Adams. High-resolution glyph-inspection based security system. In *Proceedings of the 2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 1794–1797. IEEE, 2010.

[Sch91] C.P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3) :161–174, 1991.

[SD07] G.E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th annual Design Automation Conference*, pages 9–14. ACM, 2007.

[ST96] C. Soutar and GJ Tomko. Secure private key generation using a fingerprint. In *Proceedings of CardTech/SecurTech '96 : applications in action*, pages 245–252. Rockville, Md. : CardTech/SecurTech Inc, 1996.

[ŠTO05] B. Škorić, P. Tuyls, and W. Ophey. Robust key extraction from physical uncloneable functions. In *Proceedings of the Applied Cryptography and Network Security Conference 2005*, pages 99–135. Springer Berlin, 2005.

[SV96] B. Schneier and L. Viennot. *Cryptographie appliquée : Algorithmes, protocoles et codes source en C. 2eme édition.* Vuibert, 1996.

[WGP04] T. Wollinger, J. Guajardo, and C. Paar. Security on fpgas : State-of-the-art implementations and attacks. *ACM Transactions on Embedded Computing Systems (TECS)*, 3 :534–574, 2004.

[Xil10] Xilinx Inc. *UG360 : Virtex-6 FPGA Configuration user guide.* Xilinx Inc., November 2010.

[ZWK03] B. Zhu, J. Wu, and M.S. Kankanhalli. Print signatures for document authentication. In *Proceedings of the 10th ACM conference on Computer and communications security*, pages 145–154. ACM, 2003.