Broadcast Encryption with Traitor Tracing

Mario Strefler

Diffusion chiffrée avec traçage de traîtres

Thèse de Doctorat

en vue de l'obtention du grade de

Docteur de l'École Normale Supérieure
(spécialité informatique)

présentée et soutenue publiquement le 26 septembre 2013 par

Mario Strefler

devant le jury composé de :

Directeur de thèse :	David Pointcheval	(CNRS, École Normale Supérieure)
Rapporteurs :	Marc Fischlin	(Technische Universität Darmstadt)
	Marc Joye	(Technicolor)
Examinateurs :	Dennis Hofheinz	(Karlsruher Institut für Technologie)
	Fabien Laguillaumie	(Université Lyon 1)
	David Naccache	(Univ. Paris II, École Normale Supérieure)
	Kenneth G. Paterson	(Royal Holloway, University of London)
	Duong Hieu Phan	(Université Paris 8)

École doctorale 386 : Sciences mathématiques de Paris Centre
Unité de recherche : UMR 8548 — Département d'Informatique de l'École Normale Supérieure
Laboratoire de recherche affilié au CNRS et à INRIA

Remerciements

Je souhaite d'abord exprimer ma reconnaissance et ma profonde gratitude envers mon directeur de thèse, David Pointcheval, qui m'a accompagné et soutenu pendant l'achèvement de cette travail. Grand merci pour ta disponibilité, ta bienveillance et ta patience. J'ai vraiment beaucoup appris sous ta direction.

Mes remerciements les plus vifs vont également à Duong Hieu Phan pour ses excellents idées et conseils pendant ces quatre années.

I would also like to express my gratitute towards my other co-authors, Jean Paul Degabriele, Anja Lehmann, Kenneth G. Paterson, Siamak F. Shahandashti, and Nigel P. Smart.

Je suis très reconnaissant aux rapporteurs, Marc Fischlin et Marc Joye, qui ont accepté de relire ce manuscrit, ainsi qu'aux autres membres du jury, Dennis Hofheinz, Fabien Laguillaumie, David Naccache, Kenneth G. Paterson et Duong Hieu Phan. Mes sinceres remerciements vont à Nuttapong Attrapadung, Duong Hieu Phan, Elizabeth Quaglia et Damien Vergnaud pour leur lecture et leurs corrections d'une partie de cette thése.

I would also like to thank Mihir Bellare for hosting me at UCSD, Sriram Keelveedhi, Petros Mol, and the other members of the CSE department for being welcoming, and Björn Tackmann for the time spent on an ultimately fruitless but very instructive cooperation.

Je remercie toute l'équipe crypto de l'ENS pour l'environnement très agréable, et plus particulièrement Michel Abdalla et Damien Vergnaud pour les discussions, ainsi que les autres permanents : Bruno Blanchet, Pierre-Alain Fouque, Vadim Lyubashevsky, Phong Nguyen et Oded Regev.

Je voudrais remercier spécialement les « jeunes chercheurs » : ceux qui sont partis avant moi, comme Aurélie Bauer, fondatrice des « séminaires jeunes », Olivier Blazy, Charles Bouillaget, David Cadé, Angelo De Caro, Dario Fiore, Miriam Paiola et Mehdi Tibouchi, ainsi que ceux qui sont restés — Yuan-Mi Chen, Patrick Derbez et Jérémy Jean, Leo Ducas, Aurore Guillevic, Elizabeth Quaglia et Damien Vergnaud — et enfin les nouveaux arrivants : Sonia Belaïd, Fabrice Ben Hamouda, Sorina Ionica, Tancrède Lepoint, Thomas Prest et Sylvain Ruhault.

Mes remerciements vont aussi à ceux que je n'ai rencontrés que pour une courte durée : Orr Dunkelman pour son aide avec mon téléphone portable, Céline Chevalier et Malika Izabachène pour leur accueil, et aux collégues des premières années, Georg Fuchsbauer, Gaëtan Leurent et Roch Lescuyer.

Je tiens à remercier le personnel administratif de l'ENS et d'INRIA ainsi que le SPI pour son efficacité, en particulier Joëlle Isnard pour sa disponibilité, Valérie Mongiat pour son aide sympathique, Nathalie Gaudechoux pour son investissement et Michelle Angely et Lise-Marie Bivard pour leur amabilité.

Merci à l'INRIA et à l'ANR pour le financement du projet ANR-09-VERS-016 BEST, dans le cadre duquel cette thèse a été réalisée. Merci également aux membres du projet : Renaud Dubois de Thales, Pascal Paillier et Cecile Delerablée de CryptoExperts, et Bevan Régis et Alexandre Karlov de Nagra pour les réunions instructives.

For the end of these acknowledgements, I would like to return to the beginning of my academic life and express my deep and sincere gratitude to Jörn Müller-Quade for his invaluable support and encouragement and for supervising my *Diplomarbeit* together with the ever helpful Daniel Kraschewski. I am deeply indebted to Hideki Imai and Hajime Watanabe for welcoming me at the RCIS and Kazukuni Kobara and SeongHan Shin for their patience and supervision in coauthoring my first paper, and to Hubert Comon-Lundh for encouraging me to apply at ENS.

Mein besonderer Dank gilt natürlich meinen Eltern und meiner Schwester, die mich stets unterstützt haben und mir zur Seite standen und ohne die ich nicht der wäre, der ich bin.

Table des matières

Prolégomènes

Contents

Avant de nous plonger dans les questions techniques, nous voulons réfléchir brièvement sur l'objet d'un mémoire de thèse. La raison d'écrire cette thèse est bien sûr qu'il s'agit d'une exigence pour l'obtention d'un diplôme universitaire, qui est une condition préalable pour travailler en tant que chercheur dans un domaine de recherche fascinant. Le public cible de ce mémoire est en premier les deux rapporteurs, qui attendent de la thèse qu'elle présente des résultats, déjà publiés, de façon cohérente. Il est donc important de motiver les questions ou problèmes ouverts qui ont conduit à ces résultats, de décrire comment les résultats améliorent l'état de l'art, mais aussi de montrer un certain niveau de maîtrise des outils du métier, de la terminologie et des techniques qui sont au cœur de ce domaine et qui permettent à un jeune chercheur d'identifier et de travailler indépendamment sur des problèmes ouverts.

Si cela était le seul objectif d'une thèse, il serait inutile de la publier et de la préserver dans une bibliothèque après la soutenance. Une thèse, bien que n'étant pas un manuel scolaire, devrait également être un travail qui peut se suffire à lui-même et servir de point de départ pour la recherche d'informations sur un sujet. Nous visons à rendre ce travail accessible aux lecteurs des niveaux de compétence différents. Nous commençons donc par une présentation générale du développement de la cryptologie moderne, en nous concentrant sur les découvertes que nous considérons être le plus rattachées à cette thèse. Ensuite, nous nous orientons vers des problèmes spécifiques à la protection de contenu, avant de détailler nos contributions le plus clairement possible.

Une brève histoire de la cryptologie moderne

La première contribution à la cryptologie moderne a été publiée en 1949 par Shannon, qui a défini la *sécurité parfaite* d'un système de chiffrement symétrique en termes issus de la théorie de l'information : un texte chiffré ne révèle rien sur le message qu'il contient (à l'exception de sa longueur). En d'autres termes, même un adversaire avec une puissance calculatoire illimitée n'apprend rien de plus sur un message en voyant son chiffrement qu'il n'aurait également appris à partir d'une chaîne aléatoire de même longueur. Shannon a également fourni une condition nécessaire pour atteindre la sécurité parfaite : l'espace dans lequel est choisie la clé doit être au moins aussi grand que l'espace des messages, ce qui signifie en pratique que les clés doivent être au moins aussi longues que le message qu'elles servent à chiffrer. Cette idée de définir la sécurité et de prouver qu'un système satisfait cette définition est une des idées fondamentales en cryptologie.

L'ère de la cryptologie moderne commença véritablement dans les années 1970, avec le processus de normalisation publique du chiffrement par bloc DES aux États-Unis, développé par IBM avec quelques modifications de l'agence de sécurité nationale NSA. Ce processus ouvert suivait un principe énoncé en 1883 par Kerckhoffs, que la sécurité d'un système cryptographique ne doit pas dépendre du secret

de sa conception. La conception ouverte permet un examen public du système, ce qui a contribué au développement de la cryptologie en tant que science.

La deuxième révolution de cette décennie fut la découverte de la cryptographie à clé publique. Merkle développait en 1974 un protocole d'échange de clés connu sous le nom de « puzzles de Merkle », qui permet à deux utilisateurs de s'entendre sur un secret partagé sur un canal publique, en utilisant un algorithme de chiffrement symétrique avec un espace de clés contenant κ clés pour un entier κ. L'expéditeur chiffre κ chaînes aléatoires, précédées par une chaîne publique constante, en utilisant chacune des κ clés, et envoie ces messages au récepteur. Ce préfixe constant est une première tentative de *robustesse*, la propriété qui permet à un utilisateur de savoir quand il a utilisé une mauvaise clé pour déchiffrer. Le récepteur choisit l'un de ces puzzles au hasard et tente de le déchiffrer avec des clés choisies au hasard jusqu'à ce que le message aléatoire commence par le préfixe constant connu. Il renvoie alors le message aléatoire, et la clé qu'il a utilisée pour déchiffrer est alors le secret commun entre les deux. L'expéditeur et le destinataire doivent effectuer de l'ordre de κ opérations, mais un adversaire doit résoudre par force brute en moyenne $\kappa/2$ puzzles, ce qui prend de l'ordre de $\kappa/2$ opérations, donc sa charge de calcul est de l'ordre de κ^2. Cela signifie que, pour que le protocole soit sûr, il est nécessaire que l'émetteur et le récepteur aient tous les deux environ les mêmes ressources informatiques que l'attaquant, sinon κ doit être choisi très grand.

Le protocole présenté en 1976 par Diffie et Hellman surmonte ce problème en s'appuyant sur une fonction à sens unique, c'est-à-dire une fonction qui est facile à calculer et difficile à inverser, avec des propriétés spéciales. Il existe des algorithmes efficaces pour calculer l'exponentiation discrète, la valeur g^a pour un élément g dans un groupe fini $\mathbb{G}$, tel que $\mathbb{Z}_p^*$ pour un nombre premier p, et un certain scalaire a. Pour l'opération inverse, le calcul de a, étant donné g et g^a, aucun algorithme efficace n'est connu. Diffie et Hellman ont proposé le protocole suivant entre deux parties A et B. A choisit un scalaire aléatoire a, calcule g^a et envoie cette valeur à B, qui choisit également un scalaire aléatoire b, calcule g^b et envoie cette valeur à A. Les deux parties élèvent la valeur reçue à la puissance de leur scalaire secret et obtiennent $(g^a)^b = g^{ab} = (g^b)^a$, le secret partagé. Ici, la sécurité du système repose sur la difficulté d'un problème de théorie des nombres. La différence entre la demande de calcul pour les utilisateurs et pour l'adversaire est bien plus importante que celle pour les puzzles de Merkle, parce que le nombre d'étapes nécessaires pour briser le protocole augmente non pas en κ^2, mais plus vite que tout polynôme en κ.

En 1977, Rivest, Shamir et Adleman présentent le premier système de chiffrement à clé publique, maintenant connu sous le nom RSA, qui utilise des groupes d'ordre composé et est lié à un problème différent de théorie de nombres, la factorisation d'entiers. Un utilisateur choisit deux grands nombres premiers p et q et calcule $n = pq$, $\varphi(n) = (p-1)(q-1)$, et deux exposants e et d tels que $de \equiv 1 \mod \varphi(n)$. Dans le groupe multiplicatif $\mathbb{Z}_n^*$, qui est d'ordre $\varphi(n)$, nous avons $x^{ed} \equiv x^1 = x$. Cela signifie que, ayant publié e et n, n'importe qui peut chiffrer un message m en $c \equiv m^e \mod n$, mais l'exposant secret d est nécessaire pour déchiffrer.

Ce système simple est déterministe, ce qui signifie qu'en chiffrant des messages candidats et en comparant leurs chiffrés à un texte chiffré donné, il peut être exclu que le texte clair correspond à l'un de ces messages. Cela indique clairement que la sécurité doit être définie de manière formelle. La première définition est la *sécurité sémantique*, un analogue de la sécurité parfaite dans le sens de la théorie de l'information pour des adversaires avec une puissance calculatoire limitée, qui a été proposée par Goldwasser et Micali en 1982. Ils présentent ensuite une notion équivalente, qui est plus facile à utiliser dans les preuves de sécurité, l'*indistingabilité sous des attaques à clairs choisis* (IND-CPA, pour *Indistinguishability against Chosen Plaintext Attacks*). IND-CPA est décrit sous la forme d'une expérience : un adversaire (avec temps d'exécution polynomialement borné) reçoit la clé publique, puis il choisit deux messages de même taille ; l'un d'eux est choisi au hasard, chiffré avec la clé publique, et le chiffré ainsi obtenu est donné à l'adversaire. L'adversaire gagne l'expérience s'il peut décider lequel des deux messages a été chiffré.

En 1990, Naor et Yung définissent l'*indistingabilité sous des attaques à chiffrés choisis* (plus tard nommée IND-CCA1) pour les systèmes de chiffrement à clé publique, où l'adversaire a accès à un oracle de déchiffrement pour l'aider à trouver deux messages dont les chiffrés sont faciles à distinguer.

En 1991, Rackoff et Simon donnent une définition encore plus forte, l'*indistingabilité sous des attaques adaptatifs à chiffrés choisis* (IND-CCA2), où l'adversaire a accès à l'oracle de déchiffrement même après qu'il a reçu le chiffré défi, avec la restriction qu'il ne demande pas le déchiffrement du défi.

Protection de contenu

Les systèmes de chiffrement à clé publique (PKE, pour *Public-Key Encryption*) sont utilisés pour assurer la confidentialité pour des communications *unicast* ou point-à-point comme le courrier électro-

nique (PGP) ou la navigation web (TLS). Pour la communication un-à-plusieurs utilisant des canaux *multicast* comme des listes de diffusion ou de diffusion comme la radio, une approche différente est nécessaire. Utiliser un système PKE pour chiffrer un message à destination d'un groupe d'utilisateurs devient rapidement impraticable. Lorsque le groupe change rarement, et que tous les utilisateurs de ce groupe doivent toujours être en mesure de déchiffrer, une bonne option peut être de fournir une clé commune de déchiffrement à tous les utilisateurs. Nous nous intéressons en revanche à un cas plus délicat : la *diffusion chiffrée* (BE, pour *Broadcast Encryption*), qui est une généralisation du chiffrement symétrique ou asymétrique au scénario un-à-plusieurs, où un expéditeur peut choisir au moment du chiffrement l'ensemble des utilisateurs qui pourront déchiffrer le message diffusé. Dans le monde réel, il y a plusieurs façons de transmettre un message à plusieurs utilisateurs simultanément. Le protocole IPv4 permet l'utilisation des adresses de diffusion [Mog84]. Si un message (datagramme) est envoyé à cette adresse du réseau, la passerelle réceptrice le diffuse à tous les nœuds de ce réseau. Les protocoles IPv4 et IPv6 permettent l'utilisation des adresses *multicast* [CVM10]. Ces adresses sont attribuées à des groupes d'utilisateurs, et un message envoyé à une adresse multicast est transmis à tous les membres du groupe. Cela nécessite une gestion de groupe pour assurer qu'il n'y ait pas d'ambiguïté à propos de l'appartenance au groupe. Il existe aussi la possibilité de diffusion physique par un réseau de radio ou de satellites. Toutes ces options ont des caractéristiques différentes.

Une application typique pour la diffusion chiffrée est la protection du contenu, que ce soit pour la télévision à péage, où un fournisseur veut s'assurer que seuls les utilisateurs qui ont payé leur abonnement peuvent visualiser le contenu qui est diffusé, ou pour la gestion des droits numériques (DRM, pour *Digital Rights Management*), où seuls les décodeurs qui obéissent à certaines politiques dans le processus de fabrication devraient être capables de déchiffrer. Dans la pratique, BE est utilisé par exemple par des mécanismes de protection des droits d'auteur pour les médias numériques comme les DVD, afin d'assurer que seuls les lecteurs DVD qui sont conformes à une norme peuvent déchiffrer et traiter le contenu. Si les clés pour une série de lecteurs de DVD sont connues, cette série ne sera pas capable de lire des DVD fabriqués après que la série est révoquée [NNL01].

La diffusion chiffrée s'intéresse uniquement à la confidentialité face aux utilisateurs révoqués, mais elle ne peut empêcher les utilisateurs autorisés de distribuer le contenu déchiffré ou de mettre leurs capacités de déchiffrement à la disposition des autres. Ceci est le domaine du *traçage de traîtres* (TT).

Diffusion chiffrée

Nous pensons qu'il est important de différencier les systèmes de diffusion chiffrée de la tâche de diffusion chiffrée. La tâche de diffusion chiffrée est de chiffrer un message pour un ensemble de récepteurs qui peuvent être différents pour chaque texte chiffré.

Les systèmes de diffusion chiffrée (BE) [FN94] permettent à l'expéditeur d'un message de déterminer un sous-ensemble d'utilisateurs (l'*ensemble des privilégiés*) à destination de qui le message est chiffré. On appelle le complément de l'ensemble des privilégiés l'*ensemble des révoqués*. Le complément des systèmes BE sont les *systèmes de révocation*, qui définissent l'ensemble des révoqués directement. Les deux ne font aucune hypothèse sur l'ensemble des utilisateurs. Si l'ensemble des utilisateurs a une certaine structure, il peut être plus convenable d'utiliser un système de chiffrement basé sur des attributs.

Nous nous intéressons principalement aux systèmes de diffusion chiffrée et aux systèmes de révocation, et nous ne les distinguons pas lorsque nous parlons de systèmes statiques, où l'ensemble des utilisateurs est fixé. Dans le cas dynamique, où les utilisateurs peuvent rejoindre le système après que des messages ont été chiffrés, il y a une différence entre BE et systèmes de révocation. Pour un système BE dynamique, les utilisateurs qui se joignent après l'envoi d'un message sont dans l'ensemble des révoqués, tandis que pour un système de révocation les nouveaux utilisateurs sont dans l'ensemble des privilégiés.

Une émission chiffrée se compose de trois parties : l'*en-tête identifiant* est une chaîne de bits qui identifie sans ambiguïté l'ensemble des privilégiés. Tous les systèmes BE connus sont des systèmes hybrides, où il y a une *en-tête de clé*, qui distribue une clé de session exactement aux utilisateurs privilégiés, et le *corps du message*, qui contient un chiffrement du message sous la clé de session.

Il faut N bits pour identifier un sous-ensemble d'un ensemble de taille N. Si le nombre r de révoqués est petit, il suffit d'identifier uniquement les utilisateurs révoqués. Cela peut être fait en utilisant $r \log N$ bits. La même technique s'applique au cas où l'ensemble des privilégiés est petit. Si les récepteurs maintiennent un état et la différence entre les deux ensembles des privilégies successifs est petite, la taille de l'en-tête identifiant peut être réduite en ne transmettant que cette différence. Parce que la technique utilisée ne dépend pas du système BE, la taille de l'en-tête identifiant n'est généralement pas incluse dans le calcul de la taille du chiffré lorsqu'on compare l'efficacité des systèmes. On ne prend pas en compte non plus

la taille du corps du message, qui également ne dépend pas du système BE. En général, les systèmes de diffusion chiffrée n'abordent que la transmission de l'en-tête de clé, et sont considérés comme des *mécanismes d'encapsulation de clés* (KEM, pour *Key Encapsulation Mechanism*) [CS03].

Il y a deux systèmes BE triviaux. Dans le premier, le centre partage une clé symétrique (ou une paire de clés) avec chaque utilisateur. L'en-tête de clé se compose d'un chiffrement de la clé de session pour chaque clé partagée avec un utilisateur privilégié. Ce système permet d'obtenir un texte chiffré de taille linéaire en $N - r$, avec un stockage de l'utilisateur constant, et un temps de déchiffrement constant. Le second est le système qui assigne une clé à chaque sous-ensemble de l'ensemble d'utilisateurs. Dans ce système, l'en-tête de clé est vide, comme l'en-tête identifiant définit déjà la clé de session. Par conséquent, la longueur de l'en-tête de clé est de 0, le stockage chez l'utilisateur est de $\mathcal{O}(2^N)$, et le temps de déchiffrement est constant. L'objectif est maintenant de trouver un système qui améliore ces deux systèmes ou qui fournit un meilleur compromis. La valeur la plus critique pour l'évaluation des systèmes est la longueur de l'en-tête de clé, car la bande passante est généralement le facteur limitant. Comme l'expéditeur ne peut être supposé avec beaucoup de mémoire ni une grande puissance de calcul, les autres variables sont le stockage et la complexité des calculs du récepteur.

Le scénario

Notre scénario se compose de N utilisateurs et un utilisateur spécial nommé le *centre* C. Tous les utilisateurs sont connectés par un réseau, et en plus par un canal de diffusion. Dans le scénario de base, le réseau est considéré non-sûr, ce qui signifie que tous les messages envoyés sur le réseau sont d'abord envoyés à l'adversaire $\mathcal{A}$, qui peut choisir de ne pas les transmettre, de les modifier avant de les retransmettre, de les envoyer à d'autres utilisateurs qu'à ceux initialement prévus, de modifier l'origine apparente, d'introduire des messages complètement nouveaux ou de réorganiser les messages. Dans le modèle des *canaux authentifiés* [BCK98], l'adversaire peut lire tous les messages envoyés sur le réseau et les supprimer ou les réorganiser, mais il ne peut pas les modifier ou les réacheminer, ni introduire de nouveaux messages. Cela modélise des messages qui sont authentifiés parfaitement. Nous distinguons deux types de canaux de diffusion : la diffusion par réseau et la diffusion physique. Lorsque les messages sont envoyés sur un *canal de diffusion réseau*, l'adversaire a les mêmes pouvoirs que pour les canaux non sécurisés décrits ci-dessus. Nous définissons des *canaux de diffusion authentifiés* comme ci-dessus. Par ailleurs, un canal de diffusion authentifié modélise également une diffusion physique peu fiable, où les utilisateurs risquent de ne pas recevoir des messages parce qu'ils sont hors de portée, et où les messages peuvent être réorganisées en raison de différences dans la vitesse de propagation causées par la diffraction, la réfraction, etc. Lorsque les messages sont envoyés sur un *canal de diffusion physique* (fiable), l'adversaire peut lire tous les messages, mais il est complètement passif. Cela modélise par exemple les émissions par satellite.

En plus de sa puissance sur le réseau, l'adversaire peut corrompre des utilisateurs (mais pas le centre). Lorsque cela arrive, leur état complet est révélé à l'adversaire, et leur comportement est contrôlé par lui à partir de ce moment. Dans le modèle de *corruption statique*, l'adversaire ne peut corrompre des utilisateurs qu'avant l'exécution du protocole. Dans le modèle de *corruption adaptative*, il peut corrompre des utilisateurs à tout moment au cours de l'exécution du protocole.

La généalogie

Il y a plusieurs directions de recherche qui traitent de problèmes connexes. L'*échange de clés au sein de groupes* traite le problème de l'établissement d'un secret partagé entre n'importe quel sous-groupe de N utilisateurs qui ont une clé appropriée sur un réseau de canaux authentifiés. En général, on suppose qu'une phase d'initialisation a eu lieu avant qu'un message ne soit envoyé. Une extension de ce problème est l'*échange authentifié de clés au sein de groupes* où l'adversaire a le contrôle total du réseau. Sans être formel, nous appellerons un protocole d'échange de clés un *mécanisme de transports de clés* quand un utilisateur choisit seul la clé de session.

Le cas particulier des *mécanismes de transport de clé au sein de groupes* est très proche de notre objectif, où un seul émetteur établit une clé commune par l'envoi d'un message à chaque utilisateur. On peut donc dire que la tâche de diffusion chiffrée est accomplie par des mécanismes de transport de clé au sein de groupes sans interaction, où l'émetteur diffuse un seul message à tous les utilisateurs. La principale différence est au niveau de l'exigence sur le changement de clé de session : contrairement à la diffusion chiffrée, une session protégée par un système de transport de clé peut comprendre un échange de messages ; il est donc raisonnable de s'attendre à ce que plusieurs messages soient chiffrés ou authentifiés avec une seule clé. L'autre différence réside dans les modèles de sécurité. Alors que pour les protocoles

de transport de clés, un modèle interactif est utilisé, centré sur la puissance de l'adversaire sur le réseau et les utilisateurs, la diffusion chiffrée est évaluée dans un modèle dérivé de celui pour le chiffrement à clé publique (qui comprend habituellement également la corruption des utilisateurs).

Nous nous intéressons particulièrement aux *systèmes de diffusion chiffrée sans états*, où les calculs ne dépendent que de l'état initial et du chiffré actuellement reçu. En général, les protocoles qui nécessitent des utilisateurs avec état peuvent être transformés sans état en envoyant l'information complète de la mise à jour d'état au début de chaque message [NNL01], même si cela peut être inefficace. Mais plus intéressants sont les protocoles qui ne nécessitent pas que les utilisateurs soient toujours en ligne, par exemple, pour recevoir des messages de mise à jour, ce qui est toujours le cas pour les utilisateurs sans états.

L'*échange de clés au sein de groupes dynamiques* permet des changements de l'ensemble des utilisateurs via des protocoles *Join* et *Leave*, gérés par un *gestionnaire de groupe*. L'analogue sur ce point dans le monde BE est la *diffusion chiffrée dynamique* [DPP07].

Un système où la clé de session est toujours partagée par le groupe d'utilisateurs et mise à jour lors des modifications de la composition du groupe, est généralement connu sous le nom de *chiffrement multicast* [WHA99, WGL00]. Ce problème est également connu comme *re-keying* et exige naturellement de la sécurité dans le temps, ce qui exclut un utilisateur d'avoir des informations sur le clé actuelle après qu'il a quitté le groupe, et un utilisateur nouveau d'avoir des informations sur des clés passées.

Plusieurs systèmes existent qui impliquent la diffusion chiffrée : la diffusion chiffrée peut être construite à partir du *chiffrement basé sur des attributs* (ABE, pour *Attribute-Based Encryption*) [SW05, GPSW06] avec des techniques spécifiques en fonction de l'expressivité de la politique de sécurité et selon si elle est associée à la clé ou au chiffré. Dans le *chiffrement de vecteur caché* (HVE, pour *Hidden Vector Encryption*) [BW07, IP08], des chiffrés sont associés à des vecteurs binaires, et les clés sont associées à des vecteurs de politique de sécurité qui peuvent contenir des jokers ($*$). Si la politique est plutôt associée au texte chiffré, il est possible d'émuler un BE en utilisant un vecteur de politique de sécurité de longueur N et d'identifier des utilisateurs à une position i dans le vecteur. Ensuite, chaque utilisateur i reçoit une clé avec un identifiant qui a un 1 à la position i et 0 partout ailleurs. Le vecteur de la politique de sécurité contient un 0 à chaque position correspondant à un utilisateur en dehors de l'ensemble cible, et un $*$ ailleurs. Puisque le vecteur chiffré est caché même pour les utilisateurs qui peuvent déchiffrer, HVE implique même la diffusion chiffrée *anonyme*, où aucun utilisateur n'apprend rien sur l'ensemble des privilégiés, sauf s'il en fait partie ou pas. Le *chiffrement basé sur l'identité avec jokers* (WIBE, pour *Wildcarded Identity-Based Encryption*) est une primitive où les utilisateurs sont liés aux vecteurs d'identité et les chiffrés contiennent un vecteur des identités qui peut contenir des jokers $*$ [ACD+06]. Un utilisateur peut déchiffrer si son vecteur d'identité correspond au vecteur d'identité du chiffré dans toutes les positions qui ne sont pas $*$. WIBE peut simuler HVE (sans l'anonymat) à l'aide d'une profondeur de la hiérarchie égale à la longueur du vecteur, et par conséquent il implique également le diffusion chiffrée.

Traçage de traîtres

Bien que la diffusion chiffrée puisse être vue comme une généralisation du PKE au scénario un-à-plusieurs, le traçage de traîtres s'intéresse à un problème véritablement différent et nouveau. Nous examinons d'abord le scénario dans lequel un centre chiffre un message à tous les utilisateurs. Nous voulons nous protéger contre les utilisateurs partageant leurs clés de déchiffrement secrètes avec des personnes extérieures au groupe. Les utilisateurs qui agissent de cette manière sont appelés *traîtres*. Comme il n'existe aucun moyen pour empêcher les utilisateurs de partager leurs clés secrètes ou des algorithmes qui contiennent ces clés, l'idée est de permettre l'identification d'un traître à partir de sa clé de déchiffrement, ce qui dissuadera les utilisateurs de partager leurs clés.

Traçage de traîtres

Les systèmes *traçage de traîtres* (TT) [CFN94] permettent à l'autorité de traçage de découvrir des utilisateurs qui ont contribué, avec leurs clés secrètes, à la construction d'un décodeur pirate, soit un algorithme qui peut déchiffrer. Un système TT est appelée *à clé secrète*, si la clé de chiffrement doit être gardée secrète, et *à clé publique* si n'importe qui peut chiffrer. Si n'importe quel utilisateur peut tracer, le système est dit *traçable publiquement* [CPP05]. Des systèmes de diffusion chiffrée qui permettent le traçage de traîtres (ou des systèmes de traçage de traîtres qui permettent spécifier dynamiquement les ensembles d'utilisateurs) sont appelés *traçage-et-révocation* (TR) [NP01]. Les systèmes TR permettent au centre de continuer le traçage après qu'un pirate a été découvert et que ses clés ont été désactivées. Ils

doivent faire face à l'évolution des décodeurs [KP07], qui tentent de tirer le meilleur parti de leurs clés et de générer autant de décodeurs que possible après que la génération précédente a été rendue inutile.

L'adversaire dans notre scénario est passif et peut corrompre des utilisateurs. Il peut alors utiliser leurs clés pour construire un *décodeur pirate*, c'est-à-dire un algorithme qui peut déchiffrer des messages chiffrés. L'algorithme de traçage du système TT a accès au décodeur pirate et doit trouver au moins un des utilisateurs qui ont contribué à la construction du décodeur. Selon le type d'accès de l'algorithme de traçage au décodeur pirate, nous distinguons le *traçage en boite blanche*, où l'algorithme de traçage peut extraire les clés du décodeur, le *traçage en boite noire*, où l'algorithme de traçage peut seulement interroger le décodeur, et le *traçage par accès minimal*, où le traceur apprend seulement si le message a été déchiffré correctement. Usuellement, le décodeur pirate est supposé *sans état*, ce qui modélise un décodeur hardware qui peut être remis à zéro ou un décodeur logiciel dont une nouvelle copie est utilisée pour chaque requête. Un *décodeur avec état* peut en revanche enregistrer ses interactions passées, et est donc beaucoup plus difficile à tracer.

Un paradigme dans la construction de systèmes de traçage des traîtres est d'utiliser un système qui peut adresser des sous-ensembles de l'ensemble des utilisateurs, mais où un utilisateur n'apprend rien sur l'ensemble des privilégiés, sauf s'il en est un membre ou pas. Cela permet de construire des systèmes de traçage à clés publiques à partir d'autres mécanismes tels que le *chiffrement par prédicat* [KSW08] ou *ciphertext-policy ABE anonyme* [NYO08] avec des politiques de sécurité suffisamment expressives, qui cachent la politique de sécurité qui contrôle le déchiffrement.

Traçage de traîtres à base de messages

La traçage de traîtres peut être facilement contourné en distribuant directement le message déchiffré. En effet, l'objectif d'un traître est de rendre le contenu accessible aux utilisateurs non autorisés, et il y a plusieurs façons d'y parvenir. En utilisant une coalition d'au plus t utilisateurs, un traître peut [JL07] :

a) fabriquer un décodeur pirate,

b) disséminer des messages déchiffrés,

c) disséminer la clé de session.

La stratégie a) est traitée par des systèmes de traçage des traîtres. Pour lutter contre b), l'expéditeur doit intégrer des tatouages numériques dans le message et distribuer des versions différentes du message aux différents utilisateurs. Pour combattre c), le système pourrait

i) ne pas utiliser une clé de session,

ii) prévenir l'extraction de la clé de session,

iii) permettre le traçage de la clé de session.

Le point i) semble de défavoriser la diffusion chiffrée basée sur le paradigme du chiffrement hybride. Cependant, tous les systèmes BE utilisent une clé de session implicite qui peut être extraite.

La prévention de l'extraction de la clé de session comme requis en ii) nécessite des hypothèses sur le matériel ou une sorte d'obfuscation de code. Plusieurs résultats suggèrent que l'obfuscation du code cryptographique est difficile à atteindre dans une certaine généralité [HMLS10]. Néanmoins il est mis en œuvre dans la pratique, sous le nom de cryptographie en boite blanche, et plusieurs propositions ont déjà été cassées [WMGP07]. Il peut donc être utile d'envisager de nouvelles notions de sécurité qui peuvent être atteintes de façon prouvée.

Tracer une clé de session comme dans iii) nécessite soit un type particulier de chiffrement traçable qui permet la dérivation de plusieurs clés qui ont le même comportement [BG03], ou l'utilisation de tatouage numérique dans un message comme dans la stratégie contre b). Ceci permet à l'autorité de traçage d'utiliser les clés de session pour déchiffrer et ensuite tracer sur la base du message résultant. C'est ce que nous appelons *traçage de traîtres à base de messages*.

Présentation des travaux

Nous donnons maintenant un aperçu sur des contributions décrites dans cette thèse. D'abord, nous établissons un cadre des notions de sécurité, que nous utilisons ensuite pour comparer les notions de sécurité qui ont été décrites dans la littérature. Ensuite, nous construisons un système de diffusion chiffrée dynamique qui satisfait la notion de sécurité la plus élevée. À la fin, nous présentons un système de traçage de traîtres à base de messages à partir du tatouage numérique avec un taux d'expansion optimal.

Notions de sécurité de la diffusion chiffrée

Depuis NNL, de nombreux systèmes BE ont été proposés, mais pour chaque système la preuve de sécurité a été effectuée dans un nouveau modèle de sécurité. En raison de ces modèles de sécurité divers et souvent *ad hoc*, il est difficile de comparer les mérites de ces systèmes, et il n'est pas toujours clair comment les notions de sécurité sont liées les unes aux autres.

Gentry et Waters [GW09], par exemple, ont défini une notion de sécurité qu'ils appellent *adaptative*, parce que l'adversaire peut corrompre les utilisateurs de manière adaptative avant la phase de défi. Or, il y a une notion qui est plus adaptative, où l'adversaire peut toujours corrompre des utilisateurs *après* la phase de défi. Pour différencier ces deux notions, nous les appelons « adaptative-1 » et « adaptative-2 », par analogie à IND-CCA1 et IND-CCA2. Un objectif de cette thèse est donc de fournir une meilleure approche des modèles de sécurité pour BE, et de les comparer. En particulier, nous étudions si les notions différentes de la corruption adaptative coïncident.

Nous définissons un modèle de sécurité plus systématique pour les mécanismes de diffusion chiffrée, et construisons un cadre générique des notions de sécurité pour BE. Nous prenons en compte, comme d'habitude dans le cadre de la sécurité prouvable, des oracles pour modéliser les moyens dont dispose l'adversaire, comme la possibilité d'ajouter de nouveaux utilisateurs, de corrompre des utilisateurs, et de déchiffrer des messages. Il est à noter que les petits détails peuvent avoir un impact important. Par exemple, le choix de l'ensemble des privilégiés pour le chiffré défi joue également un rôle dans la façon dont les modèles sont liés les uns aux autres. Nous étudions les relations entre les différentes notions, et constatons que, dans certains cas, deux notions sont équivalentes ou séparées en fonction de la disponibilité de certains oracles ou la résistance aux collusions d'un système BE. Après avoir décrit les relations entre les notions dans notre cadre, nous regardons de plus près les modèles de sécurité et les systèmes proposés dans la littérature et discutons où ils sont situés dans notre cadre, ce qui permet ensuite de les comparer.

Nos résultats sont significatifs pour des systèmes existants. À partir de la preuve trouvée dans [GW09], il est clair que la transformation à deux clés atteint effectivement un niveau de sécurité plus forte, noté adaptative-2.

En outre, nous définissons une notion de sécurité qui est plus forte que toutes les notions précédentes, et donnons un exemple d'un système qui répond à cette notion.

Diffusion chiffrée dynamique décentralisé

En 2001, Naor, Naor et Lotspiech [NNL01] introduisent le cadre « recouvrement de sous-ensembles » (SC, pour *Subset Cover*), qui peut être considéré comme une amélioration du système BE naïf, qui chiffre le message pour chaque utilisateur dans l'ensemble des privilégiés, par l'introduction de clés supplémentaires qui sont partagées par un sous-ensemble des utilisateurs. Si tous les utilisateurs qui partagent une clé sont dans l'ensemble des privilégiés, alors cette clé est utilisée pour chiffrer une fois, ce qui réduit la longueur du chiffré par rapport à chiffrer une fois pour chaque utilisateur. Plus les ensembles sont grands, meilleure est l'amélioration de l'efficacité, mais aussi plus grand est le risque que l'un des utilisateurs dans l'ensemble soit révoqué. La longueur du chiffré dans la version « différence de sous-ensembles » (SD, pour *Subset Difference*) de NNL dépend linéairement du nombre d'utilisateurs dans l'ensemble des révoqués, ce qui a été considéré comme suffisamment efficace pour une utilisation dans la norme DRM AACS [AAC09]. Nous généralisons le cadre SC de NNL pour à la fois obtenir un chiffrement à clé publique et faire face aux changements dynamiques de l'ensemble des utilisateurs enregistrés. Par conséquent, nous construisons le premier système de diffusion chiffrée qui est entièrement dynamique et qui satisfait la notion IND-CCA sous corruptions entièrement adaptatives, dans le modèle standard sous l'hypothèse DDH. Nous supprimons en outre la nécessité d'autorités de confiance en éliminant le gestionnaire du groupe, qui interagit généralement avec les utilisateurs pour distribuer les clés lors de la phase d'initialisation ou lorsque les utilisateurs rejoignent le système. Notre approche fait usage de l'échange de clés au sein de groupes, avec clés de sous-groupes [Man09, ACMP10], une primitive qui distribue simultanément des clés différentes pour certains sous-ensembles du groupe d'utilisateurs et s'applique bien au cadre SC si l'on peut obtenir des clés pour les sous-groupes associés à la couverture de sous-ensembles.

Nous instancions d'abord notre construction avec l'accord de clé Diffie-Hellman pour l'initialisation interactive et le chiffrement ElGamal comme chiffrement à clé publique, ce qui conduit à un système efficace. La construction « sous-arbre complet » (CS, pour *Complete Subtree*) ressemble à l'accord de clés au sein de groupes à base d'arbres dans [KPT04], mais nous créons également des paires de clés pour les noeuds internes, et nous allons au-delà de leur système dans notre construction d'arbres SD. Nous

montrons ensuite comment notre système peut être étendu en utilisant le chiffrement Cramer-Shoup pour atteindre la notion de sécurité la plus forte, qui permet de résister aux corruptions adaptatives et aux attaques à chiffrés choisis, dans le modèle standard, sous l'hypothèse DDH. En outre, nous considérons des critères différents d'efficacité : la taille du chiffré, de la partie secrète et de la partie publique des clés de déchiffrement, le nombre de tours pour la génération de clés, etc. Grâce à la modularité de notre approche, nous pouvons utiliser n'importe quel protocole d'échange de clés au sein de groupes avec des clés de sous-groupes : notre technique initiale utilise de manière itérative l'échange de clés Diffie-Hellman pour deux partis dans un arbre binaire, ce qui nécessite un nombre logarithmique de tours. Nous pouvons le remplacer par un nombre logarithmique d'exécutions parallèles du protocole Burmester-Desmedt d'échange de clés au sein de groupes [BD05], ce qui réduit le nombre de tours à deux. Nous proposons aussi deux améliorations. En plus de permettre aux utilisateurs de se joindre au système, nous donnons également un aperçu de comment deux groupes pourraient fusionner à coût faible, et une façon de révoquer de façon permanente certains utilisateurs. Notre système réalise ainsi un maximum de fonctionnalités et de sécurité sous des hypothèses minimales, tout en étant raisonnablement pratique.

Diffusion chiffrée et échange de clés au sein de groupes Comme déjà expliqué, nous voyons la diffusion chiffrée en tant que diffusion de clés encapsulées, qui sont utilisées en combinaison d'un chiffrement symétrique pour chiffrer les messages. De ce point de vue, le système NNL utilise des clés symétriques à long terme pour permettre à un centre de confiance de distribuer de façon sûre une clé de session symétrique à tous les utilisateurs dans l'ensemble des privilégiés, ce qui établit une clé éphémère.

Or, le système contient un deuxième mécanisme d'établissement de clés : des clés symétriques à long terme sont attribuées à chaque sous-ensemble du recouvrement et distribuées aux utilisateurs par le gestionnaire de groupe. Traditionnellement, cette distribution de clé de groupe est effectuée durant de la phase d'initialisation entre le gestionnaire de groupe et les utilisateurs. Il est supposé être effectué d'une manière sécurisée, et donc il ne fait pas partie du modèle de sécurité. On peut aussi bien dire que cette phase nécessite des canaux sécurisés entre le centre et les utilisateurs. Pour éviter de tels canaux sécurisés et l'autorité de confiance dans la distribution des clés, des protocoles d'échange de clés au sein de groupes sont une alternative plus réaliste. Nous allons inclure ce processus de génération de clé dans le modèle de sécurité, mais seulement contre des adversaires passifs. Cela peut être considéré comme le remplacement de l'hypothèse de « canaux sécurisés » lors de l'installation avec l'hypothèse plus faible de « canaux authentifiés ». Pour plus de sécurité contre des adversaires actifs, ce qui est nécessaire pour éliminer l'hypothèse des canaux authentifiés, un mécanisme d'authentification supplémentaire sera nécessaire ; des compilateurs génériques efficaces existent [KY07], main ceci n'est pas dans le cadre de cette thèse.

Travaux connexes La diffusion chiffrée sans autorité centrale remplace la configuration classique avec un processus d'initialisation des clés de groupe qui peut être un protocole interactif. Cette approche a été proposée sous le nom de *diffusion chiffrée contributive* (CBE, pour *Contributory Broadcast Encryption*) dans [WQZ+11], avec un système de sécurité IND-CPA semi-adaptative qui n'est pas dynamique. Une application possible pourrait être la communication dans un réseau social, où certaines informations privées sont destinées à un sous-ensemble des connaissances de l'utilisateur, et le réseau est pair-à-pair où le fournisseur de service n'est pas fiable. La première étape vers l'échange de clés de sous-groupe a été réalisée par Manulis [Man09], qui s'étend en un protocole d'échange de clés au sein de groupes (GKE, pour *Group Key Exchange*) pour permettre à tous les couples d'utilisateurs de calculer une clé commune après la phase initiale dans laquelle la clé de groupe est calculée. Suite à ces travaux, Abdalla et al. [ACMP10] ont généralisé cette approche pour permettre le calcul des clés de session pour des sous-ensembles arbitraires. Nous utilisons un tel protocole d'échange de clé au sein de groupes avec les clés de sous-groupes pour dériver des clés de chiffrement asymétriques pour des sous-ensembles. Quelque chose de semblable a été proposé sous le nom d'*accord de clés asymétriques au sein de groupes* (ASGKA, pour *Asymmetric Group Key Agreement*) [WMS+09]. Dans [WMS+09], ASGKA est défini d'une manière qui garantit seulement que les clés détenues par les participants sont bonnes pour une utilisation avec un système de chiffrement spécifique. Nous voulons généraliser cette exigence de sorte qu'à la fin de l'exécution du protocole, chaque utilisateur dispose d'un aléa, qui peut par la suite être utilisé par n'importe quel mécanisme de génération de clé, et notamment pour générer des paires de clés pour un mécanisme d'encapsulation de clé. Comme cet aléa est partagé entre les différents sous-groupes, nous appelons ce système que nous utilisons pour la configuration un *échange de clés de sous-groupe* (SKE, pour *Subgroup Key Exchange*). Kurnio, Safavi-Naini, et Wang [KSNW03] considèrent explicitement le

parrainage des candidats du groupe par les membres existants. Dans notre système, en raison de la structure de l'arbre, chaque utilisateur peut agir comme un parrain et un seul parrain est requis pour permettre à un candidat de se joindre à l'ensemble des utilisateurs.

Traçage de traîtres à base de messages

Nous proposons le terme *traçage de traîtres à base de messages* (MT, pour *Message-based Traitor Tracing*) comme un terme générique qui englobe les variantes antérieures et insiste sur le fait que nous ne traçons pas à partir des décodeurs pirates, mais à partir d'informations dans le contenu. Fiat et Tassa ont été les premiers à considérer le traçage de traîtres à base des messages ; dans [FT99], ils développent le traçage de traîtres *dynamique* pour faire face aux traîtres qui redistribuent le contenu déchiffré. Des tatouages numériques sont insérés dans le contenu selon un code binaire, et les auteurs supposent que le centre peut observer un retour du contenu de la diffusion en temps réel, de sorte que le code peut être adapté de façon dynamique. Safavi-Naini et Wang [SNW03a] notent que, dans ce cadre, les traîtres peuvent éviter le traçage en retardant la rediffusion du contenu. Pour prendre en compte cette contre-mesure, ils proposent le *traçage de traîtres séquentiel*, où l'allocation de tatouages numériques est précalculée, mais les utilisateurs sont révoqués en fonction de la rétroaction reçue. Ils construisent un système de TT séquentiel en combinant les codes correcteurs d'erreurs et les tatouages numériques. Jin et Lotspiech [JL07] supposent que les traîtres supprimeront des parties de contenu et affirment que la protection ne devrait pas augmenter la bande passante de plus de 10 %. Pour résoudre ce problème, ils ont proposé d'étendre la procédure de traçage sur plusieurs films (en utilisant des codes « internes » et « externes »). Leur système permet la révocation d'utilisateurs après qu'ils ont été tracés. Kiayias et Pehlivanoglu [KP09] montrent que le système par clés séquentielles par blocs ne permet de retracer et de retirer qu'un nombre limité d'utilisateurs, et proposent un système traçage-et-révocation à base des messages sans cette limitation.

Taux d'expansion optimal Contrairement au traçage classique où il existe des systèmes à taux d'expansion optimal, le problème de la construction d'un système MT avec un taux d'expansion optimal est toujours ouvert. Nous expliquons pourquoi les solutions de traçage classique échouent lorsqu'elles sont appliquées à un système MT et nous décrivons ensuite notre approche.

Boneh et Franklin [BF99] développent un système de traçage des traîtres avec une taille de chiffré linéaire dans le nombre maximal d'utilisateurs collaborant. Kiayias et Yung [KY02c] intégrent une version de ce système pour deux utilisateurs avec un code traçant et obtiennent le premier système TT avec un taux d'expansion constant. Dans cette méthode, l'expéditeur chiffre essentiellement tous les blocs deux fois, de sorte que le destinataire ne peut déchiffrer qu'un des deux chiffrés pour chaque bloc. La procédure de traçage consiste à utiliser le clair ou des clés réparties pour extraire un mot associé au décodeur pirate. Grâce à la capacité de traçage du code traçant, on peut alors trouver un des traîtres à partir de ce mot. Le système de Kiayias et Yung requiert un texte chiffré trois fois plus grand que le contenu initial. Fazio, Nicolosi, et Phan [FNP07] ensuite atteignent un taux d'expansion asymptotiquement de 1. Leur méthode consiste à chiffrer seulement un bloc deux fois et à appliquer ensuite une transformation tout-ou-rien (AONT, pour *All-Or-Nothing Transform*), qui garantit que les traîtres ne peuvent pas supprimer ce bloc particulier parce que l'absence de seulement un bloc rend les traîtres incapables d'obtenir des informations sur le message. L'utilisation de l'AONT dans [KY02c, FNP07] est intéressante mais peu pratique car le récepteur doit attendre jusqu'à ce qu'il ait reçu n blocs (où n est la longueur des mots du code en cours d'utilisation, donc très grand) pour commencer la procédure de déchiffrement. Nous notons que, sans chercher à optimiser le taux d'expansion, l'utilisation de AONT peut être évité en utilisant un code traçant robuste qui permet aux traîtres d'ignorer une partie des positions. Un tel code est utilisé dans [Sir07, BP08, BN08] pour réduire la taille du chiffré. Toutefois, afin d'obtenir un taux d'expansion optimal dans [FNP07], l'utilisation de AONT est obligatoire, sinon les traîtres peuvent tout simplement supprimer ce bloc particulier pour rendre inopérante la procédure de traçage.

Au regard des systèmes de traçage de traîtres à base de messages, une question naturelle est pourquoi nous n'appliquons pas simplement la méthode ci-dessus aux systèmes MT pour optimiser le taux d'expansion. Nous remarquons d'abord que, dans toutes les méthodes ci-dessus pour le traçage classique, chaque utilisateur obtient finalement le même message et si un utilisateur le redistribue, le centre n'a aucun moyen de tracer ce traître. Par conséquent, la condition nécessaire pour MT est que chaque utilisateur reçoive une version différente (donc marquée) du message. Toutefois, lorsque le message est différent pour chaque utilisateur, on ne peut pas appliquer un AONT à tout un texte clair fixe, sinon au plus un utilisateur peut déchiffrer. L'AONT ne peut donc pas s'appliquer au cas MT. Heureusement, nous

pouvons toujours utiliser la méthode qui consiste à doubler un bloc particulier en trouvant un moyen de cacher ce bloc. Notre méthode consiste à utiliser un système de diffusion chiffrée anonyme pour deux utilisateurs (2ABE, pour *2-user Anonymous Broadcast Encryption*), puis permuter de façon aléatoire les blocs. Avec un 2ABE, le traître ne peut pas détecter une différence entre un chiffrement pour les deux utilisateurs (qui est utilisé pour tous les blocs, sauf le bloc particulier) et un chiffrement destiné à seulement un des deux utilisateurs, qui est utilisé pour le bloc particulier. En combinaison avec la permutation des blocs, nous pouvons montrer que le traître est incapable de détecter le bloc protégé particulier. Par ailleurs, au-delà de l'optimisation du taux d'expansion, notre système bénéficie également de la propriété du déchiffrement séquentiel via l'utilisation d'un code traçant comme [BP08, BN08] : l'utilisateur peut déchiffrer de manière séquentielle les sous-chiffrés, et n'a pas besoin d'attendre jusqu'à ce qu'il ait reçu l'ensemble de chiffré pour lancer la procédure de déchiffrement.

Robustesse Nous utilisons un modèle réaliste d'adversaire qui prend en compte les contre-mesures que les traîtres pourraient prendre par la suppression des tatouages numériques dans le contenu. L'*hypothèse du tatouage* énonce qu'il est impossible pour un adversaire de détecter, supprimer ou modifier des tatouages s'il ne possède qu'une version du même message. Nous devons donc nous assurer que notre système ne révèle aucune information sur la position des tatouages, ce qui laisse l'adversaire avec la possibilité de supprimer des messages de manière aléatoire dans l'espoir d'éliminer également un tatouage. Nous fixons un seuil δ au delà duquel que le message devient inutile si l'adversaire supprime plus qu'une fraction δ du message. Nous devons alors faire face à un message où au plus une fraction δ des tatouages sont manquants. Cela nous permet également d'affaiblir l'hypothèse tatouage : parce que nous savons comment nous arranger des marques effacées, il suffit de demander qu'il soit difficile (mais pas impossible) d'enlever des tatouages.

Articles

Les travaux menant à cette thèse ont donné lieu à divers articles publiés :

Security Notions for Broadcast Encryption [PPS11]

avec Duong Hieu Phan et David Pointcheval.

Cet article clarifie les relations entre les notions de sécurité pour la diffusion chiffrée. Au début de nos études sur la diffusion chiffrée, nous avons noté que chaque système était accompagné par sa propre définition de sécurité, ce qui les rend difficiles à comparer. Nous définissons ainsi un ensemble de notions, comme cela a été fait pour la signature et le chiffrement, pour lesquels nous démontrons les implications et les séparations. Enfin, nous montrons le lien entre les notions existantes et celles de notre cadre. Nous trouvons quelques relations intéressantes entre des notions différentes, notamment dans la façon dont on définit l'ensemble des privilégiés du message-défi. Cet article, qui à reçu le prix du meilleur article d'un étudiant à ACNS 2011, est repris dans le chapitre 4.

On the Joint Security of Encryption and Signature in EMV [DLP+12]

avec Jean Paul Degabriele, Anja Lehmann, Kenneth G. Paterson, et Nigel P. Smart.

Nous fournissons une analyse des algorithmes de signature et de chiffrement actuels et futurs dans les normes EMV dans le cas où une seule paire de clés est utilisée pour la signature et le chiffrement. Nous donnons une attaque théorique pour les algorithmes actuels de EMV basés sur RSA, en montrant comment l'accès à un oracle partiel de déchiffrement peut être utilisé pour forger une signature sur un message choisi librement. Nous montrons comment l'attaque pourrait être intégrée dans le flux du protocole CDA de EMV, permettant à un attaquant qui peut interposer un dispositif d'interception entre le terminal et la carte bancaire de compléter une transaction hors ligne sans connaître le code PIN du porteur. Enfin, nous montrons que les systèmes de signature et de chiffrement basés sur des courbes elliptiques qui sont susceptibles d'être adoptés dans la prochaine version de la norme EMV sont sûrs même s'il sont utilisés avec une seule paire de clés. Cet article est le résultat d'une coopération au sein du réseau d'excellence européen ECRYPT II ; étant hors du sujet, il n'est pas repris dans cette thèse.

Adaptive CCA Broadcast Encryption with Constant-Size Secret Keys and Ciphertexts [PPSS13]

avec Duong Hieu Phan, David Pointcheval, et Siamak F. Shahandashti.

Nous considérons la conception de systèmes de diffusion chiffrée avec des clés secrètes et chiffrés de taille

constante, qui satisfont la sécurité IND-CCA2. Nous justifions d'abord que des transformations CPA-à-CCA connues ne sont pas applicable à ce cas. Nous proposons alors un système, qui est une modification d'un système IND-CPA sélectif proposé par Boneh, Gentry, et Waters. Notre schéma utilise des clés secrètes et chiffrés de taille constante et nous démontrons qu'il est IND-CCA sélectif sous des hypothèses standard. Notre système se caractérise par des chiffrés plus courts que ceux des propositions IND-CCA précédentes. Nous proposons ensuite un second système qui réunit la fonctionnalité de diffusion chiffrée et un mécanisme de révocation en utilisant le même ensemble de paramètres. Enfin, nous montrons qu'il est possible de prouver que notre premier schéma est IND-CCA adaptatif sous des extensions raisonnables de l'hypothèse Diffie-Hellman bilinéaire avec exposant (BDHE, pour *Bilinear Diffie-Hellman Exponent*) et l'hypothèse de connaissance des exposants (KEA, pour *Knowledge of Exponent Assumption*). Nous montrons que ces deux hypothèses sont valables dans le modèle du groupe générique. Par conséquent, notre système est le premier à atteindre à la fois des chiffrés et clés secrètes de taille constante (asymptotiquement optimales) et le niveau de sécurité IND-CCA adaptatif. Cet article est repris dans la partie 4.5.

Decentralized Dynamic Broadcast Encryption [PPS12a]

avec Duong Hieu Phan et David Pointcheval.

Un système de diffusion chiffrée implique généralement trois types d'entités : le gestionnaire de groupe qui se charge de la composition du groupe, l'émetteur qui chiffre les données aux utilisateurs enregistrés selon une politique de sécurité spécifique (l'ensemble des privilégiés), et les utilisateurs qui déchiffrent les données s'ils y sont autorisées par la politique de sécurité. On peut dire que la diffusion chiffrée à clé publique supprime le rôle particulier d'émetteur, en permettant à quiconque d'envoyer des données chiffrées. Dans cet article, nous allons un peu plus loin dans le processus de décentralisation, en supprimant le gestionnaire de groupe : l'initialisation du groupe, ainsi que l'ajout d'autres membres au système, ne nécessitent pas une autorité centrale. Notre construction utilise en boîte noire des primitives connues et peut être considérée comme une extension du cadre NNL. Elle permet des instantiations concrètes efficaces, avec des tailles de paramètres qui correspondent à celles des constructions NNL, en atteignant simultanément le niveau de sécurité le plus élevé dans le modèle standard sous l'hypothèse DDH. Cet article est repris dans le chapitre 5.

Message Tracing with Optimal Ciphertext Rate [PPS12b]

avec Duong Hieu Phan et David Pointcheval.

Le traçage de traîtres est un outil important pour décourager les utilisateurs de diffuser illégalement des contenus multimédia. Cependant, les techniques principales consistent à tracer les traîtres à partir des décodeurs pirates qu'ils ont construits en utilisant des clés secrètes d'utilisateurs légitimes malhonnêtes : en utilisant une procédure de traçage soit en boîte noire soit en boîte blanche sur le décodeur pirate, on espère trouver un des traîtres qui se sont inscrits dans le système. Or, de nouvelles techniques pour les traîtres consistent à envoyer les clés de session aux décodeurs pour le déchiffrement en temps réel, ou à rendre les contenus disponible sur le web. De cette façon, le traître n'envoie pas d'informations personnelles. Afin d'être en mesure de retracer les traîtres, il faut insérer des informations, comme des tatouages numériques, dans le contenu multimédia pour le rendre spécifique à chaque utilisateur. Cet article traite ce problème de traçage des traîtres à partir du contenu ou des clés de déchiffrement, sans trop augmenter les besoins en bande passante. Plus précisément, nous construisons un système de traçage de traîtres à base de messages avec un taux d'expansion optimal, c'est-à-dire le rapport entre la longueur globale du chiffré et la longueur du message est arbitrairement proche de 1. Cet article est repris dans le chapitre 6.

Chapter 1

Introduction

Contents

Before diving into technical matters, we want to briefly reflect on the purpose of writing a thesis. The reason for writing this thesis is of course the fact that it is a requirement for obtaining an academic degree, which is a prerequisite for working as a scientist in a fascinating field of research. The target audience are the two reviewers who will write a report, and who expect the thesis to present already published results in a coherent fashion, motivate the questions or open problems that led to these results, describe how the results are an improvement over the state of the art, and display some degree of mastery of the tools of the trade, the terminology and techniques that are at the heart of this field and which permit a young researcher to independently identify and work on open problems.

If this was all a thesis was expected to accomplish, it would be pointless to publish it and preserve it in a library after the thesis defence. A thesis, while not a textbook, should also be a work which can stand on its own and serve as a starting point in the search for information on a topic. We aim to make this work accessible to readers of varying levels of expertise. In section 1.1, we give an overview of the development of modern cryptography with a focus on the discoveries we consider most relevant to this thesis. We then turn to the specific problem of content protection in section 1.2, before detailing our contributions in a comprehensible manner in section 1.3.

1.1 An Excerpt from the History of Modern Cryptography

The first contribution to modern cryptography was published in 1949 by Shannon, who defined *perfect secrecy* of a symmetric encryption scheme in terms of information theory: A ciphertext reveals nothing about the message it contains (except for its length). In other words, even a computationally unbounded adversary learns nothing about an encrypted message by seeing its encryption that he would not also have learned from a random string of the same length. Shannon also provided a necessary condition for attaining perfect secrecy: The key space from which the key is drawn must be at least as big as the message space, which means in practice that keys have to be at least as long as the message they encrypt. This idea of defining security and proving that a scheme meets this definition is one of the fundamental ideas of cryptology.

The era of modern cryptography began in the 1970's, with the public standardisation process of the data encryption standard DES in the USA, developed by the civilian corporation IBM with some modifications by the national security agency. This open process followed a principle stated in 1883 by Kerckhoffs, that the security of a cryptographic system should not depend on its secrecy. This allows public review of the system, which contributed to the development of cryptology as a science.

The second revolution of this decade was the discovery of public-key cryptography. Merkle developed in 1974 a key exchange protocol known as Merkle's puzzles, which allows two users to agree on a shared secret over a public channel, using a symmetric cipher with a key space containing κ keys for some integer κ. The sender encrypts κ random strings, prefixed by a public constant string, using each of the κ keys and sends these messages to the receiver. This constant prefix is a first attempt at *robustness*, the property that lets a user know when he used an incorrect key to decrypt. The receiver chooses one of these puzzles at random and tries to decrypt it with randomly chosen keys until the random message starts with the known constant prefix. He then sends back the random message, and the key he used to decrypt is now the common secret between the two. Both sender and receiver need to perform around κ operations, but an eavesdropper needs to brute-force on average $\kappa/2$ puzzles, which takes around $\kappa/2$ operations each, so his workload is in the order of κ^2. This means that for the protocol to be secure, it is necessary that the sender and receiver both have approximately the same computing resources as the attacker, otherwise κ needs to be chosen very large.

The protocol presented in 1976 by Diffie and Hellman overcomes this problem by relying on a one-way function, a function which is easy to compute and hard to invert, with special properties. There are efficient algorithms for computing discrete exponentiation, i. e. the value g^a for some element g in a finite group $\mathbb{G}$, such as $\mathbb{Z}_p^*$ for a prime number p, and some scalar a. For the inverse operation, computing a, given g and g^a, no efficient algorithms are known. Diffie and Hellman proposed the following protocol between two parties A and B. Party A chooses a random scalar a, computes g^a and sends the value to B. Party B also chooses a random scalar b, computes g^b and sends the value to A. Both parties exponentiate the received value with their secret scalar and obtain $(g^a)^b = g^{ab} = (g^b)^a$ as their shared secret. Here, the security of the scheme rests on the hardness of a number-theoretic problem. The difference in the computational demand on the users and the adversary is of a different quality than for Merkle's puzzles, because the number of steps necessary to break the protocol increases not as κ^2 but faster than any polynomial in κ.

In 1977, Rivest, Shamir, and Adleman presented the first public-key encryption scheme, now known as RSA, which works in groups of composite order and is related to a different number-theoretic problem, integer factorization. A user chooses two large primes p and q and computes $n = pq$, $\varphi(n) = (p-1)(q-1)$, and two exponents e and d such that $de \equiv 1 \mod \varphi(n)$. In the multiplicative group $\mathbb{Z}_n^*$, which is of order $\varphi(n)$, we have $x^{ed} \equiv x^1 = x$. This means that by making e, n public, anyone can encrypt a message m to $c \equiv m^e \mod n$, but the secret exponent d is necessary to decrypt.

This simple scheme is deterministic, which means that by encrypting candidate messages and comparing the encryptions to a given ciphertext, it can be excluded that the ciphertext will decrypt to one of these messages. This makes it clear that security needs to be formally defined. The first such definition was *semantic security*, a computational analogue to the information-theoretic perfect secrecy, which was proposed by Goldwasser and Micali in 1982. They later presented an equivalent notion, which is easier to use in security proofs, *indistinguishability under chosen-plaintext attacks* (IND-CPA). IND-CPA is described in the form of an experiment. An adversary (with polynomially bounded running time) is given the public key, then he chooses two equal-length messages. One of these is chosen at random, encrypted with the public key, and the resulting ciphertext is given to the adversary. The adversary wins the experiment, if he can tell which of the two messages was encrypted.

In 1990, Naor and Yung defined *indistinguishability under chosen-ciphertext attacks* (later named IND-CCA1) for public-key encryption schemes, where the adversary is given access to a decryption oracle to help him find two messages whose encryptions are easy to distinguish.

In 1991, Rackoff and Simon gave an even stronger definition, *indistinguishability under adaptive chosen-ciphertext attacks* (IND-CCA2), where the adversary has access to the decryption oracle even after he receives the challenge ciphertext, with the restriction that he may not ask the challenge ciphertext to the decryption oracle.

1.2 Content Protection

Public-key encryption schemes are used to provide confidentiality for unicast or point-to-point communication such as email (PGP) or web browsing (RSA encryption in TLS). For one-to-many communication using multi- or broadcast channels such as mailing lists or radio, a different approach is needed. Using PKE to encrypt a message to many users quickly becomes impractical. When the user set changes rarely, and all users should always be able to decrypt, a good option can be to share a key among all users. We are concerned with a more challenging task. *Broadcast encryption* (section 1.2.1) is a generalization of symmetric or asymmetric encryption to the one-to-many setting, where a sender can choose

at the time of encryption the set of users that can decrypt a broadcasted message. In the real world, there are several ways to transmit a message to several users simultaneously. The IPv4 protocol supports broadcast addresses [Mog84]. If a message (datagram) is sent to this address on a remote network, the receiving gateway will broadcast it to all nodes of this network. The IPv4 and IPv6 protocols both support multicast addresses [CVM10]. These addresses are assigned to user groups, and a message sent to a multicast address is delivered to all group members. This makes group management necessary to ensure that there is no ambiguity about group membership. There is also the possibility of physical broadcast via a radio network or satellites. All these options have different characteristics.

A typical application for broadcast encryption is content protection, either for Pay-TV, where a provider wants to ensure that only users who paid for their subscription can view the content that is broadcasted across a large area, or for digital rights management (DRM), where only decoders that obey certain policies in handling content should be able to decrypt. In practice, BE is used for example by copyright protection mechanisms for digital media such as DVDs, to ensure that only DVD players that comply with a standard can decrypt and process the content. If the keys for a series of DVD players become known, this series will not be able to play DVDs produced after the series is revoked [NNL01].

Broadcast encryption is only concerned with providing confidentiality against revoked users, it cannot prevent authorized users from distributing the decrypted content or from making their decryption capabilities available to others. This is the domain of *traitor tracing* (section 1.2.2).

1.2.1 Broadcast Encryption

We think it is important to distinguish between broadcast encryption schemes and the task of broadcast encryption. The task of broadcast encryption is to encrypt a message to a set of receivers which can be different for each ciphertext.

Broadcast encryption (BE) schemes [FN94] enable the sender of a message to determine a subset of users to which the message will be encrypted (the *target set* or *privileged set*). If the set of all users is static, the complement of the target set is called the *revoked set*. The complement to BE schemes are *revocation schemes*, which define the revoked set directly. Both assume nothing about the user set. If the user set has some structure, it might be more appropriate to use attribute-based encryption.

We concern ourselves mainly with broadcast encryption and revocation schemes, and sometimes do not differentiate between them when we talk about static schemes, where the user set is fixed. In the dynamic case, where users can join after messages have been encrypted, there is a difference between BE and revocation schemes. For a dynamic BE scheme users that join after the message is sent are in the revoked set, while for a revocation scheme new users are in the privileged set.

A broadcast encryption ciphertext consists of three parts: The *id header* is a bit string that unambiguously identifies the target set. All known BE schemes are hybrid schemes, where there is a *key header*, which distributes a session key to exactly the privileged users and the *message body* contains an encryption of the message under the session key.

To uniquely identify a subset of a set of size N requires N bits. If the size r of the revoked set is small, it is sufficient to identify only the revoked users. This can be done using $r \log N$ bits. The same technique applies if the target set is small. If the receivers are keeping state and the difference between two successive target set is small, the size of the id header can be reduced by transmitting only this difference. Because the technique used does not depend on the BE scheme, the size of the id header is usually not included in the calculation of the ciphertext length when BE schemes are compared. The message body, which does not depend on the BE scheme either, is also not taken into account. In general, BE schemes are concerned only with the transmission of the key header, and are considered to be key encapsulation mechanisms (KEM) [CS03].

There are two trivial BE schemes. In the first one, the centre shares a symmetric key with each user. The key header then consists of an encryption of the session key for every key shared with a privileged user. This scheme achieves ciphertext length $N - r$, user storage 1, and constant decryption time. The second one is the power-set scheme, which assigns a key to each subset of the user set. In this scheme, the key header is empty, as the id header already defines the session key. Therefore, the length of the key header is 0, user storage is $\mathcal{O}(2^N)$, and the decryption time is constant. The goal is now to find a BE scheme that improves on these two or provides a better trade-off. The most critical value when evaluating BE schemes is the length of the key header, as bandwidth is usually the bottleneck. Because the sender can be assumed to have a large storage and computational power, the other important variables are storage and computational complexity at the receiver end.

The Setting

Our setting consists of N users and a special user we call the *centre* C. All users are connected through a network, and additionally via a broadcast channel. In the basic setting, the network is considered *insecure*, which means that all messages sent over the network are first sent to the adversary $\mathcal{A}$, who can choose not to forward them, modify them before forwarding, send them to other users than intended by the sender, modify the apparent origin, introduce completely new messages, or reorder messages. In the *authenticated channels* (AC) model [BCK98], the adversary can read all messages sent over the network and drop or reorder them, but he cannot modify or reroute them, or introduce new messages. This models messages that are perfectly authenticated. We distinguish two types of broadcast channels: network broadcast and physical broadcast. When messages are sent over a *network broadcast channel*, the adversary has the same powers as for insecure channels described above. We define *authenticated broadcast channels* as above. Incidentally, the authenticated broadcast channel also models an unreliable physical broadcast channel, where users might not receive messages because they are out of range, and where messages could be reordered due to differences in propagation speed caused by diffraction, refraction, etc. When messages are sent over a (reliable) *physical broadcast channel*, the adversary can read all messages, but is completely passive. This models for example satellite broadcasts.

In addition to his power over the network, the adversary is allowed to corrupt users (but not the centre). When this happens, their complete state is revealed to the adversary, and their behavior is controlled by him from then on. In the model of *static corruption*, the adversary must corrupt all users before the protocol is run. In the *adaptive corruption* model, he can corrupt users at any time during protocol execution.

Genealogy

There are several research directions that deal with related problems. *Group key exchange* is the problem of establishing a shared secret between any subset of the group of N users over a network with authenticated channels. In general, before any message is sent, a setup phase is assumed to take place. An extension of this problem is *authenticated group key exchange*, where the adversary has full control over the network. Without being formal about it, we will call a key exchange protocol a *key agreement* when the session key is determined jointly by all participants, and a *key transport* when one user chooses the session key.

The special case of *group key transport* is very close to our goal, where a single sender establishes a common key by sending a message to every user. It could thus be said that the task of broadcast encryption is accomplished by a non interactive group key transport protocol, where the sender sends a single message to all users. The main difference is that the requirement on the session key change: Unlike in the broadcast encryption case, a session protected by a group key transport can consist of several messages, so it is reasonable to expect that several messages will be encrypted or authenticated with a single key. The other difference lies in the security model used. While for key transport protocols an interactive model is used that is centred on the adversary's power over the network and user corruption, broadcast encryption is evaluated in a model derived from public-key encryption (which usually also includes user corruption). Of special interest are *stateless* broadcast encryption schemes, where computations depend only on the initial state and the last received ciphertext. In general, protocols that require stateful users can be transformed into allowing stateless users by sending the complete state-update information at the beginning of each message [NNL01], although that can be inefficient. Also interesting are protocols that do not require the users to always be online, e. g. to receive update messages. This is always the case for stateless users.

Dynamic group key exchange allows for changes of the user set modeled via *Join* and *Leave* protocols run by a *group manager*. The analogue to this in the BE world is *dynamic broadcast encryption* [DPP07].

A scheme where the session key is always shared by the user group and updated when the group composition changes, is usually known as *multicast encryption* [WHA99, WGL00]. This problem is also known as *re-keying*, and naturally demands forward and backward security.

Several schemes exist that imply broadcast encryption: Broadcast encryption can be constructed from *attribute-based encryption* (ABE) [SW05, GPSW06], with the technique depending on the expressiveness of the policy and whether it is associated with the key or the ciphertext. In *hidden-vector encryption* (HVE) [BW07, IP08], ciphertexts are associated with binary vectors, and keys are associated with policy vectors that may contain "don't care" entries ($*$). If the policy is instead associated with the ciphertext, it is possible to emulate BE by using a policy vector of length n and identifying users with a position i in the vector. Then every user is given a key with an identifier that has 1 at position i, and 0

everywhere else. The policy vector contains a 0 at every position corresponding to a user not in the receiver set, and a $*$ everywhere else. Because the ciphertext vector is hidden even to users who can decrypt, HVE even implies *anonymous* broadcast encryption, where no user learns any information about the target set except whether he is contained in it or not. *Wildcard identity-based encryption* (WIBE) is a primitive where users are identity vectors and ciphertexts contain a vector of identities and wildcards "$*$" [ACD+06]. A user can decrypt if his identity vector matches the ciphertext identity vector in all positions that are not $*$. WIBE can simulate HVE (without the anonymity) using a hierarchy depth equal to the length of the vector, and therefore it also implies broadcast encryption.

1.2.2 Traitor Tracing

While broadcast encryption can be seen as a generalization of PKE to the one-to-many setting, traitor tracing is concerned with a genuinely new problem. We first look at the setting where a centre encrypts a message to all users. We want to protect against users sharing their secret decryption keys with people outside the group. Users who act in this way are called *traitors*. Since there is no way to prevent users from sharing their secret keys or algorithms that contain these keys, the idea is that being able to identify a traitor from his decryption key will prevent users from giving their keys away.

Traitor Tracing

Traitor tracing (TT) schemes [CFN94] allow a tracing authority to find out users who contributed secret keys to a pirate decoder, an algorithm that can decrypt ciphertexts. A TT scheme is called *private-key*, if the encryption key must be kept secret, and *public-key* if anybody can encrypt. If any user can trace, the scheme is called *publicly traceable* [CPP05]. Broadcast encryption schemes that allow for traitor tracing (or traitor tracing schemes that allow for selective addressing of user sets) are called *trace-and-revoke* (TR) schemes [NP01]. TR schemes allow the centre to continue tracing after a pirate has been found out and its keys disabled. They have to deal with evolving decoders [KP07], which try to make the most of their keys and generate as many generations of decoders as possible after the previous generation has been rendered useless.

The adversary in our setting is passive and can corrupt users. He can then use their keys to construct a *pirate decoder*, an algorithm that can decrypt ciphertexts. The tracing algorithm of the TT scheme then gets access to the pirate decoder and has to find at least one of the users who contributed to the decoder. Depending on what kind of access the tracing algorithm gets to the pirate decoder, we distinguish between *white-box tracing*, where the tracing algorithm can extract the keys of the decoder, *black-box tracing*, where the tracing algorithm only gets oracle access, and *minimal access tracing*, where the tracer only has access to an oracle indicating whether the message was decrypted correctly. Usually, the pirate decoder is assumed to be *stateless*, which models a resettable hardware decoder or a software decoder of which a new copy is used for each query. A *stateful* decoder, which can remember its past interactions, is much harder to trace.

One paradigm in the construction of traitor tracing schemes is to use a scheme that can address subsets of the user set, but where a user does not learn anything about the target set except whether he is a member of it or not. This allows construction of public-key traitor tracing schemes from other schemes such as *predicate encryption* [KSW08] or *anonymous ciphertext-policy ABE* [NYO08] with sufficiently expressive policies, which hide the policy that controls decryption.

Message-based Traitor Tracing

Traitor tracing can easily be circumvented by distributing the decrypted message itself. The goal of a traitor is to make content available to unauthorized users, and there are several ways to accomplish this. Using a coalition of at most t users, a traitor can [JL07]:

a) produce a pirate decoder,

b) provide the decrypted message after decryption,

c) or provide the session key after he has computed it.

Strategy a) is dealt with by traitor tracing schemes. To combat b), the sender must embed watermarks in the message and distribute different variants of the message to different users. To combat c), the scheme must either

i) not use a session key,

ii) prevent extraction of the session key,

iii) or allow the session key to be traced.

Point i) seems to put broadcast encryption based on the hybrid encryption paradigm at a disadvantage. However, all BE schemes use an implicit session key which can be extracted.

Preventing extraction of the session key as required in ii) necessitates assumptions on the hardware or some kind of code obfuscation. Several results suggest that cryptographic code obfuscation is hard to achieve in some generality [HMLS10]. Nonetheless, under the name of "White Box Cryptography" it is being implemented in practice, and several proposals have already been broken [WMGP07]. It may therefore be worth considering what kind of security can provably be achieved.

Tracing a session key as in iii) requires either a special kind of traceable ciphers which allow the derivation of several keys that result in the same behavior [BG03], or the use of watermarks in a message as in counter-strategy b). This allows the tracing authority to use the session keys to decrypt and then trace based on the resulting message. This is what we call *message-based tracing*.

1.3 Overview

We now give an overview of the contributions described in this thesis. Chapter 4 establishes a clean framework of security notions, which we then use to compare the existing notions of security. In chapter 5, we then construct a dynamic broadcast encryption scheme that fulfills the highest security notion. We then construct a message-based traitor tracing scheme with optimal ciphertext rate in chapter 6.

1.3.1 Security Notions for Broadcast Encryption

Since NNL, many BE schemes have been proposed, but for each scheme the security proof was done in a new security model. Because of these various and often *ad-hoc* security models, it is hard to compare the merits of these schemes, as it is not always clear how the security notions relate to each other.

Gentry and Waters [GW09], for example, defined a security notion they call "adaptive", because the adversary can corrupt users adaptively before the challenge phase. But there is a notion that is "even more" adaptive, where the adversary can still corrupt users *after* the challenge phase. One goal of this thesis is thus to provide a better picture of the meaningful security models for BE, and to compare them. In particular, we investigate whether the various adaptive notions of corruption coincide or not.

We define a more systematic security model for broadcast encryption schemes, and construct a generic security framework for BE. We take into account, as usual in the "provable security framework", oracles to model the means available to the adversary, such as the possibility to join new users, to corrupt users, and to decrypt messages. It is worth noting that small details can have a high impact. For example, the choice of the target set for the challenge ciphertext also plays a role in how the models relate to each other. We investigate the relationships between the different notions, and find that in some cases, two notions are equivalent or separated depending on the availability of some oracles or the collusion-resistance of a BE scheme. After describing the relationships between notions in our framework, we have a closer look at the security models and the schemes proposed in the literature, and discuss where they are in our framework, which then helps to compare them.

Our results are relevant for existing BE schemes. From the proof found in [GW09], it is clear that the two-key transformation actually achieves the stronger 2-adaptive security level.

In addition, we define a security notion that is stronger than all previous ones, and give an example of a scheme that fulfills this notion.

1.3.2 Decentralized Dynamic Broadcast Encryption

In 2001, Naor, Naor, and Lotspiech [NNL01] introduced the subset-cover framework, which can be seen as reducing the ciphertext length of the naive BE scheme, which encrypts the message once to each user in the target set, by defining additional keys that are shared by users. If all users who share a key are in the target set, then this key is used once to encrypt, which reduces the ciphertext length compared to encrypting once to each user. The larger the sets, the bigger the efficiency improvement, but also the bigger the risk that one of the users in the set is revoked. The ciphertext length of the subset-difference (SD) version of NNL depends linearly on the number of users in the revoked set, which was considered to be efficient enough for use in the AACS DRM standard [AAC09]. We generalize the subset-cover framework of NNL to deal with both public-key encryption and dynamic changes of the set of registered users. In doing so, we construct the first broadcast encryption scheme that is fully

dynamic and IND-CCA secure under fully adaptive corruptions, in the standard model under the DDH assumption. We furthermore remove the need for trusted authorities by eliminating the group manager, who typically interacts with users to distribute keys at the setup phase or when users join the system. Our approach makes use of group key exchange with subgroup keys [Man09, ACMP10], a primitive that simultaneously distributes different keys to certain subsets of the user group and applies well to the subset-cover framework if one can assign keys for the subgroups involved in the subset cover.

We first instantiate our construction with the Diffie-Hellman key agreement for the interactive setup and the ElGamal encryption for the public-key encryption, which leads to quite an efficient scheme. The complete-subtree (CS) construction resembles the tree-based group key agreement in [KPT04], but we also create key pairs for internal nodes, and we go beyond their scheme in our construction of SD trees. We then show how our scheme can be extended to achieve the strongest security notion by using Cramer-Shoup encryption, which allows adaptive corruptions and chosen-ciphertext attacks, in the standard model, under the DDH assumption. In addition, we consider various criteria of efficiency: ciphertext size, secret part and public part of the decryption keys, number of rounds for the key generation, etc. Thanks to the modularity of our approach, we can use any appropriate group key exchange with subgroup keys: our initial technique iteratively uses the two-party Diffie-Hellman key exchange in a binary tree, which requires a logarithmic number of rounds; we can replace it by logarithmically many parallel executions of the Burmester-Desmedt group key exchange protocol [BD05], which reduces the number of rounds to two. Besides allowing members to join the system, we also sketch how groups could merge at low cost, and how to permanently revoke some users. Our scheme thus achieves a maximum of functionality and security under minimal assumptions, while still being reasonably practical.

Broadcast Encryption vs. Group Key Exchange As already explained, we see broadcast encryption as a broadcast key encapsulation mechanism only, which is used in combination with a symmetric encryption scheme to encrypt messages. From this point of view, the NNL scheme uses long-term symmetric keys to allow a trusted centre to securely send a symmetric session key to all users in the target set, which establishes an ephemeral key.

The scheme contains a second key establishment: Long-term symmetric keys are assigned to each subgroups in the subset cover, and distributed to users by the group manager. Traditionally, this group key distribution is conducted during the setup phase, between the group manager and the users. It is assumed to be performed in a secure way, and thus it is not part of the security model. As an alternative, we can say that this phase requires secure channels between the centre and the users. To avoid such secure channels and trusted authority in the key distribution, group key exchange protocols are a more realistic alternative. We will include this key generation process into the security model, but only against passive adversaries. This can be seen as replacing the "secure channels" assumption during setup with the weaker "authenticated channels" assumption. For security against active adversaries, which is needed to remove the authenticated channels assumption, some additional authentication mechanism will be required; efficient generic compilers exist [KY07], and this is outside the scope of this thesis.

Related Work Broadcast encryption without a central authority replaces the traditional setup with a group key exchange process that can be an interactive protocol. It was proposed under the name "contributory broadcast encryption" (CBE) in [WQZ+11], along with a semi-adaptively IND-CPA secure scheme that is not dynamic. A possible application of this could be communication in a social network, where some private information is meant to be read only by a subset of a user's acquaintances, and the network is either peer-to-peer or the service provider is not trusted. The first step towards subgroup key exchange were done by Manulis [Man09], who extended a group key exchange (GKE) protocol to allow any two users to compute a common key after the initial phase in which the group key is computed. Following this work, Abdalla et al. [ACMP10] generalized this approach to allow the computation of session keys for arbitrary subsets. We use such a group key exchange protocol with subgroup keys to derive asymmetric encryption keys for subsets. Something similar has been done under the name of "asymmetric group key agreement" (ASGKA) [WMS+09]. In [WMS+09], ASGKA is defined in a way that guarantees only that the keys held by the participants are good for use with a specific encryption scheme. We want to generalize this requirement so that at the end of the protocol run, each user has some randomness, which can thereafter be used for any key generation, and namely to generate key pairs for any key encapsulation mechanism. Since this randomness is shared between various subgroups, we call the scheme we use for the setup "subgroup key exchange" (SKE). Kurnio, Safavi-Naini, and Wang [KSNW03] explicitly consider sponsorship of group candidates by existing members. In our scheme, because of the tree structure, each user can act as a sponsor, and only one sponsor is required for a candidate to join

the user set.

1.3.3 Message-based Traitor Tracing

We propose the term "message-based traitor tracing" as a generic term that subsumes earlier variants and emphasizes the fact that we do not trace from pirate decoders but from information embedded in the content. Fiat and Tassa were the first to consider message-based traitor tracing; in [FT99], they developed *dynamic traitor tracing* to deal with traitors that rebroadcast decrypted content. They assume that there is a real-time feedback of the broadcast content to the centre, so that the watermarks can be adapted to the feedback. Safavi-Naini and Wang [SNW03a] noted that in this setting, dynamic TT can be prevented by delaying the rebroadcasting of the content. To take this countermeasure into account, they proposed *sequential traitor tracing*, where the mark allocation is precomputed, but users are removed according to the feedback received. They construct a sequential TT scheme by combining error-correcting codes and watermarking. Jin and Lotspiech [JL07] claimed that protection should not increase the bandwidth by more than 10 %. To solve this problem, they proposed to extend the tracing procedure over several movies (using "inner" and "outer" codes) and assumed that the traitors will not drop any block. Their sequence key block scheme permits the revocation of users after they have been traced through the rebroadcasted messages. Kiayias and Pehlivanoglu [KP09] showed that the sequence key block scheme allows only to trace and to revoke a limited number of users, and proposed a message-trace-and-revoke scheme without this limitation.

Optimal ciphertext rate Contrary to the classical tracing where schemes with optimal ciphertext rate exist, the problem of constructing a scheme with optimal ciphertext size for message-based traitor tracing is still open. We explain why the solutions for classical tracing fail when applied to message-based traitor tracing and we then describe our approach.

Boneh and Franklin [BF99] developed a traitor tracing scheme with a ciphertext size linear in the maximal number of colluding users. Kiayias and Yung [KY02c] further integrated a 2-user version of this scheme with a fingerprinting code into the first TT scheme with a constant ciphertext rate. This method can be summarized as follows. The sender essentially encrypts all the blocks twice, so that the recipient can only decrypt one of the two ciphertexts for each block. The tracing procedure consists in using the decrypted ciphertext or the distributed keys to extract a word associated to the pirate decoder. Thanks to the tracing capability of a fingerprinting code, one can then trace back one of the traitors. Kiayias and Yung's scheme leads to a ciphertext three times bigger than the initial content. Fazio, Nicolosi, and Phan [FNP07] then achieved a ciphertext rate asymptotically 1. Their method is to encrypt just one particular block twice each time and then apply an all-or-nothing transform (AONT), which guarantees that the traitors cannot drop this particular block because missing just one block makes the traitors unable to get any information on the plaintext. The use of AONT in [KY02c, FNP07] is interesting but quite impractical because the receiver should wait until he has received n blocks (where n is the code length of the code in use, and thus quite large) to start the decryption procedure. We note that, without aiming to optimize the ciphertext rate, the use of AONT can be avoided by using robust fingerprinting code which allows traitors to drop a fraction of the positions. This is used in [Sir07, BP08, BN08] to reduce the ciphertext size. However, in order to get optimal ciphertext rate in [FNP07], the use of AONT is compulsory, otherwise the traitors could simply drop the particular block to defeat the tracing procedure.

Focusing now on message-based traitor tracing, one natural question is why we do not simply apply the above method of optimizing the ciphertext rate. We argue that this method cannot work for message-based traitor tracing. We first notice that in all of the above methods for classical tracing, each user finally gets the same plaintext and if a user redistributes this plaintext, we have no way to trace back the traitor from the distributed message. Therefore, the necessary condition for message-based traitor tracing is that each user receives a different (marked) version of the plaintext. However, when the plaintext is different for each user, one cannot apply AONT for a whole fixed plaintext, otherwise all but at most one user can decrypt. The use of AONT for message-based traitor tracing is thus irrelevant. Fortunately, we can still use the method of doubling one particular block by finding out a way to hide this block. Our method consists in using a 2-user anonymous broadcast encryption scheme and then randomly permuting the blocks. With a 2-user anonymous broadcast encryption scheme, the traitor cannot detect any difference between an encryption for both users (which is used for all blocks but the particular block) and an encryption for one of the two users that is used for the particular protected block. Combining with the permutation of the blocks, we can show that the traitor is prevented from detecting the particular

protected block. Moreover, beyond the optimization of ciphertext rate, by not using AONT, our scheme also enjoys the property of the sequential decryption via the use of fingerprinting code as in [BP08,BN08]: The user can sequentially decrypt the sub-ciphertexts, and does not need to wait until he has received the whole ciphertext and can apply the AONT transform to start the decryption procedure.

Robustness We use a realistic adversarial model which takes into account countermeasures traitors might take against being traced by removing watermarking information from the content. The *watermarking* assumption says that it is infeasible for an adversary to detect, remove, or change watermarks unless he has more than one version of the same message. We therefore need to make sure that our scheme reveals no information about the placement of watermarks, which leaves the adversary with the possibility of deleting random parts of the message in the hope of also removing a watermark. We set a threshold δ such that if the adversary drops more than a fraction δ of the message, it becomes useless. We then need to deal with a message where at most a fraction δ of the watermarks are missing. This allows us also to relax the watermarking assumption: Because we know how to deal with erased marks, we need only ask that it is hard (but not infeasible) to remove watermarks.

Chapter 2

Preliminaries

Contents

In this chapter, we lay the theoretical foundations for the discussion of our results. After introducing some notation in section 2.1, we skim the surface of computational complexity theory in section 2.2, which enables us to define what it means for a problem to be "hard". In section 2.3, we give an overview of the provable security paradigm, which requires us to give proofs that relate the security of cryptographic schemes to assumptions about the hardness of certain problems. We define some of these problems and explain basic proof techniques. We then proceed to define symmetric primitives in section 2.4 and asymmetric primitives in section 2.5, before taking a closer look at broadcast encryption in section 2.6 and traitor tracing in section 2.7.

2.1 Notation

$\mathbb{N}$ denotes the set of non-negative integers $\{0, 1, 2, \ldots\}$, $\mathbb{R}_{>0}$ the set of positive reals $\{x \in \mathbb{R} \mid x > 0\}$. For $m, n \in \mathbb{N}, m \leq n$, we define $[m, n] \stackrel{\mathsf{def}}{=} \{m, \ldots, n\}$ and $[n] \stackrel{\mathsf{def}}{=} \{1, \ldots, n\}$, but $[0, 1] \stackrel{\mathsf{def}}{=} \{x \in \mathbb{R} \mid 0 \leq x \leq 1\}$.

For a string $x \in \{0,1\}^*$ and an integer n, x^n denotes the concatenation of n copies of x, and $|x|$ denotes the length of x. For example, $1^n = 111\ldots1$, and $|1^n| = n$.

For a finite set S of size $|S|$, we write $x \xleftarrow{\$} S$ to denote that x is an element of S drawn uniformly at random. For a probabilistic algorithm $\mathcal{A}$, we write $y \leftarrow \mathcal{A}(x)$ to denote that $\mathcal{A}$ outputs y on input x. To make the randomness r used by $\mathcal{A}$ explicit, we write $y = \mathcal{A}(x;r)$. We let $[\mathcal{A}(x)]$ denote the set of all possible outputs of $\mathcal{A}$ on input x.

We often want to talk about asymptotic behaviour of algorithms. We first give a name to series that vanish faster than the inverse of any polynomial.

Definition 2.1 (Negligible function)**.** A series $\nu : \mathbb{N} \to [0,1]$ is called a *negligible function*, if

$$\forall c \in \mathbb{N}\ \exists n_c \in \mathbb{N}\ \forall n > n_c : \nu(n) < n^{-c}.$$

We can use this to define probabilities that are asymptotically so small that we can neglect them.

Definition 2.2 (Negligible probability)**.** We say that the probability of an event $E(\kappa)$ depending on a variable $\kappa \in \mathbb{N}$ is *negligible*, if $\Pr[E(\kappa)]$ is a negligible function. We say that the probability of $E(\kappa)$ is *overwhelming*, if $1 - \Pr[E(\kappa)]$ is a negligible function.

For asymptotic comparisons of functions $f : \mathbb{N} \to \mathbb{N}$, we define the set $\mathcal{O}(f)$ of functions that asymptotically do not grow faster than f, and its opposite, the set $\Omega(f)$ of functions that asymptotically do not grow slower than f.

$$\mathcal{O}(f) \stackrel{\mathsf{def}}{=} \{g \mid \exists \varepsilon \in \mathbb{R}_{>0}\ \exists x_\varepsilon \in \mathbb{N}\ \forall x \geq x_c : g(x) \leq \varepsilon \cdot f(x)\}$$

$$\Omega(f) \stackrel{\mathsf{def}}{=} \{g \mid \exists \varepsilon \in \mathbb{R}_{>0}\ \exists x_\varepsilon \in \mathbb{N}\ \forall x \geq x_c : g(x) \geq \varepsilon \cdot f(x)\}$$

The sets $o(f)$ of functions that asymptotically grow slower than f and the set $\omega(f)$ of functions that asymptotically grow faster then f are defined as

$$o(f) \stackrel{\mathsf{def}}{=} \{g \mid \forall \varepsilon \in \mathbb{R}_{>0}\ \exists x_\varepsilon \in \mathbb{N}\ \forall x \geq x_c : g(x) \leq \varepsilon \cdot f(x)\},$$

$$\omega(f) \stackrel{\mathsf{def}}{=} \{g \mid \forall \varepsilon \in \mathbb{R}_{>0}\ \exists x_\varepsilon \in \mathbb{N}\ \forall x \geq x_c : g(x) \geq \varepsilon \cdot f(x)\}.$$

2.2 Complexity Theory Basics

One of the fundamental notions in cryptology is the hardness of a problem. To formalize this, we recall some notions from complexity theory, using Turing machines as our model of computation. A Turing machine consists of an infinitely long tape on which finitely many non-blank symbols are written, and a tape head that can move as well as read and write symbols. Without loss of generality, we restrict ourselves to the alphabet $\{0,1\}$ and Turing machines with a single band.

Definition 2.3. A (deterministic) *Turing machine* (TM) T is a tuple $(\square, Q, q_0, q_H, \delta)$, where
- $\square$ is a "blank" symbol,
- Q is the finite set of states that T can be in,
- $q_0 \in Q$ is the start state,
- $q_H \in Q$ is the halting state, and
- $\delta : Q \setminus \{q_H\} \times \{0,1\} \cup \{\square\} \to Q \times \{0,1\} \cup \{\square\} \times \{L,S,R\}$ is the transition function that describes the rules T follows in each step.

The TM will read the symbol under the tape head in the starting state, then write a symbol on the tape and move the tape head according to δ. The output of the TM is the content of the tape when it is in the halting state.

We say that a TM T is *polynomial-time*, if there is a polynomial p such that for all inputs x, T halts on input x after at most $p(|x|)$ steps.

Any decision problem can be formalized as deciding whether a string is part of a set of strings.

Definition 2.4. A *language* L is a subset $L \subset \{0,1\}^*$. We say that a Turing machine *decides* a language L, if it computes its characteristic function $\chi_L : \{0,1\}^* \to \{0,1\}, \chi_L(x) = 1 \Leftrightarrow x \in L$.

We can now define the class of problems that is often considered to contain those problems that can be efficiently solved.

Definition 2.5. The class **P** is the set of languages that can be decided by a deterministic polynomial-time Turing machine.

This class is usually contrasted with the class of problems that have efficiently verifiable solutions.

Definition 2.6. A *nondeterministic Turing machine* (NDTM) is a TM with an additional accepting state q_A, and two transition functions δ_0, δ_1, a random one of which is applied in each step. For any input x, $T(x) = 1$ if there is a sequence of steps such that T reaches q_A, and $T(x) = 0$ if every sequence of steps makes T halt without accepting.

We say that a NDTM T is *polynomial-time*, if there is a polynomial p such that for all inputs x and for all possible choices, T halts or accepts on input x after at most $p(|x|)$ steps.

Definition 2.7. The class **NP** is the set of languages that can be decided by a nondeterministic polynomial-time Turing machine.

P is sometimes thought of as the class of efficiently solvable problems. When talking about hard problems, we cannot assume that all problems outside of **P** are hard, because it would be too restrictive to say that a problem is hard when it cannot be solved with certainty. For this reason, we introduce a probabilistic class, where it is permissible to use randomness to solve problems most of the time.

Definition 2.8. A *probabilistic Turing machine* (PTM) is a TM with two transition functions δ_0, δ_1. In each step, the function to be applied is determined at random with probability 1/2, independently of previous choices. The TM only outputs 0 or 1. We denote by $T(x)$ the random variable corresponding to T's output.

We say that a PTM T is *polynomial-time*, if there is a polynomial p such that for all inputs x, T halts on input x after at most $p(|x|)$ steps, regardless of the choices it makes.

Definition 2.9. The class **BPP** is the set of languages that can be decided by a probabilistic polynomial-time TM with probability at least $\frac{2}{3}$.

Obviously $\mathbf{P} \subseteq \mathbf{BPP}$, but it is unknown whether the inclusion is strict. It is entirely possible that $\mathbf{P} = \mathbf{BPP}$. While we can say that problems in **BPP** are "easy", we cannot say that problems in **NP** are "hard", because $\mathbf{P} \subseteq \mathbf{NP}$. In order to identify the hard problems in **NP**, we first define a way to meaningfully state that a problem h is harder than a problem e.

Definition 2.10. We say that a function $f : \{0,1\}^* \to \{0,1\}^*$ is *polynomial-time computable*, if there is a deterministic polynomial-time TM T such that for all $x \in \{0,1\}^*$, $T(x) = f(x)$.

Definition 2.11. A language L_e is *polynomial-time reducible* to a language L_h, written as $L_e \leq_p L_h$, if there is a polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that for every $x \in \{0,1\}^*$, $x \in L_e$ if and only if $f(x) \in L_h$.

For a complexity class C, we say that L_h is C-hard, if $L \leq_p L_h$ for all $L \in C$. We say that L_h is C-complete, if L_h is C-hard and $L_h \in C$. This means that the C-complete languages encode the hardest problems in C.

We believe in the correctness of the the Church-Turing thesis and will from now on refer only to "algorithms" instead of Turing machines and "problems" instead of languages.

2.3 Provable Security

Until the end of the 20th century, cryptography consisted of ingenious people proposing ways to encrypt messages, which were then broken by other ingenious people. To gain some justified confidence that a cryptographic scheme will not be broken, it is desirable to have a formal proof of its security.

Sometimes cryptographers make strong assumptions about some components of cryptographic schemes, which can lead to more efficient schemes while excluding classes of adversaries. We present the most common idealized models in section 2.3.1. Since security of even the weakest cryptographic primitives implies that $\mathbf{P} \neq \mathbf{NP}$, which would resolve a longstanding open problem, we need to make some assumptions on which we can base our proof. We describe some of these assumptions in section 2.3.2. Proofs will therefore take the form of complexity-theoretic reductions, showing that any adversary who can break the scheme can also solve another problem. In this case, we say that breaking the scheme is at least as hard as solving the problem. If this problem is hard, in the sense that there is no efficient algorithm that solves it, then there is also no efficient algorithm that can break the scheme, where "breaking" a

scheme needs to be carefully defined for each scheme. In section 2.3.3, we will describe some standard proof techniques.

A security proof of this kind is only a statement about the asymptotic behaviour, which means that the scheme is secure for a sufficiently large security parameter. To help choose concrete parameters, Bellare, Kilian, and Rogaway [BKR00] proposed to consider the *tightness* of a security reduction, and compare the resources used by adversaries against the scheme and against the hardness assumption. We will expand on this in section 2.3.4.

2.3.1 Idealized Models

In the so-called *standard model*, the adversary can be any algorithm running in probabilistic polynomial time (PPT). It can be easier to design efficient schemes when considering only security against a class of adversaries that are "generic" in some sense. To accomplish this, a security proof is given in a model that idealizes some aspect of a scheme. Such a result is weaker than a proof in the standard model without these idealizations, and guarantees only that in order to break the scheme's security, an adversary must take into consideration an aspect of a scheme's instantiation that contravenes the idealization.

A security proof in the random oracle model guarantees security against all adversaries that model a hash function used in the scheme as a random function; a successful adversary must exploit some property of the concrete hash function that is used. The generic group model covers generic attacks that do not take into account some specifics of the representation of elements in an algebraic group.

The Random Oracle Model

A random oracle is an oracle which is available to all parties in a protocol and which models a truly random function. When it is queried on some input, it first checks whether it was queried on the same input before. If this is the case, it returns the same output as the last time. If not, it draws an element uniformly at random from its output domain, adds the tuple (input, output) to its query list, and returns the output.

Random oracles have been used in cryptography by Fiat and Shamir [FS87] to remove interaction from protocols. The random oracle model (ROM) was introduced to cryptography by Bellare and Rogaway [BR93] to formalize assumptions about hash functions and allow the construction of efficient protocols.

Bellare and Rogaway [BR93, sec. 1.1] noted that no real function is like a random oracle, because is has a short description. This was exploited by Canetti, Goldreich, and Halevi [CGH04] to show that there are signature and encryption schemes that are provably secure in the ROM but insecure for any instantiation of the random oracle. Nielsen [Nie02] showed that non-interactive non-committing encryption exists in the ROM, but that this is due to the fact that the random oracle can be programmed. In the security proof, the simulator can choose the outputs of the random oracle, and he only needs to do so when the oracle is queried. Nielsen proposed the non-programmable random oracle model (NPROM), and shows that there is no non-interactive non-committing encryption scheme in the NPROM (and by extension also in the standard model).

The Generic Group Model

The generic group model (GGM) abstracts from the concrete representation of an algebraic group and so excludes algorithms that rely on specific properties of the representation of group elements.

We consider the additive group $\mathbb{Z}_n$ and let $S \subseteq \{0,1\}^*$ be a set with $|S| \geq n$. An *encoding function* from $\mathbb{Z}_n$ to S is an injective map $\sigma : \mathbb{Z}_n \to S$. The addition oracle $\mathcal{O}_+ : S \times S \times \{0,1\} \to S$ is defined as $\mathcal{O}(\sigma(x_i), \sigma(x_j), b) = \sigma(x_i + (-1)^b x_j)$.

The generic-group model was proposed by Shoup [Sho97] to give tight bounds on the complexity of "generic algorithms" that solve computational problems. We fix a function $l : \mathbb{N} \to \mathbb{N}$ with $l(x) \geq x$. For the set $S = \{0,1\}^{l(\lfloor \log n \rfloor + 1)}$ an encoding function $\sigma : \mathbb{Z}_n \to S$ is chosen at random. A *generic algorithm* is an algorithm $\mathcal{A}^{\mathcal{O}_+(\cdot,\cdot,\cdot)}(\sigma(x_1), \ldots, \sigma(x_k))$. For a generic algorithm, it is possible to efficiently test for equality of group elements (due to the injectivity of σ) and to perform additions, inversions (assuming the list of representations contains $\sigma(0)$), and subtractions using the oracle.

The GGM is often used to do a "sanity-check" of hardness assumptions. If an assumption does not even hold in the generic group model, it is obviously false. Dent [Den02] showed that for a carefully chosen oracle, the oracle discrete-logarithm problem is hard in the GGM, but easy when the random

$\mathsf{Gen}(1^\kappa)$
$p, q \xleftarrow{\$} \mathbb{P} \cap \{0,1\}^\kappa$;
$n = pq$;
$\varphi(n) = (p-1)(q-1)$;
$e \xleftarrow{\$} \mathbb{Z}_n^*$;
$d \equiv e^{-1} \mod \varphi(n)$;
`return` n, e, d;

$\mathrm{Exp}^{\mathsf{rsa}}_{\mathsf{Gen},\mathcal{A}}(\kappa)$
$(n, e, d) \leftarrow \mathsf{Gen}(1^\kappa)$;
$y \xleftarrow{\$} \mathbb{Z}_n^*$;
$x \leftarrow \mathcal{A}(n, e, y)$;
`if` $x^e \equiv y \mod n$ `then return` 1;
`else return` 0;

Figure 2.1: RSA key generation and the RSA experiment.

encoding function is replaced by any encoding ensemble (a collection of actual encoding functions). He extended this result to show that there is a signature scheme that is secure in the GGM but insecure in any concrete group.

The generic bilinear group model is an extension of the GGM to groups equipped with bilinear maps. We have three encodings $\sigma_1 : \mathbb{Z}_n \to S_1, \sigma_2 : \mathbb{Z}_n \to S_2, \sigma_T : \mathbb{Z}_n \to S_T$, oracles that compute the group operation for each of the groups, and an additional oracle $\mathcal{O}_e : S_1 \times S_2 \to S_T$ that computes the pairing.

The generic ring model is an extension of the GGM to rings, and was used to study hardness assumption defined over rings. Aggarwal and Maurer [AM09] showed that in the generic ring model, breaking RSA in $\mathbb{Z}_n$ is equivalent to factoring n. Subsequently, Jager and Schwenk [JS13] proved in the generic ring model that computing the Jacobi symbol of an integer modulo n is equivalent to factoring n. Since the Jacobi symbol can be computed efficiently using non-generic algorithms, this is an example of a computational problem that is hard in the GRM, but easy in practice.

2.3.2 Hardness Assumptions

We present some hardness assumptions that we will reference in later chapters. One of the most well-known problems is the RSA problem.

The RSA problem The problem of factoring large integers has occupied mathematicians for at least two thousand years, and until now no polynomial-time algorithm (not using quantum computers) has been found. This suggests that factoring large integers is hard, which is why this problem was used in one of the first public-key cryptosystems. The decision version of the factoring problem asks, given two integers n, m, whether there is an integer $i \in [m]$ with $\gcd(i, n) \neq 1$. Equivalently, factoring can be stated as the function problem of, given an integer n, finding a factor of n.

The RSA cryptosystem, which is now better described as a trapdoor one-way permutation (TOWP, see section 2.5.1), has at its core two randomly chosen large primes p and q of equal bit-length. The public key is $n = pq$ and an integer e that is coprime to n, which guarantees that e is invertible in $\mathbb{Z}_n^*$. The order of $\mathbb{Z}_n^*$ is $\varphi(n) = (p-1)(q-1)$, and knowing $\varphi(n)$ allows inverting e efficiently. Together with the modulus n, the inverse $d \equiv e^{-1} \mod \varphi(n)$ forms the secret key. This key generation procedure is depicted in figure 2.1. Then the RSA permutation is $\pi_{(e,n)}(x) \equiv x^e \mod n$ and its inverse is $\pi^{-1}_{(d,n)}(x) \equiv x^d \mod n$.

It was sometimes said that the security of this TOWP rests on the factoring assumption, which is ambiguous. Breaking the scheme in the sense of computing the secret key from the public key is equivalent to the factoring assumption [May04], but computing x from $x^e \mod n$ is not known to be equivalent to factoring. This problem is known as the RSA problem, and we define it using the experiment in figure 2.1.

Definition 2.12 (RSA problem)**.** Let Gen be an RSA generator. We say that the *RSA problem* is hard relative to Gen, if for all PPT algorithms $\mathcal{A}$, the success probability

$$\mathsf{Succ}^{\mathsf{rsa}}_{\mathsf{Gen},\mathcal{A}}(\kappa) = \Pr[\mathrm{Exp}^{\mathsf{rsa}}_{\mathsf{Gen},\mathcal{A}}(\kappa) = 1]$$

is negligible in κ.

$\mathrm{Exp}^{\mathsf{dlog}}_{\mathbf{G},\mathcal{A}}(\kappa)$
$(q, \mathbb{G}) \leftarrow \mathbf{G}(1^\kappa)$;
$g \xleftarrow{\$} \mathsf{Gens}(\mathbb{G})$;
$h \xleftarrow{\$} \mathbb{G}$;
$x \leftarrow \mathcal{A}(q, \mathbb{G}, g, h)$;
`if` $g^x = h$ `then return` 1;
`else return` 0;

(a) The DLOG experiment

$\mathrm{Exp}^{\mathsf{ddh}\text{-}b}_{\mathbf{G},\mathcal{A}}(\kappa)$
$(q, \mathbb{G}) \leftarrow \mathbf{G}(1^\kappa)$;
$g \xleftarrow{\$} \mathsf{Gens}(\mathbb{G})$;
$(x, y, z) \xleftarrow{\$} [q]$;
$(g_0, g_1) \stackrel{\mathsf{def}}{=} (g^z, g^{xy})$;
$b' \leftarrow \mathcal{A}(1^\kappa, g, g^x, g^y, g_b)$;
`return` b';

(b) The DDHexperiment

Figure 2.2: The DLOG and DDH experiments

Clearly the factoring problem is at least as hard as the RSA problem, as factoring n allows computing d from e.

Currently, the most efficient algorithm for factoring a large integer n is the general number field sieve (GNFS), which was used to factorize a 768-bit RSA number in 2009 [KAF+10]. The GNFS always returns a result and has a running time of asymptotically

$$Ae^{(C+o(1))(\ln n)^{1/3}(\ln\ln n)^{2/3}}$$

for constants A and $C = \left(\frac{64}{9}\right)^{1/3}$. Substituting the bit-length of n, $\lambda = \log n = \frac{\ln n}{\ln 2}$, we obtain a running time of

$$Ae^{(C+o(1))(\lambda \ln 2)^{1/3}(\ln(\lambda \ln 2))^{2/3}},$$

which shows more clearly that the running time is superpolynomial in λ. Since no way of breaking the RSA problem without factoring n is known, we assume that RSA is as hard as factoring.

Hardness Assumptions in Cyclic Groups

The discrete logarithm problem One of the first problems that cryptographers tried to leverage in order to obtain secure public-key cryptosystems was the discrete logarithm problem. Let $\mathbf{G}$ be a generator of (multiplicatively written) cyclic groups, which on input 1^κ outputs a tuple $(q, \mathbb{G})$, where $|q| = \kappa$ and $\mathbb{G}$ is a cyclic group of order q. We assume that the order of a group is part of its description. We denote by $\mathsf{Gens}(\mathbb{G})$ the set of generators of $\mathbb{G}$ and assume that it is easy to draw a generator uniformly at random. The DLOG-experiment is defined in figure 2.2a, and we say that DLOG in $\mathbf{G}$ is hard if every PPT adversary has only a negligible chance to win the experiment.

Definition 2.13 (DLOG)**.** Let $\mathbf{G}$ be a generator of cyclic groups. We say that the discrete logarithm problem is hard in $\mathbf{G}$ if for all PPT algorithms $\mathcal{A}$, the success probability

$$\mathsf{Succ}^{\mathsf{dlog}}_{\mathbf{G},\mathcal{A}}(\kappa) = \Pr[\mathrm{Exp}^{\mathsf{dlog}}_{\mathbf{G},\mathcal{A}}(\kappa) = 1]$$

is negligible in κ.

The fastest known algorithm to solve DLOG in finite fields is a variant of the GNFS. In elliptic curves, the generic Pollard's rho algorithm is the fastest known algorithm with a running time of $\mathcal{O}(\sqrt{p})$, or $\mathcal{O}(2^{|p|/2})$, where p is the largest prime factor of the group order. No algorithm with sub-exponential running time is known to exist in the case of elliptic curves.

The discrete logarithm problem was first used in the context of the Diffie-Hellman key exchange protocol, where to parties draw secret scalars x, y and exchange their public keys g^x, g^y. Their common key is then $(g^y)^x = g^{xy} = (g^x)^y$. The situation here is similar to the one for RSA: Computing secret keys from public keys is equivalent to the discrete logarithm problem, but computing the secret key g^{xy} from the public keys g^x, g^y is known as the *computational Diffie-Hellman problem* (CDH). DLOG is at least as hard as CDH, and the two are not known to be equivalent, but this is often assumed.

The decisional Diffie-Hellman problem A classical number-theoretic problem to base cryptography on is the decisional Diffie-Hellman problem. For a generator of cyclic groups $\mathbf{G}$, $\mathsf{DDH}_{\mathbf{G}}$ is the decision problem for the language $\{(1^\kappa, g, g^x, g^y, g^{xy}) \mid \kappa \in \mathbb{N}, (q, \mathbb{G}) \in [\mathbf{G}(1^\kappa)], g \in \mathsf{Gens}(\mathbb{G}), x, y \in [q]\}$. It is generally believed that for p, q prime, $p = 2q + 1$, $\mathbb{G}$ the subgroup of quadratic residues in $\mathbb{Z}_p^*$ of order q, $\mathsf{DDH}_{\mathbf{G}}$ is not in **BPP**.

This fact alone is not sufficient to base cryptography on. It is conceivable that only some instances of DDH are hard, while on average, DDH-instances can be solved efficiently. In other words, we are interested in the average-case complexity of problems, not in the worst-case complexity. It is therefore desirable to be able to reduce any instance of a problem to a random instance, which implies that the problem is as hard on average as it is in the worst case. This property is known as *random self-reducibility*. DDH is random self-reducible [Sta96]: Given an instance (g, X, Y, Z) in a group $\mathbb{G}$ of order q, draw random exponents $a, b, c \in [q]$. Then $(g, X^a g^b, Y g^c, Z^a X^{ac} Y^b g^{bc})$ is a new instance that is a DDH-tuple if and only if the original tuple was a DDH-tuple, and the three last elements are distributed uniformly in $\mathbb{G}$ and are independent of the original tuple. Note that since we generate three new elements, we need to add three new random values to ensure that the tuple is information-theoretically independent.

If there is an algorithm $\mathcal{A}$ that solves DDH for a significant fraction of the instances in $\mathbb{G}$, then $\mathcal{A}$ can be used to solve DDH for arbitrary instances in $\mathbb{G}$. If $X = g^x, Y = g^y, Z = g^{xy}$, then $X^a g^b = g^{ax+b}$ and $Y g^c = g^{y+c}$, so $Z^a X^{ac} Y^b g^{bc} = g^{axy+axc+by+bc} = g^{(ax+b)(y+c)}$, and because the new tuple is uniform in $\mathbb{G}$, it is possible to derive random instances from (g, X, Y, Z) until $\mathcal{A}$ breaks one of the random instances.

We rewrite the definition of the DDH problem to make it more amenable for use in proofs, and to be able to make exact statements about how hard the problem is. In figure 2.2b, we define two experiments, that tests an adversary's ability to distinguish between a DDH tuple (when $b = 1$) and a random tuple (when $b = 0$). We call his probability of correctly guessing which sort of tuple he received his *advantage* in winning the game.

Definition 2.14 (DDH)**.** Let $\mathbf{G}$ be a generator of cyclic groups. We say that the *decisional Diffie-Hellman problem* is hard in $\mathbf{G}$ if for all PPT algorithms $\mathcal{A}$, the advantage function

$$\mathrm{Adv}^{\mathsf{ddh}}_{\mathbf{G},\mathcal{A}}(\kappa) = |\Pr[\mathrm{Exp}^{\mathsf{ddh\text{-}0}}_{\mathbf{G},\mathcal{A}}(\kappa) = 1] - \Pr[\mathrm{Exp}^{\mathsf{ddh\text{-}1}}_{\mathbf{G},\mathcal{A}}(\kappa) = 1]|$$

is negligible in κ.

Hardness Assumptions in Bilinear Groups

Many advanced cryptographic schemes are based on groups equipped with bilinear maps, that is groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of the same order for which a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ exists. In order to be able to talk about asymptotics, we define a generator of groups with a bilinear map $\mathbf{G}$ which takes as input a security parameter in unary and outputs a description $(n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ of groups of order n with a bilinear map e. Security of a scheme will rest on specific hardness assumptions, but as a minimum, we need the DLP to be hard in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$.

A bilinear map is called *symmetric*, when there is an efficiently computable homomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ and in the other direction. The presence of a symmetric bilinear map makes the DDH problem in $\mathbb{G}_1 \approx \mathbb{G}_2$ easy, since it need only be checked whether $e(g^x, g^y) = e(g, g^z)$.

The decisional bilinear Diffie-Hellman exponent problem The l-decisional bilinear Diffie-Hellman exponent assumption (l-BDHE) was introduced by Boneh, Boyen, and Goh [BBG05, sec. A.1 of the full version].

Definition 2.15 (BDHE)**.** Let $\mathbf{G}$ be a generator of bilinear-map groups together with a symmetric bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. We say that the *$l$-decision bilinear Diffie-Hellman exponent problem* is hard in $\mathbf{G}$, if for all PPT algorithms $\mathcal{A}$, the advantage in winning the l-BDHE-experiment defined in figure 2.3,

$$\mathrm{Adv}^{l-\mathsf{bdhe}}_{\mathbf{G},\mathcal{A}}(\kappa) = |\Pr[\mathrm{Exp}^{l\text{-}\mathsf{bdhe\text{-}0}}_{\mathbf{G},\mathcal{A}}(\kappa) = 1] - \Pr[\mathrm{Exp}^{l\text{-}\mathsf{bdhe\text{-}1}}_{\mathbf{G},\mathcal{A}}(\kappa) = 1]|$$

is negligible in κ.

The generalized decisional bilinear Diffie-Hellman exponent problem The (P, Q, f)-GBDHE assumption was first introduced by [BBG05, sec. A.2 of the full version] as a generalization of the BDHE assumption.

$\mathrm{Exp}^{l\text{-bdhe-}b}_{\mathbf{G},\mathcal{A}}(\kappa)$
$(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \leftarrow \mathbf{G}(1^\kappa);$
$g \xleftarrow{\$} \mathsf{Gens}(\mathbb{G}_1);$
$h \xleftarrow{\$} \mathsf{Gens}(\mathbb{G}_2);$
$\alpha, z \xleftarrow{\$} [q];$
$\vec{g}_\alpha \stackrel{\mathsf{def}}{=} (g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^l}, g^{\alpha^{l+2}}, \ldots, g^{\alpha^{2l}});$
$(T_0, T_1) \stackrel{\mathsf{def}}{=} (e(g^z, h), e(g^{\alpha^l}, h));$
$b' \leftarrow \mathcal{A}(1^\kappa, g, h, \vec{g}_\alpha, T_b);$
`return` $b';$

Figure 2.3: The BDHE experiment

Let p be a prime, $s, n \in \mathbb{N}$. We define two s-tuples of n-variate polynomials $P = (p_1, \ldots, p_s)$ and $Q = (q_1, \ldots, q_s)$, with $p_1 = q_1 = 1$, and a polynomial f, where $\forall i, k : p_i, q_k, f \in \mathbb{F}_p[X_1, \ldots, X_n]$. Let $g^P = (g^{p_1}, \ldots, g^{p_s})$. We say that f is independent of (P, Q) if there are no constants $a_{i,j}$ and b_k such that it can be written as $f = \sum_{i,j=1}^{s} a_{i,j} p_i p_j + \sum_{k=1}^{s} b_k q_k$.

Definition 2.16 (GBDHE)**.** Let $\mathbf{G}$ be a generator of bilinear-map groups together with a symmetric bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, $(p, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathbf{G}(1^\kappa)$. The *generalized decision bilinear Diffie-Hellman exponent (GBDHE)* problem is defined as follows: Let g be a random generator of $\mathbb{G}$, $g_T \stackrel{\mathsf{def}}{=} e(g, g)$, $r, x_1, \ldots, x_n \xleftarrow{\$} \mathbb{F}_p, \vec{x} \stackrel{\mathsf{def}}{=} (x_1, \ldots, x_n)$. We define the advantage of an algorithm in solving the (P, Q, f)-GBDHE problem as

$$\mathrm{Adv}^{\mathsf{gddhe}}_{\mathbf{G},\mathcal{A}}(P, Q, f) = |\Pr[\mathcal{A}(g^{P(\vec{x})}, g_T^{Q(\vec{x})}, g_T^{f(\vec{x})}) = 1] - \Pr[\mathcal{A}(g^{P(\vec{x})}, g_T^{Q(\vec{x})}, g_T^{r}) = 1]|,$$

where the probability is over the random choice of g, $r, x_1, \ldots, x_n$, and the random bits of $\mathcal{A}$.

The GBDHE assumption says that it is hard to solve the GBDHE problem if f is independent of (P, Q).

We have a lower bound on the hardness of the (P, Q, f)-GBDHE problem in the generic bilinear group model. We let d_f denote the total degree of f, $d_P = \max\{d_f : f \in P\}$.

Theorem 2.17 (Th. A.2 in the full version of [BBG05])**.** *Let $P, Q \in \mathbb{F}_p[X_1, \ldots, X_n]^s, f \in \mathbb{F}_p[X_1, \ldots, X_n]$, $d = \max\{2d_P, d_Q, d_f\}$, $\sigma : \mathbb{Z}_p \to S, \sigma_T : \mathbb{Z}_p \to S_T$ two encodings, $\mathbb{G} = \{\sigma(x) | x \in \mathbb{Z}_p\}, \mathbb{G}_T = \{\sigma_T(x) | x \in \mathbb{Z}_p\}$. If f is independent of (P, Q), then for any $\mathcal{A}$ that makes a total of at most q queries to the oracles computing the group operations and the bilinear map, we have: If $b \xleftarrow{\$} \{0, 1\}, x_1, \ldots, x_n, y, \xleftarrow{\$} \mathbb{F}_p, t_b \leftarrow f(x_1, \ldots, x_n), t_{1-b} \leftarrow y$,*

$$|\Pr[\mathcal{A}(p, \sigma(P(x_1, \ldots, x_n)), \sigma_T(Q(x_1, \ldots, x_n)), \sigma_T(t_0), \sigma(t_1)) = b] - \frac{1}{2}| \leq \frac{(q + 2s + 2)^2 \cdot d}{2p}.$$

2.3.3 Game-based Proofs

A common first step to prove a security property S of a cryptographic scheme Π is to give the security definition as a game in which the adversary $\mathcal{A}$ interacts with an instance of Π, similar to the way we defined computational problems in section 2.3.2. Typically one experiment, $\mathrm{Exp}^{\mathsf{real}}_{\Pi,\mathcal{A}}(\kappa)$, models the interaction of the adversary with the scheme in the real world, the other one, $\mathrm{Exp}^{\mathsf{ideal}}_{\Pi,\mathcal{A}}(\kappa)$, models an ideal world where the security property clearly holds, e. g. an encryption scheme outputs ciphertexts that are independent of the encrypted message, and the experiment outputs a single bit to indicate whether the adversary has won. This means that we can model the output of the game as a random variable, and define the advantage of the adversary in distinguishing between the two experiments as

$$\mathrm{Adv}^{S}_{\Pi,\mathcal{A}}(\kappa) = |\Pr[\mathrm{Exp}^{\mathsf{real}}_{\Pi,\mathcal{A}}(\kappa) = 1] - \Pr[\mathrm{Exp}^{\mathsf{ideal}}_{\Pi,\mathcal{A}}(\kappa) = 1]|.$$

We then need to show that the advantage of any PPT adversary in winning the experiment is negligible in the security parameter κ.

A proof by reduction reduces the problem of breaking the security of the scheme to the task of solving a presumably hard problem P, e. g. DDH. This is accomplished by replacing the experiment with a PPT *challenger* $\mathcal{C}$, who has an identical interface, but takes as input an instance I of P. It is important that the adversary cannot distinguish between interacting with the experiment and the challenger. For example, we might want that $\mathcal{C}(g, g^x, g^y, g^{xy})$ is indistinguishable from $\mathrm{Exp}^{\mathsf{real}}$, and $\mathcal{C}(g, g^x, g^y, g^z)$ is indistinguishable from $\mathrm{Exp}^{\mathsf{ideal}}$. Now, when $\mathcal{A}$ wins the security game, it can distinguish between the two experiments, and therefore also between interacting with $\mathcal{C}(I)$ for a DDH- and a random tuple I. Now, the combination of $\mathcal{A}$ and $\mathcal{C}$ solves DDH, so we can also think of $\mathcal{C}$ as an "adapter" that turns an adversary $\mathcal{A}$ against Π into an adversary $\mathcal{A}||\mathcal{C}$ against DDH. Since by assumption, no successful adversaries against DDH exist, and $\mathcal{C}$ exists by construction, $\mathcal{A}$ cannot exist. Moreover, we can directly relate the maximal advantage of any PPT adversary against the security of Π to the maximal advantage of any PPT adversary against $\mathsf{DDH}_{\mathbf{G}}$.

$$\mathrm{Adv}^{S}_{\Pi}(\kappa) \leq \mathrm{Adv}^{\mathsf{ddh}}_{\mathbf{G}}(\kappa)$$

What we just described is of course the ideal case, where we get a *tight* reduction to the hardness assumption, with a small constant factor. Sometimes the challenger needs to guess where to embed the instance I, in which case we "lose" a factor in the reduction that corresponds to the probability of $\mathcal{C}$ guessing correctly.

Especially when analysing constructions that consist of real-world primitives, there are no asymptotics involved, since the key lengths of primitives are already fixed. To deal with this, Bellare, Kilian, and Rogaway [BKR00] pioneered the *concrete security* approach, where Adv is no longer a function from $\mathbb{N}$ to $[0, 1]$, but a concrete probability. $\mathrm{Adv}^{S}_{\Pi}(t, q) \in [0, 1]$ denotes the maximal success probability against the S security of a scheme Π of any adversary running in time at most t and asking at most q queries to some oracle (which we leave unspecified, but which will be defined for any concrete security notion S). Security reductions will then be of the form

$$\mathrm{Adv}^{S}_{\Pi(p,g)}(t, q) \leq \mathrm{Adv}^{\mathsf{ddh}}_{(p,g)}(t')$$

for a concrete problem instance (p, g) and t' will be given in terms of t and q.

The adversary's use of resources can of course also be considered in asymptotic statements, where $\mathrm{Adv}^{S}_{\Pi}(\kappa, t, q)$ is again a function from $\mathbb{N}$ to $[0, 1]$, and t and q are functions in kappa, from $\mathbb{N}$ to $\mathbb{N}$.

For complex schemes, direct proofs by reduction can become complex and hard to read. To better structure proofs, a good approach is to proceed by sequences of games.

A standard way to accomplish this was formalized independently by Bellare and Rogaway [BR04] and Shoup [Sho04]. It consists of defining a sequence of games $G_0, G_1, \ldots, G_n$ of length at most polynomial in the security parameter, the first of which is the experiment $\mathrm{Exp}^{\mathsf{real}}_{\Pi,\mathcal{A}}(\kappa)$, while the last game is the experiment $\mathrm{Exp}^{\mathsf{ideal}}_{\Pi,\mathcal{A}}(\kappa)$. It is clear that the advantage of the adversary is bounded by the sum of the differences between any two successive games. It is then only necessary to show that the success probability of the adversary changes only by at most a negligible amount between any two successive games, to infer that the advantage of the adversary in winning the security game is negligible.

To make the proof easy to understand, changes between successive games should be kept small. Shoup identifies three types of transitions that can occur between successive games.

Transitions based on indistinguishability A game is derived from another by substituting a probability distribution with another that is indistinguishable in such a way that detecting the change implies an efficient distinguisher. For example, under the right circumstances a game G_{i+1} might be derived from G_i by changing a value g^{xy} to a random group element and the difference between the two games could be bounded by $\mathrm{Adv}^{\mathsf{ddh}}_{\mathbf{G},\mathcal{A}}(\kappa)$, the advantage of $\mathcal{A}$ in winning the DDH-experiment.

Transitions based on failure events In [BR04], games G_i, G_{i+1} are called *identical-until-bad*, if they proceed identically until a flag *bad* is set. Then

$$|\Pr[G_i \to 1] - \Pr[G_{i+1} \to 1]| \leq \Pr[G_i \text{ sets } bad].$$

This is known as the *difference lemma* [Sho04] or the *fundamental lemma of game-playing* [BR04] and it allows us to change a game for cases that happen only with a negligible probability, which is particularly useful when dealing with intractability, e. g. aborting a game if the adversary finds a collision in a collision-resistant hash function.

Bridging steps It is generally desirable that the description of a game differs only little from the description of the preceding game. Intermediate games may therefore change some description into an equivalent description without changing probability distributions.

2.3.4 Choosing Parameters

When we want to compare the efficiency, of cryptographic schemes, we need to find the exact number of group elements e. g. in a ciphertext, which can be extracted from the description of a scheme, and the size of a group element, which depends on the assumptions we make on the group. We state now some simple guidelines we will follow in determining parameters for schemes we find in the literature.

We usually define the security level κ of a scheme as the logarithm of the number of operations an adversary has to execute to break the scheme, i. e. a scheme with a security level of 100 can be broken in 2^{100} steps, but the success probability drops of rapidly if less steps are executed. For symmetric schemes, κ is the length of a key in bits. We therefore sometimes use "bit" as the unit of security. For a secure symmetric scheme, which can only be broken by exhaustively searching the key space of size 2^κ, we have the advantage function $\mathrm{Adv}(\kappa, t) = 2^{-\kappa+\log t}$, where t is the number of steps executed by an adversary.

The actual key length of a cryptographic scheme depends on the underlying hardness assumption and the tightness of the security reduction. For example, imagine a scheme Π whose S security (for some security notion S) is based on the hardness of the $\mathsf{DDH}_{\mathbf{G}}$ problem in a cyclic group generated by a generator $\mathbf{G}$. We assume that we have a security proof which relates the advantage $\mathrm{Adv}^S_\Pi(\kappa, t)$ of any t-time adversary in breaking the scheme to the advantage of any t-time adversary in distinguishing DDH tuples from random tuples via

$$\mathrm{Adv}^S_\Pi(\kappa, t) \leq n\mathrm{Adv}^{\mathsf{ddh}}_{\mathbf{G}}(\kappa, t)$$

for some number n. If we want to bound $\mathrm{Adv}^S_\Pi(\kappa, t)$ by, say 2^{-80}, we have to choose parameters for the group such that $\mathrm{Adv}^{\mathsf{ddh}}_{\mathbf{G}}(\kappa, t) \leq 2^{-(80+\log n)}$. We now look at how the hardness assumption influences key length.

RSA

We do not know any algorithm that breaks RSA in $\mathbb{Z}_n$ without factoring n, so we look at factoring algorithms to determine RSA key lengths. Let λ denote the length of an RSA key, κ the length of a symmetric key of an equivalent security level. While the notion of "computational step" is certainly not equivalent in the two settings, we can nonetheless derive the following relationship between the two from the running time of the general number field sieve [ECR12, sec. 6.2.1]. We first set the number of steps needed by the GNFS equal to 2^κ, treating the $o(1)$-term as zero.

$$2^\kappa = A \cdot \exp\left[\left(\frac{64}{9}\right)^{1/3} (\lambda \ln 2)^{1/3} (\ln(\lambda \ln 2))^{2/3}\right]$$

The constant $A = 2^{-14}$ is chosen based on comparisons between the running time of algorithms to break DES and RSA [ECR12, sec. 6.2.1].

$$\begin{aligned}
\kappa &= \frac{1}{\ln 2}\left(\frac{64}{9}\right)^{1/3} (\lambda \ln 2)^{1/3} (\ln(\lambda \ln 2))^{2/3} - 14 \\
(\kappa + 14)^3 &= \left(\frac{1}{\ln 2}\right)^3 \left(\frac{64}{9}\right) (\lambda \ln 2)(\ln(\lambda \ln 2))^2 \\
\left(\frac{9 \ln^2 2}{64}\right)(\kappa + 14)^3 &= \lambda (\ln(\lambda \ln 2))^2
\end{aligned}$$

To obtain (rough) upper and lower bounds, we apply the inequalities $\sqrt{\lambda} > \ln \lambda + \ln \ln 2 > 1$ for $\lambda \geq 4 > \frac{e}{\ln 2}$.

For an upper bound, we can derive $0.07(\kappa + 14)^3 \geq \lambda$ for $\lambda \geq 4 > \frac{e}{\ln 2}$, which means that $\lambda \in \mathcal{O}(\kappa^3)$, and due to the $\ln \lambda$-term we eliminated in the simplification, we even have $\lambda \in o(\kappa^3)$. For a lower bound, we have $0.06(\kappa+14)^3 \leq \lambda^2$, or $\lambda \in \Omega(\kappa^{3/2})$ and due to the substitution of $\ln \lambda$ with $\sqrt{\lambda}$ even $\lambda \in \omega(\kappa^{3/2})$. We note that the GNFS does not have the same linearly increasing success probability we have for a

brute-force search of a key space. The GNFS consists of several steps, and it only during the last step that we can expect results.

Our standard security level will be 128 bits, which corresponds to a 3248-bit modulus. A standard size for moduli in the same order of magnitude is 3072 bits, which corresponds to a security level of 124.7 bits. We feel that this is close enough and use 3072-bit moduli, which is also the NIST recommendation for 128-bit security.

DDH

We do not know any way to break DDH faster than computing the discrete logarithm, and the fastest algorithm for computing discrete logarithms in elliptic curves are generic algorithms, which need $\mathcal{O}(\sqrt{q})$ steps in a group of prime order q. If we choose a prime q with $|q| = 2\kappa$, then, using current knowledge, computing discrete logarithms in groups of order q on suitable elliptic curves — and therefore breaking DDH in these groups — will take approximately 2^κ steps. In finite fields of order q, algorithms to compute discrete logarithms exist that are as efficient as algorithms for factoring integers of size $|q|$, so the size of group elements for DDH-based schemes in finite fields and RSA-based schemes are comparable.

Pairing-Friendly Curves

Bilinear maps are usually instantiated on special elliptic curves. To define an elliptic curve, we need an affine Weierstraß equation over the algebraical closure $\bar{\mathbb{K}}$ of a field $\mathbb{K}$, which is is an equation of the form

$$E : Y^2 + a_1XY + a_3Y - (X^3 + a_2X^2 + a_4X + a_6) = 0$$

with $a_1, a_2, a_3, a_4, a_6 \in \mathbb{K}$.

Definition 2.18 (Elliptic curve)**.** Let $\mathbb{K}$ be a field with algebraic closure $\bar{\mathbb{K}}$. For an irreducible polynomial $C \in \mathbb{K}[X, Y]$, the *affine plane curve* over $\bar{\mathbb{K}}$, which we also denote by C, is the set of points $(x, y) \in \bar{\mathbb{K}}^2$ for which $C(x, y) = 0$. A point (x, y) is a *singular point* of C, if $\frac{\mathrm{d}C}{\mathrm{d}X}(x, y) = \frac{\mathrm{d}C}{\mathrm{d}Y}(x, y) = 0$. A curve without singular points is called nonsingular. An *elliptic curve* $E(\mathbb{K})$ over a field $\mathbb{K}$ is a nonsingular affine plane curve given in Weierstraß normal form together with a point at infinity O.

$$E(\mathbb{K}) \stackrel{\mathsf{def}}{=} \{(x, y) \in \mathbb{K}^2 \mid y^2 + a_1xy + a_3y - (x^3 + a_2x^2 + a_4x + a_6) = 0\} \cup \{O\}$$

While the points on an elliptic curve form an additive group, some elliptic curves $E(\mathbb{F}_{q^k})$ over a finite field $\mathbb{F}_{q^k}$ are *pairing-friendly*. This means there are groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order r with $|r| = 2\kappa$, and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, which must be non-degenerate (i. e. $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$ for generators g_1, g_2 of $\mathbb{G}_1, \mathbb{G}_2$). Usually, we have $\mathbb{G}_1 \subset E(\mathbb{F}_q), \mathbb{G}_2 \subset E(\mathbb{F}_{q^k}), \mathbb{G}_T \subset \mathbb{F}^*_{q^k}$, and k is called the *embedding degree*. [GPS06]

The size of a scalar in $\mathbb{Z}_r$ will be 2κ. The group size depends on the requirement that the DLP needs to be hard in $\mathbb{F}^*_{q^k}$ as well as in $E(\mathbb{F}_q)$, so $k \cdot |q|$ needs to be as big as usual DH security parameters for finite fields, and the size of a group element of $\mathbb{G}_T$ will be the same as an RSA modulus of the same security level.

The size of an element of $\mathbb{G}_1$ will be $\frac{1}{k}$ of the size of an element of $\mathbb{G}_T$, but at least 2κ. For asymmetric pairings, the size of an element of $\mathbb{G}_2$ is the same as the size of an element of $\mathbb{G}_T$, but can in practice usually be reduced by a factor of six by using twisted curves. [FST10]

In composite-order groups, it must be hard to factorize the curve order $\#E(\mathbb{F}_q)$, which should therefore have the size of an RSA modulus. It is at this point that constraints in the availability of constructions of pairing-friendly curves come into play. We introduce the parameter $\rho \stackrel{\mathsf{def}}{=} \frac{\log q}{\log r}$. For ordinary (non-supersingular) curves, only constructions with $\rho \approx 2$ and $k = 1$ are known. Therefore, the size of $\mathbb{G}_T$ will be twice the size of an RSA modulus of the same security level, and the size of $\mathbb{G}$ (for a symmetric pairing) or $\mathbb{G}_1$ and $\mathbb{G}_2$ (for an asymmetric pairing) will be the size of an RSA modulus of the same security level.

Freeman, Scott, and Teske [FST10] give an overview of known pairing-friendly curves. If we want to use a symmetric pairing on a prime-order curve, we can choose a Barreto-Naehrig curve with an embedding degree of 12 and $\rho = 1$. For a security parameter of 128 bits, this requires a 256-bit prime, resulting in elements of $\mathbb{G}$ having 256-bit representations, and elements in $\mathbb{G}_T$ having representations of length $12 \cdot 256 = 3072$ bits.

Parameters for q-type Assumptions

For q-type assumptions, there are generic attacks that need to be taken into account. As an example, we take the *q-strong Diffie-Hellman problem* (q-SDH) in bilinear-map groups $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ with an efficiently computable isomorphism $\varphi : \mathbb{G}_2 \to \mathbb{G}_1$. We define the success probability $\mathsf{Succ}^{q\text{-}\mathsf{SDH}}_{\mathcal{A},\mathbf{G}}(\kappa)$ of an algorithm $\mathcal{A}$ in solving the q-SDH problem in a group $\mathbf{G}$ to be

$$\Pr[\mathcal{A}(g_1, g_2, g_2^x, \ldots, g_2^{x^q}) = (c, g_1^{\frac{1}{x+c}}) | (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \varphi) \leftarrow \mathbf{G}(1^\kappa), g_2 \xleftarrow{\$} \mathbb{G}_2, g_1 \stackrel{\mathsf{def}}{=} \varphi(g_2), x \xleftarrow{\$} \mathbb{Z}_p^*]$$

Theorem 2.19 (Th. 3 in [BB04])**.** *In the generic bilinear group model, any adversary who makes at most q_G queries in total to the oracles computing the operations in the groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ or order p, the oracle computing the isomorphism φ, and the oracle computing the pairing e has success probability*

$$\mathsf{Succ}^{q\text{-}SDH}_{\mathcal{A},\mathbf{G}}(\kappa) \leq \frac{(q_G + q + 2)^2 q}{p} \in \mathcal{O}\left(\frac{q_G^2 q + q^3}{p}\right).$$

Theorem 2.20 (Cor. 1 in [BB04])**.** *Any adversary that solves the q-SDH problem with constant probability $\varepsilon > 0$ in generic groups of order p such that $q < o(\sqrt[3]{p})$ requires $\Omega(\sqrt{\varepsilon p / q})$ generic group operations.*

As a rule of thumb, a q-type problem with security parameter $\kappa + \frac{1}{2}\log q$ is as hard as the DDH problem for security parameter κ. This does not influence group sizes that are based on the number field sieve (RSA size).

2.4 Symmetric Primitives

All of cryptography relies on unproven (but plausible) assumptions. The minimal assumption we have to make is the existence of functions that are easy to compute, but hard to invert. We follow the definitions from [Gol04]

Definition 2.21 (One-way function)**.** A polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ is a *one-way function* (OWF) if $\forall \mathcal{PPTA} : \Pr[x \xleftarrow{\$} \{0,1\}^\kappa; y \leftarrow \mathcal{A}(1^\kappa, f(x)); f(y) = f(x)]$ is negligible in κ.

It is often easier to use OWFs when we can view them as families of functions. We therefore give an equivalent characterization.

Definition 2.22 (OWF family)**.** Let I be an infinite set of indices. A family of functions $\{f_i : D_i \to \{0,1\}^*\}_{i \in I}$ with finite domains D_i is *one-way* if there is a three-tuple of PPT algorithms $(\mathsf{Gen}, \mathsf{Samp}, \mathsf{Eval})$:
- $\mathsf{Gen}(1^\kappa)$ returns an index $i \in I \cap \{0,1\}^\kappa$;
- $\mathsf{Samp}(i)$ returns an element $x \in D_i$;
- $\mathsf{Eval}(i, x)$ outputs $f_i(x)$;

and $\forall \mathcal{PPTA} : \Pr[i \leftarrow \mathsf{Gen}(1^\kappa); x \leftarrow \mathsf{Samp}(i); y \leftarrow \mathcal{A}(i, f_i(x)); f_i(x) = f_i(y)]$ is negligible in κ.

The existence of a OWF implies that $\mathbf{NP} \not\subseteq \mathbf{BPP}$, and therefore $\mathbf{P} \neq \mathbf{NP}$. If there is a family of groups $\mathbf{G}$ in which DLOG is hard, then OWF families exist. Since the existence of OWF is equivalent to the existence of secure pseudo-random generators, message authentication codes, and symmetric encryption schemes, the existence of all the primitives described in the rest of this section is assured under this very weak computational assumption.

For one of our constructions, we will need a OWF with an additional property.

Definition 2.23 (Homomorphic OWF family)**.** Let $\{(G_i, +)\}_{i \in I}$ and $\{(H_i, *)\}_{i \in I}$ be two families of groups with $2^{|i|-1} \leq |G_i| \approx |H_i| \leq 2^{|i|}$. A family of OWF $\{f_i : G_i \to H_i\}_{i \in I}$ is *homomorphic* if $\forall \kappa \in \mathbb{N}, i \in [\mathsf{Gen}(1^\kappa)], x, y \in G_i : f_i(x + y) = f_i(x) * f_i(y)$.

An example for a homomorphic OWF is discrete exponentiation in finite groups in which DLOG is hard.

Definition 2.24 (Universal one-way hash function)**.** Let $l : \mathbb{N} \to \mathbb{N}$, U_n the uniform distribution over $[n]$. A collection of functions $\{h_s : \{0,1\}^* \to \{0,1\}^{l(|s|)}\}_{s \in \{0,1\}^*}$ is called *universal one-way hash function* (UOWHF) if there is a PPT algorithm I so that the following holds:

1. For some polynomial p, all sufficiently large κ, and every s in the range of $I(1^\kappa)$, $\kappa \leq p(|s|)$, and κ can be computed in polynomial time from s.
2. There is a PPT algorithm A such that $A(s, x) = h_s(x)$.

$\mathrm{Exp}^{\mathsf{prg}\text{-}b}_{\mathcal{F},\mathcal{A}}(\kappa)$
  $b' \leftarrow \mathcal{A}^{\mathsf{OF}()}(1^\kappa)$;
  return b';

$\mathsf{OF}()$
  $x \xleftarrow{\$} X$; $y \xleftarrow{\$} Y$;
  if $b = 0$ then return $\mathcal{F}(x)$;
  else return y;

Figure 2.4: $\mathcal{PRG}$: Pseudo-randomness

$\mathrm{Exp}^{\mathsf{ind}\text{-}\mathsf{cca}\text{-}b}_{\mathcal{SE},\mathcal{A}}(\kappa)$
  $\mathcal{Q}_D \leftarrow \emptyset$, $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\kappa)$;
  $(state, m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{ODecrypt}(\cdot),\mathsf{OEncrypt}(\cdot)}(\mathsf{FIND}; 1^\kappa)$;
  $c^* \leftarrow \mathsf{Encrypt}(\mathsf{ek}, m_b)$;
  $b' \leftarrow \mathcal{A}^{\mathsf{ODecrypt},\mathsf{OEncrypt}(\cdot)}(\mathsf{GUESS}, state; c^*)$;
  if $c^* \in \mathcal{Q}_D$ then return 0;
  else return b';

$\mathsf{ODecrypt}(c)$
  $\mathcal{Q}_D \leftarrow \mathcal{Q}_D \cup \{c\}$;
  $m \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$;
  return m;

$\mathsf{OEncrypt}(c)$
  $c \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$;
  return c;

Figure 2.5: $\mathcal{SE}$: Indistinguishability against adaptive chosen-ciphertext attacks (IND-CCA2)

3. For every polynomial q, every deterministic PT algorithm A_0, every PPT algorithm $\mathcal{A}$, and all sufficiently large κ,

$$\Pr[h_{I(1^\kappa)}(\mathcal{A}(I(1^\kappa), U_{q(\kappa)})) = h_{I(1^\kappa)}(A_0(U_{q(\kappa)})) \wedge \mathcal{A}(I(1^\kappa), U_{q(\kappa)}) \neq A_0(U_{q(\kappa)})]$$

   is negligible in κ, where the probability is taken over $U_{q(n)}$ and the randomness used by $\mathcal{A}$ and I.

The existence of OWF is equivalent to the existence of UOWHF [Gol04, th. 6.4.29].

Definition 2.25 (Pseudo-random generator)**.** A PPT-algorithm $\mathcal{F} : X \to Y$ is a (t, q_F, ε)-pseudo-random generator (PRG) if for all $\kappa \in \mathbb{N}$, in the security game presented in figure 2.4, the advantage $\mathrm{Adv}^{\mathsf{prg}}_{\mathcal{F}}(\kappa, t, q_F)$ of any t-time adversary $\mathcal{A}$ asking at most q_F sampling queries to the OF oracle is bounded by ε:

$$\mathrm{Adv}^{\mathsf{prg}}_{\mathcal{F}}(\kappa, t, q_F) = \max_{\mathcal{A}}\{\Pr[\mathrm{Exp}^{\mathsf{prg}\text{-}1}_{\mathcal{F},\mathcal{A}}(\kappa) = 1] - \Pr[\mathrm{Exp}^{\mathsf{prg}\text{-}0}_{\mathcal{F},\mathcal{A}}(\kappa) = 1]\}.$$

The existence of OWF is equivalent to the existence of pseudo-random generators [Gol01, prop. 3.3.8, th. 3.5.12].

In the following, Y may be the product of two sets $Y_1 \times Y_2$. We will then parse $\mathcal{F}(x) = (f_1(x), f_2(x))$. If $\mathcal{F}$ is a bijection (which implies that the PRG is not expanding), then $\mathcal{F}$ is a perfect generator, with $\varepsilon = 0$ and no computational assumption.

Definition 2.26 (Symmetric encryption scheme)**.** A symmetric encryption scheme is a 3-tuple of PPT algorithms $\mathcal{SE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with an associated key space $\mathcal{K} = \{\mathcal{K}_\kappa\}_{\kappa\in\mathbb{N}}$:
- $\mathsf{KeyGen}(1^\kappa)$, where κ is the security parameter, generates a secret key $\mathsf{sk} \in \mathcal{K}_\kappa$.
- $\mathsf{Enc}(\mathsf{sk}, m)$ takes as input the secret key sk and a message m and outputs a ciphertext c.
- $\mathsf{Dec}(\mathsf{sk}, c)$ takes as input the secret key sk and a ciphertext c and outputs a plaintext m or the symbol $\perp$.

that fulfills the following correctness requirement. For every $\kappa \in \mathbb{N}$, every $\mathsf{sk} \in [\mathsf{KeyGen}(1^\kappa)]$ and every $m \in \{0,1\}^*$,

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{sk}, m)) = m] = 1,$$

where the probability is taken over the randomness used by Enc and Dec.

Examples for symmetric encryption schemes used in practice are the stream cipher RC4 or AES-CBC.

Definition 2.27 (IND-CCA)**.** A symmetric encryption scheme $\mathcal{SE}$ is said to be $(t, q_D, q_E, \varepsilon)$-IND-CCA secure (indistinguishability against adaptive chosen-ciphertext attacks) if for all $\kappa \in \mathbb{N}$, in the security game presented in figure 2.5, the advantage $\mathrm{Adv}^{\mathsf{ind}\text{-}\mathsf{cca}}_{\mathcal{SE}}(\kappa, t, q_D, q_E)$ of any t-time adversary $\mathcal{A}$ asking at

$\mathrm{Exp}^{\mathsf{suf\text{-}cma}}_{\mathcal{MAC},\mathcal{A}}(\kappa)$
  $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\kappa)$;
  $\mathcal{Q}_S \leftarrow \emptyset$; $\mathcal{Q}_V \leftarrow \emptyset$;
  $\mathcal{A}^{\mathsf{OGenMac}(\cdot),\mathsf{OVerifMac}(\cdot,\cdot)}(1^\kappa)$;
  `if` $\exists(m,\sigma) \in \mathcal{Q}_V, (m,\sigma) \notin \mathcal{Q}_S$ `then return` 1;
  `else return` 0;

$\mathsf{OGenMac}(m)$
  $\sigma \leftarrow \mathsf{GenMac}(\mathsf{sk}, m)$;
  $\mathcal{Q}_S \leftarrow \mathcal{Q}_S \cup \{(m,\sigma)\}$
  `return` σ;

$\mathsf{OVerifMac}(m,\sigma)$
  $v = \mathsf{VerifMac}(\mathsf{sk}, m, \sigma)$;
  `if` $v = 1$ `then` $\mathcal{Q}_V \leftarrow \mathcal{Q}_V \cup \{(m,\sigma)\}$;
  `return` c;

Figure 2.6: $\mathcal{MAC}$: Strong unforgeability against chosen-message attacks (SUF-CMA)

most q_D decryption queries to the ODecrypt oracle and at most q_E queries to the OEncrypt oracle is bounded by ε:

$$\mathrm{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{SE}}(\kappa, t, q_D, q_E) = \max_{\mathcal{A}}\{\Pr[\mathrm{Exp}^{\mathsf{ind\text{-}cca\text{-}1}}_{\mathcal{SE},\mathcal{A}}(\kappa) = 1] - \Pr[\mathrm{Exp}^{\mathsf{ind\text{-}cca\text{-}0}}_{\mathcal{SE},\mathcal{A}}(\kappa) = 1]\}.$$

This definition includes IND-CPA (for Chosen-Plaintext Attacks) when $q_D = 0$ and IND-EAV (indistinguishability against eavesdroppers) when the adversary does not have access to any oracles in both phases.

The existence of non-uniformly hard (i. e. hard to invert for circuit families) OWF implies the existence of IND-CCA secure symmetric encryption schemes [Gol04, th. 5.4.21].

Definition 2.28 (Message authentication code)**.** A message authentication code is a 3-tuple of PPT algorithms $\mathcal{MAC} = (\mathsf{KeyGen}, \mathsf{GenMac}, \mathsf{VerifMac})$ with an associated key space $\mathcal{K} = \{\mathcal{K}_\kappa\}_{\kappa\in\mathbb{N}}$:
- $\mathsf{KeyGen}(1^\kappa)$, where κ is the security parameter, generates a secret key $\mathsf{sk} \in \mathcal{K}_\kappa$.
- $\mathsf{GenMac}(\mathsf{sk}, m)$ takes as input the secret key sk and a message m, and generates the MAC value σ.
- $\mathsf{VerifMac}(\mathsf{sk}, m, \sigma)$ takes as input the secret key sk, the message m and a signature σ. It checks the validity of the signature and returns a bit b (1 if it is valid, 0 else).

that fulfills the following correctness requirement. For every $\kappa \in \mathbb{N}, \mathsf{sk} \in [\mathsf{KeyGen}(1^\kappa)]$ and every $m \in \{0,1\}^*$,

$$\Pr[\mathsf{VerifMac}(\mathsf{sk}, m, \mathsf{GenMac}(\mathsf{sk}, m)) = 1] = 1,$$

where the probability is taken over the randomness used by GenMac and VerifMac.

In the following, we will require strong unforgeability against chosen-message attacks: even after seeing valid message-tag-pairs, the adversary cannot generate a new valid pair, even for the already authenticated message. This strong unforgeability is formalized in the security game presented in figure 2.6, where the adversary wins if it successfully verifies a message-tag-pair that has not been generated by the GenMac algorithm.

Definition 2.29 (SUF-CMA)**.** A message authentication code $\mathcal{MAC}$ is said to be $(t, q_M, q_V, \varepsilon)$-SUF-CMA secure (strong existential unforgeability against chosen-message attacks) if for all $\kappa \in \mathbb{N}$, in the security game presented in figure 2.6, the success probability $\mathrm{Succ}^{\mathsf{suf\text{-}cma}}_{\mathcal{MAC}}(\kappa, t, q_M, q_V)$ of any t-time adversary $\mathcal{A}$, asking at most q_M queries to the OGenMac oracle and q_V queries to the OVerifMac oracle is bounded by ε:

$$\mathrm{Succ}^{\mathsf{suf\text{-}cma}}_{\mathcal{MAC}}(\kappa, t, q_M, q_V) = \max_{\mathcal{A}}\{\Pr[\mathrm{Exp}^{\mathsf{suf\text{-}cma}}_{\mathcal{MAC},\mathcal{A}}(\kappa) = 1]\}.$$

This definition includes one-time security of the MAC when $q_M = 1$ and EUF-CMA security (existential unforgeability against chosen-message attacks) with the stronger restriction that $(m, *) \notin \mathcal{Q}_S$ (producing additional signatures for already signed messages is not a valid attack). In most real-world constructions, GenMac is deterministic, and in this case each message has a unique valid signature and SUF-CMA security and EUF-CMA security are the same notion. This means that the existence of OWF is equivalent to the existence of SUF-CMA secure message authentication codes [Gol04, th. 6.3.3].

2.5 Public-Key Primitives

We now turn to public-key primitives which will serve as building blocks for our constructions. We define trapdoor one-way permutations in section 2.5.1. In section 2.5.2 we present public-key encryption,

$\mathrm{Exp}^{\mathsf{ind\text{-}cca}\text{-}b}_{\mathcal{PKE},\mathcal{A}}(\kappa)$
  $\mathsf{param} \leftarrow \mathsf{Setup}(1^\kappa)$;
  $\mathcal{Q}_D \leftarrow \emptyset$, $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$;
  $(state, m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{ODecrypt}(\cdot)}(\mathsf{FIND}; \mathsf{param}, \mathsf{ek})$;
  $c^* \leftarrow \mathsf{Encrypt}(\mathsf{ek}, m_b)$;
  $b' \leftarrow \mathcal{A}^{\mathsf{ODecrypt}}(\mathsf{GUESS}, state; c^*)$;
  `if` $c^* \in \mathcal{Q}_D$ `then return` 0;
  `else return` b';

$\mathsf{ODecrypt}(c)$
  $\mathcal{Q}_D \leftarrow \mathcal{Q}_D \cup \{c\}$;
  $m \leftarrow \mathsf{Decrypt}(\mathsf{dk}, c)$;
  `return` m;

Figure 2.7: $\mathcal{PKE}$: Indistinguishability against Chosen-Ciphertext Attacks (IND-CCA)

with ElGamal and Cramer-Shoup encryption as examples. We then show how public-key and secret-key encryption are used together in practice as hybrid encryption in section 2.5.3. We move on to more advanced schemes, presenting identity-based encryption in section 2.5.4 and attribute-based encryption in section 2.5.5. Since we emphasize the security of our constructions, we present in section 2.5.6 generic transformations of IND-CPA secure schemes to IND-CCA secure ones. We also look at key exchange protocols in section 2.5.7, presenting the Diffie-Hellman protocol as an example, and at group key agreement in section 2.5.8, where we present the Burmester-Desmedt protocol.

2.5.1 Trapdoor Permutations

Trapdoor one-way permutations (TOWP) are one-way functions which are permutations and which are easy to invert given some auxiliary input (the "trapdoor").

Definition 2.30 (TOWP)**.** Let I be an infinite set of indices. A family of permutations $\{f_i : D_i \to D_i\}_{i \in I}$ with finite domains D_i is called a *trapdoor permutation* if there is a four-tuple of PPT algorithms $(\mathsf{Gen}, \mathsf{Samp}, \mathsf{Eval}, \mathsf{Inv})$:
- $\mathsf{Gen}(1^\kappa)$ returns an index $i \in I \cup \{0,1\}^\kappa$ and a trapdoor $\mathsf{td} \in \{0,1\}^*$;
- $\mathsf{Samp}(i)$ returns an element $x \in D_i$ that is uniformly distributed in D_i;
- $\mathsf{Eval}(i, x)$ outputs $y \in D_i$;
- $\mathsf{Inv}(\mathsf{td}, y)$ outputs $x \in D_i$;

We require that
- for all $(i, \mathsf{td}) \in [\mathsf{Gen}(1^\kappa)], x \in [\mathsf{Samp}(i)]$, both $\Pr[\mathsf{Eval}(i, x) = f_i(x)]$ and $\Pr[\mathsf{Inv}(\mathsf{td}, \mathsf{Eval}(i, x)) = x]$ are both overwhelming in κ (correctness)
- and $\forall \mathcal{PPTA} : \Pr[(i, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa); x \leftarrow \mathsf{Samp}(i); y \leftarrow \mathcal{A}(i, f_i(x)); f_i(x) = f_i(y)]$ is negligible in κ.

The standard instance of a TOWP is RSA, which we already mentioned in section 2.3.2, but now redefine using the above definition.

Definition 2.31 (RSA)**.** The RSA TOWP is given by the four algorithms
- $\mathsf{Gen}(1^\kappa)$ draws two primes $2^\kappa < p, q < 2^{\kappa+1}$ uniformly at random, sets $n = pq, \varphi(n) = (p-1)(q-1)$, finds e with $\gcd(e, \varphi(n)) = 1$, sets $d = e^{-1} \mod \varphi(n)$ and returns $i = (e, n)$ and $\mathsf{td} = (d, n)$;
- $\mathsf{Samp}(i)$ parses $i = (e, n)$ and returns a random element $x \in \mathbb{Z}_n^*$;
- $\mathsf{Eval}(i, x)$ parses $i = (e, n)$ and outputs $y = x^e \mod n$;
- $\mathsf{Inv}(\mathsf{td}, y)$ parses $\mathsf{td} = (d, n)$ outputs $x = y^d \mod n$;

The one-wayness of RSA is exactly the RSA problem from definition 2.12.

2.5.2 Public-Key Encryption

To make it easier to combine PKE with other schemes, we distinguish between the Setup algorithm, which generates public parameters, from the KeyGen algorithm, which uses these parameters to generate key pairs. KeyGen can be run several times on the same parameters to create key pairs that share the same parameters.

Definition 2.32 (Public-key encryption scheme)**.** A public-key encryption scheme is a 4-tuple of PPT algorithms $\mathcal{PKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$:
- $\mathsf{Setup}(1^\kappa)$, where κ is the security parameter, generates the global parameters param of the system;

- $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys, the public encryption key ek and the associated private decryption key dk;
- $\mathsf{Encrypt}(\mathsf{ek}, m; r)$ produces a ciphertext c on the input message m and the public key ek, using the random coins r (we may omit r when the notation is obvious);
- $\mathsf{Decrypt}(\mathsf{dk}, c)$ decrypts the ciphertext c under the private key dk. It outputs the plaintext, or $\perp$ if the ciphertext is invalid.

The correctness requirement is that for all $\mathsf{param} \in [\mathsf{Setup}(1^\kappa)], (\mathsf{ek}, \mathsf{dk}) \in [\mathsf{KeyGen}(\mathsf{param})]$, we have that $\Pr[\mathsf{Decrypt}(\mathsf{dk}, \mathsf{Encrypt}(\mathsf{ek}, m)) = m]$ is overwhelming in κ.

Definition 2.33 (IND-CCA)**.** A public-key encryption scheme $\mathcal{PKE}$ is said to be (t, q_D, ε)-IND-CCA secure (indistinguishability against chosen-ciphertext attacks) if for all $\kappa \in \mathbb{N}$, in the security game presented in figure 2.7, the advantage $\mathrm{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{PKE}}(\kappa, t, q_D)$ of any t-time adversary $\mathcal{A}$ asking at most q_D decryption queries to the $\mathsf{ODecrypt}$ oracle is bounded by ε:

$$\mathrm{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{PKE}}(\kappa, t, q_D) = \max_{\mathcal{A}}\{\Pr[\mathrm{Exp}^{\mathsf{ind\text{-}cca\text{-}1}}_{\mathcal{PKE},\mathcal{A}}(\kappa) = 1] - \Pr[\mathrm{Exp}^{\mathsf{ind\text{-}cca\text{-}0}}_{\mathcal{PKE},\mathcal{A}}(\kappa) = 1]\}.$$

This definition includes IND-CPA (for Chosen-Plaintext Attacks) when $q_D = 0$. The security of PKE in the multi-user setting was investigated by Bellare, Boldyreva, and Micali in [BBM00], who found that in the generic case, the security for N users encrypting a total of q_E messages degraded linearly in N and q_E.

$$\mathrm{Adv}^{\mathsf{ind\text{-}cca\text{-}}N}_{\mathcal{PKE}}(\kappa, t, q_D, q_E) \leq Nq_E \cdot \mathrm{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{PKE}}(\kappa, t', q_D),$$

where $\mathrm{Adv}^{\mathsf{ind\text{-}cca\text{-}}N}_{\mathcal{PKE}}(t, q_D, q_E)$ is the maximum advantage of any adversary running in time t, getting N public keys generated from $\mathsf{param} \in [\mathsf{Setup}(1^\kappa)]$ as input, and access to N decryption and left-or-right encryption oracles for these keys to which he asks at most q_D resp. q_E queries in total, and $t' = t + \mathcal{O}(\log(Nq_E))$. The LOR encryption oracles take as input two messages and all of them encrypt either the first or the second message.

The ElGamal encryption scheme [Elg85] was one of the first public-key encryption schemes.

Definition 2.34 (ElGamal)**.** Let $\mathbf{G}$ be a generator of cyclic groups.

- $\mathsf{Setup}(1^\kappa)$ calls $\mathbf{G}(1^\kappa)$ to obtain $(q, \mathbb{G})$, draws $g \xleftarrow{\$} \mathsf{Gens}(\mathbb{G})$ and outputs $\mathsf{param} = (g, q)$.
- $\mathsf{KeyGen}(g, q)$ draws a random element $a \xleftarrow{\$} \mathbb{Z}_q^*$, and outputs $(\mathsf{ek}, \mathsf{dk}) = (g^a, a)$.
- $\mathsf{Encrypt}(A, m)$ draws a random element $r \xleftarrow{\$} \mathbb{Z}_q^*$, and outputs $(R, C) = (g^r, A^r m)$.
- $\mathsf{Decrypt}(a, (R, C))$ outputs $m = C/R^a$.

Theorem 2.35. *The ElGamal encryption scheme is IND-CPA if $\mathsf{DDH}_{\mathbf{G}}$ is hard, and we have*

$$\mathrm{Adv}^{\mathsf{ind\text{-}cpa}}_{ElG(g,q)}(t) \leq 2 \cdot \mathrm{Adv}^{\mathsf{ddh}}_{(g,q)}(t).$$

Thank to the random self-reducibility of DDH, we can obtain a tight result in the multi-user case.

Theorem 2.36 (Th. 2 in [BBM00])**.** *Let $\mathbb{G}$ be a cyclic group of prime order q, g a generator of $\mathbb{G}$. Let ElG be the ElGamal scheme. Then for any $N, q_E, t \in \mathbb{N}$*

$$\mathrm{Adv}^{\mathsf{ind\text{-}cpa\text{-}}N}_{ElG(g,q)}(t, q_E) \leq 2 \cdot \mathrm{Adv}^{\mathsf{ddh}}_{(q,g)}(t') + \frac{1}{q}$$

where $t' = t + \mathcal{O}(Nq_E \cdot T_{exp}(q))$.

The Cramer-Shoup encryption scheme [CS98] uses a UOWHF (def. 2.24) to tie the ciphertext components together and ensure non-malleability.

Definition 2.37 (CS)**.** Let $\mathbf{G}$ be a generator of cyclic groups, $\mathcal{H} = \{h_s\}_{s\in\{0,1\}^*}$ a family of UOWHF.

- $\mathsf{Setup}(1^\kappa)$ calls $\mathbf{G}(1^\kappa)$ to obtain $(q, \mathbb{G})$, draws $g, h \xleftarrow{\$} \mathsf{Gens}(\mathbb{G})$ and an index $s \leftarrow I(1^\kappa)$ and outputs $\mathsf{param} = (g, h, q, s)$.
- $\mathsf{KeyGen}(g, h, q, s)$ draws random elements $v, w, x, y, z \xleftarrow{\$} \mathbb{Z}_q^*$, and outputs the key pair $(\mathsf{ek}, \mathsf{dk}) = ((g^x h^v, g^y h^w, g^z, s), (v, w, x, y, z, s))$.
- $\mathsf{Encrypt}\,((A, B, C, s), m)$ draws a random element $r \xleftarrow{\$} \mathbb{Z}_q^*$, and sets $U_1 = g^r, U_2 = h^r, E = C^r m, \alpha = h_s(U_1, U_2, E), V = A^r B^{r\alpha}$. It outputs (U_1, U_2, E, V).
- $\mathsf{Decrypt}\,((v, w, x, y, z, s), (U_1, U_2, E, V))$ computes $\alpha = h_s(U_1, U_2, E)$ and tests if $U_1^{x+y\alpha} U_2^{v+w\alpha} = V$, and outputs $\perp$ if not. Otherwise, it computes $m = E/U_1^z$ and outputs m.

Theorem 2.38 (Th. 1 in [CS98]). *The Cramer-Shoup encryption scheme is IND-CCA if $\mathsf{DDH}_{\mathbf{G}}$ is hard and $\mathcal{H}$ is a family of universal one-way hash functions, and as long as $q_D \leq q/2$, we have*

$$\mathrm{Adv}^{\mathsf{ind\text{-}cca}}_{CS(g,q)}(t, q_D) \leq 2 \cdot \mathrm{Adv}^{\mathsf{ddh}}_{(q,g)}(t) + 2 \cdot \mathrm{Adv}^{cr}_{\mathcal{H}}(t) + \frac{2(4q_D + 1)}{q}.$$

For the multi-user setting, we can see that the number of users N is much less important than the number of encryption queries.

Theorem 2.39 (Th. 3 in [BBM00]). *Let $\mathbb{G}$ be a cyclic group of prime order q, g a generator of $\mathbb{G}$, $\mathcal{H}$ a family of UOWHF. Let CS be the Cramer-Shoup scheme. Then for $N, q_D, q_E, t \in \mathbb{N}$ with $q_D \leq q/2$*

$$\mathrm{Adv}^{\mathsf{ind\text{-}cca\text{-}}N}_{CS(g,q)}(t, q_D, q_E) \leq 2q_E \cdot \mathrm{Adv}^{\mathsf{ddh}}_{(q,g)}(t') + 2q_E \cdot \mathrm{Adv}^{cr}_{\mathcal{H}}(t') + \frac{2Nq_E(4q_D + 1)}{q},$$

where $t' = t + \mathcal{O}(N \cdot T_{exp}(q))$.

2.5.3 Hybrid Encryption

When using PKE to encrypt a large amount of data, it is more efficient to encrypt the data using a symmetric encryption scheme and then encrypt the symmetric key using the PKE scheme. This is known as hybrid encryption, and was first formalized by Shoup [Sho00].

A *key encapsulation mechanism* (KEM) with key space $\mathcal{K} = \{\mathcal{K}_\kappa\}_{\kappa \in \mathbb{N}}$ is very similar to a PKE, except that the Encrypt and Decrypt algorithms are replaced with Encaps and Decaps.

- $\mathsf{Encaps}(\mathsf{ek}; r)$ produces a ciphertext H and a symmetric key $K \in \mathcal{K}_\kappa$;
- $\mathsf{Decaps}(\mathsf{dk}, H)$ decrypts the ciphertext H under the private key dk. It outputs a symmetric key K, or $\perp$ if the ciphertext is invalid.

The correctness requirement is similar to the one for PKE. The security notions for a KEM are similar to IND-CPA and IND-CCA for PKE, except that the adversary does not get to choose two messages to be encrypted, and instead gets the ciphertext and, in random order, the symmetric key and a key chosen uniformly at random from the key space $\mathcal{K}_\kappa$. He then has to determine which of the two keys is encapsulated in the ciphertext. We do not go into more details here, as we will define security for a broadcast KEM in definition 2.54.

When combining a KEM and a DEM (for data encapsulation mechanism), we need the key spaces to be compatible. Usually we want the symmetric key that is output by the KEM to be uniformly distributed in $\{0,1\}^\kappa$, because that is what the symmetric scheme expects as its key. If the KEM generates a random element of some group, it should transform its output to a uniform distribution on $\{0,1\}^\kappa$, e. g. by using a key derivation function [CS03]. Every PKE can be used as a KEM by choosing a random symmetric key and encrypting this key. The hybrid encryption or KEM/DEM paradigm ensures that using the key output by a KEM to encrypt a message using a DEM is secure.

Theorem 2.40 (Th. 10.13 in [KL08]). *If $\mathcal{KEM}$ is an IND-CPA secure KEM and $\mathcal{SE}$ an IND-EAV secure DEM, the resulting hybrid encryption scheme is an IND-CPA secure PKE.*

If both component schemes are IND-CCA secure, then the hybrid scheme is IND-CCA secure as well. Before looking at the security result, we need to give a definition. A key pair $(\mathsf{ek}, \mathsf{dk}) \in [\mathsf{KeyGen}(\mathsf{Setup}(1^\kappa))]$ is *bad* if for some $(K, H) \in [\mathsf{Encaps}(\mathsf{ek})]$, $\mathsf{Decaps}(\mathsf{sk}, H) \neq K$. Let $Bad_{\mathcal{KEM}}(\kappa)$ denote the probability that KeyGen generates a bad key pair.

Theorem 2.41 (Th. 5 in [CS03]). *If $\mathcal{KEM}$ is an IND-CCA secure KEM and $\mathcal{SE}$ an IND-CCA secure symmetric encryption scheme, then the resulting hybrid encryption scheme $\mathcal{PKE}$ is IND-CCA secure and*

$$\mathrm{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{PKE}}(\kappa) \leq Bad_{\mathcal{KEM}}(\kappa) + \mathrm{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{KEM},\mathcal{A}}(\kappa) + \mathrm{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{SE},\mathcal{A}}(\kappa).$$

We can relax the security requirement on the KEM a little if we use authenticated encryption as the symmetric primitive [HK07, th. 1].

2.5.4 Identity-based Encryption

The problem of identity-based encryption (IBE) was posed by Shamir [Sha85] in 1984 to solve key-management problems. Instead of having to distribute public keys via an authenticated channel, any bit string could be a public key, and a trusted authority is able to generate secret keys corresponding to any

bit string. Of course, the user secret keys have to be transported to the users over a secure channel. The first fully collusion resistant, pairing-based IBE scheme was constructed by Boneh and Franklin [BF01] in 2001.

Definition 2.42 (Identity-based encryption scheme)**.** An identity-based encryption scheme is a 4-tuple of PPT algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$:

- $\mathsf{Setup}(1^\kappa)$, where κ is the security parameter, generates the master public key MPK and the master secret key MSK the system;
- $\mathsf{KeyGen}(\mathsf{MSK}, \mathsf{id})$ generates the user decryption key $\mathsf{dk}_{\mathsf{id}}$ for identity id;
- $\mathsf{Encrypt}(\mathsf{MPK}, \mathsf{id}, m)$ produces a ciphertext c for identity id on the input message m and the public key MPK;
- $\mathsf{Decrypt}(\mathsf{dk}_{\mathsf{id}}, c)$ decrypts the ciphertext c under the private key $\mathsf{dk}_{\mathsf{id}}$. It outputs a plaintext, or $\perp$ if the ciphertext is invalid.

The correctness requirement is that for all $(\mathsf{MPK}, \mathsf{MSK}) \in [\mathsf{Setup}(1^\kappa)]$, $\mathsf{id} \in \{0,1\}^*$, $\mathsf{dk}_{\mathsf{id}} \in [\mathsf{KeyGen}(\mathsf{MSK}, \mathsf{id})]$, $\Pr[\mathsf{Decrypt}(\mathsf{dk}_{\mathsf{id}}, \mathsf{Encrypt}(\mathsf{MPK}, \mathsf{id}, m)) = m]$ is overwhelming in κ.

An extension of IBE is hierarchical IBE (HIBE), where users are organized as the nodes of a tree of depth L, and the identity of a user at level $0 \leq l \leq L$ is a vector $\mathsf{id} \in \{0,1\}^l$. The only difference to IBE is that the KeyGen algorithm can be employed by any user at a level $l < L$ to derive a key for a user at level $l + 1$.

Definition 2.43 (Hierarchical identity-based encryption scheme)**.** A hierarchical identity-based encryption scheme is a 4-tuple of PPT algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ where Setup, $\mathsf{Encrypt}$, and $\mathsf{Decrypt}$ are defined in the same way as for IBE and:

- $\mathsf{KeyGen}(\mathsf{dk}_{(\mathsf{id}_1,\ldots,\mathsf{id}_l)}, \mathsf{id}_{l+1})$ generates the user decryption key $\mathsf{dk}_{\mathsf{id}}$ for identity $\mathsf{id} = (\mathsf{id}_1, \ldots, \mathsf{id}_{l+1})$;

By viewing the MSK as the only key at level 0, IBE can be seen as a 1-level HIBE.

2.5.5 Attribute-based Encryption

Sahai and Waters [SW05] introduced attribute-based encryption (ABE) under the name of "fuzzy identity-based encryption". User decryption keys and ciphertexts were associated with attributes, and a key was able to decrypt a ciphertext if a certain number of their attributes overlapped. Goyal, Pandey, Sahai, and Waters [GPSW06] specified two types of ABE: ciphertext-policy attribute-based encryption (CP-ABE) and key policy ABE (KP-ABE). In CP-ABE, a user is described by several attributes, which are associated with his secret key. A ciphertext contains a policy, which describes which attributes a key must have to be able to decrypt it. Of course, users should not be able to combine their keys in order to enhance their decryption capabilities. KP-ABE is the pendant to CP-ABE where attributes are associated to ciphertexts and keys contain access policies that describe which ciphertexts they can decrypt.

Definition 2.44 (KP-ABE)**.** An attribute-based encryption scheme for a universe of attributes Γ is a 4-tuple of PPT algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$:

- $\mathsf{Setup}(1^\kappa)$, where κ is the security parameter, generates the master public key MPK and the master secret key MSK the system;
- $\mathsf{KeyGen}(\mathsf{MSK}, A)$ generates the user decryption key dk_A for an access structure A;
- $\mathsf{Encrypt}(\mathsf{MPK}, \gamma, m)$ produces a ciphertext c for a set of attributes $\gamma \subset \Gamma$ on the input message m and the public key MPK;
- $\mathsf{Decrypt}(\mathsf{dk}_A, c)$ decrypts the ciphertext c under the private key dk_A. It outputs a plaintext, or $\perp$ if the ciphertext is invalid.

The correctness requirement is that for all $(\mathsf{MPK}, \mathsf{MSK}) \in [\mathsf{Setup}(1^\kappa)]$, access structures $A \in 2^\Gamma$, $\mathsf{dk}_A \in [\mathsf{KeyGen}(\mathsf{MSK}, A)]$, $\gamma \subset \Gamma$ we have that if $\gamma \in A$, then $\Pr[\mathsf{Decrypt}(\mathsf{dk}_A, \mathsf{Encrypt}(\mathsf{MPK}, \gamma, m)) = m]$ is overwhelming in κ.

2.5.6 Generic Transformations

There are generic ways to construct $\mathsf{IND\text{-}CCA}$ secure PKE schemes. Those transforms were proved for PKE and it is not clear that they can be applied to BE schemes.

Naor-Yung The Naor-Yung transform (NY), first presented in [NY90], constructs an IND-CCA1 secure PKE scheme from an IND-CPA secure PKE scheme and a *non-interactive zero-knowledge proof* (NIZK). The plaintext is encrypted twice using different public keys and the ciphertext contains both encryptions and a NIZK that both ciphertexts contain the same message. The intuition behind this is that an adversary that manages to obtain two encryptions of the same message must already know the plaintext.

Dolev, Dwork, and Naor [DDN00] gave a related, but involved, construction that achieved IND-CCA2 security. Sahai [Sah99] showed that IND-CCA2 secure PKE can be constructed with the NY-technique when using a *one-time simulation-sound NIZK* (there called *non-malleable NIZK*. Lindell [Lin06] gave a simplified construction of one-time simulation-sound NIZK.

Canetti-Halevi-Katz The Canetti-Halevi-Katz transform (CHK) presented in [CHK04] constructs an IND-CCA secure PKE scheme from a selectively IND-CPA secure IBE (where the adversary needs to select the target identity before the setup phase) and a one-time SUF-CMA secure signature scheme. The encryption key of the PKE is the MPK of the IBE, the decryption key is the MSK, and to encrypt, the sender generates a key pair $(\mathsf{vk}, \sigma\mathsf{k})$ of the signature scheme, encrypts the message to the identity vk to obtain a ciphertext c and signs c using $\sigma\mathsf{k}$, obtaining a signature s. The receiver decrypts the ciphertext (vk, c, s) by first verifying the signature s using vk, rejecting if the verification fails. It then generates the decryption key $\mathsf{dk}_{\mathsf{vk}}$ and decrypts c.

ROM transformations Several generic transformations using random oracles exist. The optimal asymmetric encryption padding (OAEP) was presented in [BR94], where it was claimed that it would transform any TOWP into an IND-CCA secure PKE. In [Sho02], Shoup showed a gap in the proof and proposed an improved version, OAEP+. Fujisaki, Okamoto, Pointcheval, and Stern showed that OAEP still guarantees IND-CCA security under the partial-domain onewayness of the TOWP [FOPS04], which holds for RSA.

The Fujisaki-Okamoto transform [FO99] and REACT [OP01] use a random oracle to convert an IND-CPA secure PKE scheme into an IND-CCA secure PKE scheme.

2.5.7 Key Agreement

A key agreement protocol allows two users to generate a common secret over a public channel.

Definition 2.45 (Key agreement)**.** A key agreement protocol with key space $\mathcal{K} = \{\mathcal{K}_\kappa\}_{\kappa\in\mathbb{N}}$ for a set of user ids $\mathcal{UI} \subset \{0,1\}^*$ is a 3-tuple of PPT algorithms and protocols $\mathcal{KA} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{CommonKey})$:

- $\mathsf{Setup}(1^\kappa)$, where κ is the security parameter, generates the global parameters param of the system;
- $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys, the public key pk and the associated private key sk;
- $\mathsf{CommonKey}(\mathsf{sk}_i, j)$ is an interactive protocol that takes as input the secret key of user i and the identity $j \in \mathcal{UI}$ of the user that i wants to establish a key with and produces a key $K \in \mathcal{K}_\kappa$ that is shared by i and j.

Some two-round key agreement protocols like the Diffie-Hellman protocol consist only of sending the public key to the other party, and computing the common key as a function of this public key and the own secret key. In this case, we can assume that a long-term public-key is known to everyone and write CommonKey as an algorithm that takes the public key as input.

The correctness requirement is that for all $\kappa, s \in \mathbb{N}, \mathsf{param} \in [\mathsf{Setup}(1^\kappa)], S \subset \mathcal{UI}$ with $|S| = s, \{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i\in S} \in [\mathsf{KeyGen}(\mathsf{param})]^s$, we have $\mathsf{CommonKey}(\mathsf{sk}_i, j) = \mathsf{CommonKey}(\mathsf{sk}_j, i)$ for any $i, j \in S$.

We base our definition of security on the one given by Abdalla, Fouque, and Pointcheval [AFP05], with two modifications. We only consider security against eavesdroppers, which is a way to modularize protocol construction, as passively secure protocols can be made secure against active adversaries using generic conversions, such as [BCK98] for KA protocols or [KY07] for GKA, with additional authentication mechanisms. This means that in the security model, the adversary can only ask for transcripts τ of executions. We also take a simplifying view and identify parties with their public keys, excluding protocols that use both long-term and ephemeral keys. This means that any two parties can exchange keys only once, because we allow the session key to be determined by the involved parties' key pairs.

Definition 2.46. A key agreement protocol $\mathcal{KA}$ is said to be $(t, N, q_E, q_T, \varepsilon)$-IND secure (indistinguishability of session keys from random keys) if for all $\kappa \in \mathbb{N}$, in the security game presented in figure 2.8, the advantage $\mathrm{Adv}^{\mathsf{ind}}_{\mathcal{KA}}(\kappa, t, N, q_E, q_T)$ of any t-time adversary $\mathcal{A}$ creating at most N users (OKeyGen oracle),

1. We assume here that keys which do not exist are created using OKeyGen.

$\mathrm{Exp}^{\mathsf{ind}\text{-}b}_{\mathcal{KA},\mathcal{A}}(\kappa)$
$\mathcal{Q}_E \leftarrow \emptyset$; $\mathcal{Q}_T \leftarrow \emptyset$;
$\mathsf{param} \leftarrow \mathsf{Setup}(1^\kappa)$;
$b' \leftarrow \mathcal{A}^{\mathsf{OKeyGen}(\cdot),\mathsf{OExecute}(\cdot,\cdot)\mathsf{OTest}(\cdot,\cdot)}(\mathsf{param})$;
return b';

$\mathsf{OExecute}(i,j)$
if $\exists K : (i,j,K) \in \mathcal{Q}_E$ then return $\perp$;
$K \leftarrow \mathsf{CommonKey}(\mathsf{sk}_i, j)$; [1]
$\mathcal{Q}_E \leftarrow \mathcal{Q}_E \cup \{(i,j,K)\}$;
return τ;

$\mathsf{OTest}(i,j)$
if $\nexists K : (i,j,K) \in \mathcal{Q}_E$ then return $\perp$;
if $\exists K : (i,j,K) \in \mathcal{Q}_T$ then return K;
if $b = 0$ then find $K : (i,j,K) \in \mathcal{Q}_E$;
else $K \xleftarrow{\$} \mathcal{K}_\kappa$;
$\mathcal{Q}_T \leftarrow \mathcal{Q}_T \cup \{(i,j,K)\}$;
return K;

$\mathsf{OKeyGen}(i)$
$(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{param})$;
return pk_i;

Figure 2.8: $\mathcal{KA}$: Key Privacy (IND)

running the protocol at most q_E times (OExecute oracle) and asking for at most q_T session keys (OTest oracle), is bounded by ε:

$$\mathrm{Adv}^{\mathsf{ind}}_{\mathcal{KA}}(\kappa, t, N, q_E, q_T) = \max_{\mathcal{A}}\{\Pr[\mathrm{Exp}^{\mathsf{ind}\text{-}1}_{\mathcal{KA},\mathcal{A}}(\kappa) = 1] - \Pr[\mathrm{Exp}^{\mathsf{ind}\text{-}0}_{\mathcal{KA},\mathcal{A}}(\kappa) = 1]\}.$$

The first example for a key agreement protocol is the Diffie-Hellman key agreement. The Diffie-Hellman key agreement protocol [DH76] was the first public-key key agreement protocol.

Definition 2.47 (DHKA)**.** Let $\mathbf{G}$ be a generator of cyclic groups.

- $\mathsf{Setup}(1^\kappa)$ runs $\mathbf{G}(1^\kappa)$ to obtain $(q, \mathbb{G})$, draws $g \xleftarrow{\$} \mathsf{Gens}(\mathbb{G})$ and outputs $\mathsf{param} = (g, q)$.
- $\mathsf{KeyGen}(g, q)$ draws a random element $a \xleftarrow{\$} \mathbb{Z}_q^*$, and outputs $(\mathsf{pk}, \mathsf{sk}) = (g^a, a)$.
- $\mathsf{CommonKey}(a, B)$ outputs $K = B^a$. Here we identify a used with its public key.

Theorem 2.48. *The Diffie-Hellman key agreement protocol is IND secure if $\mathsf{DDH}_{\mathbf{G}}$ is hard.*

2.5.8 Group Key Agreement

Key agreement protocols allow two users to agree on a common secret. Group key agreement (GKA) protocols are a generalization of key agreement protocols to any number of users that wish to share a session key for secure group communication.

Definition 2.49 (Group key agreement)**.** A group key agreement protocol with key space $\mathcal{K} = \{\mathcal{K}_\kappa\}_{\kappa\in\mathbb{N}}$ for a set of user identifiers $\mathcal{UI} \subset \{0,1\}^*$ is a 3-tuple of PPT algorithms and interactive protocols $\mathcal{GKA} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{CommonKey})$, where Setup and KeyGen are defined as for a two-user key agreement protocol, and

- $\mathsf{CommonKey}(\mathsf{sk}_i, \mathcal{S})$ is an interactive protocol that takes as input the secret key of user i and the set of users $\mathcal{S}$ that i wants to establish a key with and produces a key $K \in \mathcal{K}_\kappa$ that is shared by all users in $\mathcal{S}$ (we assume that $i \in \mathcal{S}$).

For correctness we require that for all $\kappa, s \in \mathbb{N}, \mathsf{param} \in [\mathsf{Setup}(1^\kappa)], S \subset \mathcal{UI}$ with $|S| = s, U \subseteq S$, $\{(\mathsf{pk}_i, \mathsf{sk}_i)\}_{i\in S} \in [\mathsf{KeyGen}(\mathsf{param})]^s$, we have $\mathsf{CommonKey}(\mathsf{sk}_i, U) = \mathsf{CommonKey}(\mathsf{sk}_j, U)$ for any $i, j \in U$.

The security definition for GKA protocols is based on the same considerations as the security definition for KA protocols.

Definition 2.50. A group key agreement protocol $\mathcal{GKA}$ is said to be $(t, N, q_E, q_T, \varepsilon)$-IND secure (indistinguishability of session keys from random keys) if for all $\kappa \in \mathbb{N}$ in the security game presented in figure 2.9, the advantage $\mathrm{Adv}^{\mathsf{ind}}_{\mathcal{GKA}}(\kappa, t, N, q_E, q_T)$ of any t-time adversary $\mathcal{A}$ creating at most N users (OKeyGen oracle), running the protocol at most q_E times (OExecute oracle) and asking for at most q_T session keys (OTest oracle), is bounded by ε:

$$\mathrm{Adv}^{\mathsf{ind}}_{\mathcal{GKA}}(\kappa, t, N, q_E, q_T) = \max_{\mathcal{A}}\{\Pr[\mathrm{Exp}^{\mathsf{ind}\text{-}1}_{\mathcal{GKA},\mathcal{A}}(\kappa) = 1] - \Pr[\mathrm{Exp}^{\mathsf{ind}\text{-}0}_{\mathcal{GKA},\mathcal{A}}(\kappa) = 1]\}.$$

2. We assume here that keys which do not exist are created using OKeyGen.

$\mathrm{Exp}^{\mathsf{ind}\text{-}b}_{\mathcal{KA},\mathcal{A}}(\kappa)$
$\mathcal{Q}_E \leftarrow \emptyset$; $\mathcal{Q}_T \leftarrow \emptyset$;
$\mathsf{param} \leftarrow \mathsf{Setup}(1^\kappa)$;
$b' \leftarrow \mathcal{A}^{\mathsf{OKeyGen}(\cdot),\mathsf{OExecute}(\cdot,\cdot)\mathsf{OTest}(\cdot,\cdot)}(\mathsf{param})$;
return b';

$\mathsf{OExecute}(S)$
if $\exists K : (S,K) \in \mathcal{Q}_E$ then return $\perp$;
$i \xleftarrow{\$} S; K \leftarrow \mathsf{CommonKey}(\mathsf{sk}_i, S)$; [2]
$\mathcal{Q}_E \leftarrow \mathcal{Q}_E \cup \{(S,K)\}$;
return τ;

$\mathsf{OTest}(S)$
if $\nexists K : (S,K) \in \mathcal{Q}_E$ then return $\perp$;
if $\exists K : (S,K) \in \mathcal{Q}_T$ then return K;
if $b = 0$ then find $K : (S,K) \in \mathcal{Q}_E$;
else $K \xleftarrow{\$} \mathcal{K}_\kappa$;
$\mathcal{Q}_T \leftarrow \mathcal{Q}_T \cup \{(S,K)\}$;
return K;

$\mathsf{OKeyGen}(i)$
$(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{param})$;
return pk_i;

Figure 2.9: $\mathcal{GKA}$: Key Privacy (IND). The adversary receives a transcript τ of the CommonKey protocol run as input.

In 1994, Burmester and Desmedt [BD95] presented a two-round ring-based GKA with communication complexity linear in the number of users, for which they gave a security proof in [BD05]. They later constructed a three-round tree-based GKA with communication complexity logarithmic in the number of users [BD97]. Proofs for the security claims of the second construction are given in [DLB07] along with a transformation into an authenticated GKA.

Definition 2.51 (BD94)**.** The Burmester-Desmedt '94 protocol arranges the N users $[N]$ in a circle, so that user N and user 1 are neighbours. Let $\mathbf{G}$ be a generator of cyclic groups.

- $\mathsf{Setup}(1^\kappa)$ runs $\mathbf{G}(1^\kappa)$ to obtain $(q, \mathbb{G})$, draws $g \xleftarrow{\$} \mathsf{Gens}(\mathbb{G})$ and outputs $\mathsf{param} = (g, q)$.
- $\mathsf{KeyGen}(g, q)$ draws a random element $a \xleftarrow{\$} \mathbb{Z}_q^*$, and outputs $(\mathsf{pk}, \mathsf{sk}) = (g^a, a)$.
- $\mathsf{CommonKey}(a_i, \mathcal{S})$ The protocol consists of two rounds:
  1. Each user broadcasts his public key $A_i \stackrel{\mathsf{def}}{=} g^{a_i}$.
  2. Each user computes $V_i \stackrel{\mathsf{def}}{=} (A_{i+1}/A_{i-1})^{a_i} \mod q$ and broadcasts V_i.

  Each user outputs $K_i \stackrel{\mathsf{def}}{=} (A_{i-1}^{a_i})^N \cdot V_i^{N-1} \cdot V_{i+1}^{N-2} \cdot \ldots \cdot V_{i-1} \mod q$.

We have $A_{i-1}^{a_i} = g^{a_{i-1}a_i}$ and $V_i = g^{-a_{i-1}a_i + a_i a_{i+1}}$, so for all users i, $K_i = g^{a_1a_2 + a_2a_3 + \cdots + a_N a_1}$.

Theorem 2.52 (Th. 3.4 in [BD05])**.** *The BD94 GKA protocol is IND secure if $\mathsf{DDH}_{\mathbf{G}}$ is hard.*

The concrete security in a different security model is given in [KY07, th. 2], where the advantage of the adversary is bounded by four times the advantage against DDH plus a term depending on the number of protocol executions the adversary sees.

2.6 Broadcast Encryption

Broadcast encryption (BE) schemes are a kind of one-to-many encryption schemes which suppose the existence of a broadcast channel to transmit a ciphertext to all registered users. Broadcast encryption enables the sender of a message to specify a subset of the registered users (the *target set* or *privileged set*), who will be able to decrypt the ciphertext. The complement of the target set (in the set of the registered users) is called the *revoked set*.

Throughout this thesis, we model broadcast encryption as a key encapsulation mechanism. A broadcast encryption scheme can be constructed from a broadcast encapsulation scheme and a symmetric cipher using the hybrid encryption paradigm (sec 2.5.3). For this reason, we only deal with the encapsulation part (which we still call broadcast encryption).

To accomplish user revocation when sending a message, a BE generally generates three parts: the *id header*, a bit string that unambiguously identifies the target set/revoked set; the *key header*, which encapsulates a session key for the privileged users; and the *message body*, that contains the payload encrypted under the session key.

Since the id header and the message body do not depend on the broadcast encryption scheme that is used, we will focus on the key header part only. Furthermore, when no more information is given, we will consider a public-key key encapsulation system with possibly stateful decoders: encryption key is

$\mathrm{Exp}^{\mathsf{corr}}_{\mathcal{DBE}}(\kappa, U, S, \mathsf{id}^*)$
$(\mathsf{MSK}, \mathsf{EK}, Reg_0) \leftarrow \mathsf{Setup}(1^\kappa)$;
`for all` $\mathsf{id} \in U$: $(\mathsf{usk}_{\mathsf{id}}, \mathsf{upk}_{\mathsf{id}}, Reg_{\tau+1}) \leftarrow \mathsf{Join}(\mathsf{MSK}, Reg_\tau, \mathsf{id})$;
$(H, K) \leftarrow \mathsf{Encaps}(\mathsf{EK}, Reg_{|U|}, \mathcal{S})$.
`if` $\mathsf{Decaps}(\mathsf{usk}_{\mathsf{id}}, \mathcal{S}, H) = K$ `then return` 1;
`else return` 0

Figure 2.10: $\mathcal{DBE}$: Correctness

$\mathrm{Exp}^{\mathsf{ind-d}xay\mathsf{cca}z\text{-}b}_{\mathcal{DBE},\mathcal{A}}(\kappa)$
$\mathcal{Q}_C \leftarrow \emptyset$; $\mathcal{Q}_D \leftarrow \emptyset$;
$(\mathsf{MSK}, \mathsf{EK}, Reg_0) \leftarrow \mathsf{Setup}(1^\kappa)$;
$(st, \mathcal{S}, \tau) \leftarrow \mathcal{A}^{\mathsf{OJoin}(\cdot),\mathsf{OCorrupt}(\cdot),\mathsf{ODecaps}(\cdot,\cdot,\cdot)}(\mathsf{EK})$;
$(H, K) \leftarrow \mathsf{Encaps}(\mathsf{EK}, Reg_\tau, \mathcal{S})$; $K_b \leftarrow K$; $K_{1-b} \xleftarrow{\$} \mathcal{K}$;
$b' \leftarrow \mathcal{A}^{\mathsf{OJoin}(\cdot),\mathsf{OCorrupt}(\cdot),\mathsf{ODecaps}(\cdot,\cdot,\cdot)}(st; \mathcal{S}, H, K_0, K_1)$;
`if` $\exists i \in \mathcal{S}, (i, \mathcal{S}, H) \in \mathcal{Q}_D$ `or` $i \in \mathcal{Q}_C$
`then return` 0;
`else return` b';

$\mathsf{OJoin}(i)$
$(\mathsf{usk}_i, \mathsf{upk}_i) \leftarrow \mathsf{Join}(\mathsf{MSK}, i)$;
$Reg_{t+1} = Reg_t || (i, \mathsf{upk}_i)$;
`return` upk_i;

$\mathsf{OCorrupt}(i)$
$\mathcal{Q}_C \leftarrow \mathcal{Q}_C \cup \{i\}$;
`return` usk_i;

$\mathsf{ODecaps}(i, \mathcal{S}, H)$
$\mathcal{Q}_D \leftarrow \mathcal{Q}_D \cup \{(i, \mathcal{S}, H)\}$
$K \leftarrow \mathsf{Decaps}(\mathsf{usk}_i, \mathcal{S}, H)$;
`return` K;

Figure 2.11: $\mathcal{DBE}$: Key Privacy (IND-ACCA)

public, the decryption keys of the users can evolve, but the updates will be global and sent on a public channel, and ephemeral keys are distributed to be used together with symmetric encryption. We will nevertheless sometimes make remarks about alternative cases.

Definition 2.53 (Dynamic broadcast encapsulation)**.** A dynamic broadcast encapsulation scheme (DBE) with key space $\mathcal{K} = \{\mathcal{K}_\kappa\}_{\kappa\in\mathbb{N}}$ for a set of user identifiers $\mathcal{UI} \subset \{0,1\}^*$ is a 4-tuple of PPT algorithms $\mathcal{DBE} = (\mathsf{Setup}, \mathsf{Join}, \mathsf{Encaps}, \mathsf{Decaps})$:

- $\mathsf{Setup}(1^\kappa)$, where κ is the security parameter, generates and returns a master secret key MSK and an encryption key EK. It also initiates an empty list Reg_0, the user register, which can be seen as the variable part of the encryption key. If the scheme is asymmetric, EK is public, otherwise it can be seen as a part of the MSK.
- $\mathsf{Join}(\mathsf{MSK}, Reg_\tau, \mathsf{id})$ takes as input the master secret key, the list Reg_τ, and a user identifier id. If $\mathsf{id} \in \mathcal{UI}$ and $\mathsf{id} \notin Reg_\tau$, outputs a user secret key $\mathsf{usk}_{\mathsf{id}}$ and a public user tag $\mathsf{upk}_{\mathsf{id}}$. The pair $(\mathsf{id}, \mathsf{upk}_{\mathsf{id}})$ is appended to Reg_τ to obtain $Reg_{\tau+1}$. Else, outputs $\perp$.
- $\mathsf{Encaps}(\mathsf{EK}, Reg_\tau, \mathcal{S})$ takes as input the encryption key, the list Reg_τ, and a target set $\mathcal{S}$ and outputs a key header H and a session key $K \in \mathcal{K}_\kappa$.
- $\mathsf{Decaps}(\mathsf{usk}_{\mathsf{id}}, \mathcal{S}, H)$ takes as input a user secret key, the target set $\mathcal{S}$, and the key header H. If $\mathsf{id} \in \mathcal{S}$, outputs the session key K.

The correctness requirement is that for any (polynomial size) set of joined users $U \subset \mathcal{UI}$, any target set $\mathcal{S} \subset U$ and for any $\mathsf{id}^* \in \mathcal{S}$, $\Pr[\mathrm{Exp}^{\mathsf{corr}}_{\mathcal{DBE}}(\kappa, U, \mathcal{S}, \mathsf{id}^*) = 1]$ is overwhelming in κ, where the experiment is defined in figure 2.10.

We adopt the security notion to be satisfied by such a dynamic broadcast encapsulation scheme from [PPS11]. This definition extends all the previous ones by giving the adversary unlimited access to the Join oracle (dynamic), the $\mathsf{Corrupt}$ oracle (adaptive) and Decaps oracle (chosen-ciphertext security). The security game is presented in figure 2.11: the restriction for the adversary is not to ask for the decapsulation of the challenge ciphertext nor corrupt any user in the target set at the time of the challenge. We let the adversary choose any earlier time period τ. The time period changes at each Join-query, and the current state (e. g. Reg) is indexed by this time period.

Definition 2.54 (IND-ACCA)**.** A dynamic broadcast encapsulation scheme $\mathcal{DBE}$ is $(t, N, q_C, q_D, \varepsilon)$-IND-ACCA secure (indistinguishability under adaptive corruption and chosen-ciphertext attacks) if for all

$\kappa \in \mathbb{N}$, in the security game presented in figure 2.11, the advantage $\mathrm{Adv}^{\mathsf{ind\text{-}acca}}_{\mathcal{DBE}}(\kappa, t, N, q_C, q_D)$ of any t-time adversary $\mathcal{A}$ creating at most N users (OJoin oracle), corrupting at most q_C of them ($\mathsf{OCorrupt}$ oracle), and asking for at most q_D decapsulation queries ($\mathsf{ODecaps}$ oracle), is bounded by ε:

$$\mathrm{Adv}^{\mathsf{ind\text{-}acca}}_{\mathcal{DBE}}(\kappa, t, N, q_C, q_D) = \max_{\mathcal{A}}\{\Pr[\mathrm{Exp}^{\mathsf{ind\text{-}acca\text{-}1}}_{\mathcal{DBE},\mathcal{A}}(\kappa) = 1] - \Pr[\mathrm{Exp}^{\mathsf{ind\text{-}acca\text{-}0}}_{\mathcal{DBE},\mathcal{A}}(\kappa) = 1]\}.$$

This definition includes $\mathsf{IND\text{-}ACPA}$ (for Chosen-Plaintext Attacks) when $q_D = 0$.

Remark 2.55 (Forward-secrecy)**.** *This definition includes forward-secrecy. For a definition without forward secrecy, the adversary is prohibited from corrupting users that joined after the challenge phase.*

Dynamic broadcast encryption Delerablée, Paillier, and Pointcheval formulated three conditions that a broadcast encryption scheme must fulfill to be called dynamic [DPP07].

1. the system setup as well as the ciphertext size are fully independent from the expected number of users,
2. a new user can join anytime without implying a modification of preexisting user decryption keys,
3. the encryption key is incrementally updated, and this operation must be of complexity at most $\mathcal{O}(1)$.

2.6.1 Terminology

Join algorithms When the Join algorithm can be run at the setup phase only, with no later evolution of the group, we say the scheme is *static* (instead of *dynamic*). In this case, the Join algorithm can be omitted, and the Setup algorithm takes the number of users N as an additional input and outputs $\mathsf{EK}, \mathsf{usk}_1, \ldots, \mathsf{usk}_N$. For a dynamic scheme, several kinds of Join functionalities are possible:

Passive, no input (except a counter i); it generates a public tag upk_i to identify the user;

Active, the input is id; it generates a public tag $\mathsf{upk}_{\mathsf{id}}$ to identify the user;

Identity-based, the input is id, and the public tag $\mathsf{upk}_{\mathsf{id}}$ is simply id. In addition, usually the space of user identifiers is of superpolynomial size.

We stress that the default case in this thesis (when no other version is specified) is that Join is passive.

Target set A broadcast encryption scheme specifies the target set by the list of authorized users, a *revocation scheme* specifies the target set by its complement $\mathcal{R}$, the set of revoked users. Static BE schemes and static revocation schemes are equivalent, they are usually named for the case where they are more efficient: Many authorized users or many revoked users. Dynamic BE schemes and dynamic revocation schemes are not equivalent: A user that joins after a broadcast is not able to decrypt the message in the BE case, because he is not in the authorized set. In a revocation scheme, he can decrypt, because he is not in the revoked set. Another difference arises in the identity-based case. Because the set of identifiers is usually superpolynomial and an identity-based revocation (IBR) scheme can only revoke polynomially many of them, it will address superpolynomially many users.

Key encapsulation mechanisms We described above a key encapsulation mechanism (KEM) where only a key is generated. The payload is then encrypted with a symmetric mechanism to get a full encryption scheme. All the broadcast encryption schemes known to the authors can be written as KEMs, e. g. the bilinear BE schemes from [BGW05, GW09] generate a random group element which is then multiplied to the message. This random group element can be considered as the symmetric key, and group multiplication as the symmetric encryption. To achieve $\mathsf{CCA2}$ security for the full broadcast encryption, given a $\mathsf{CCA2}$ secure key encapsulation, we additionally need to bind all the components of the ciphertext together.

Encryption and decryption keys The encryption key can be either public (asymmetric) or private (symmetric), in the former case, we talk about *public-key* broadcast encryption, in the latter we say this is a *private-key* broadcast encryption. The decryption keys can either be defined and sent to the users at the join phase and never modified again, or be updated each time another user joins the system. In the former case, the decoders are said to be *stateless* since there is no state to evolve. In the latter case, the decoders are called *stateful* because they have to keep their state up-to-date. They thus have to be always on-line to receive the update information.

$\mathsf{ODecrypt}(i, H)$
$\mathcal{Q}_D \leftarrow \mathcal{Q}_D \cup \{(i, H)\}$;
$K \leftarrow \mathsf{Decaps}(\mathsf{usk}_i, H)$; return K;

$\mathsf{OCorrupt}(i)$
$\mathcal{Q}_C \leftarrow \mathcal{Q}_C \cup \{i\}$;
return usk_i;

$\mathrm{Exp}^{\mathsf{ind\text{-}acca}\text{-}b}_{\Pi,\mathcal{A}}(\kappa, N)$
$(\{\mathsf{usk}_i\}, \mathsf{EK}) \leftarrow \mathsf{Setup}(1^\kappa, N)$; $\mathcal{Q}_C \leftarrow \emptyset$; $\mathcal{Q}_D \leftarrow \emptyset$;
$(state, S) \leftarrow \mathcal{A}^{\mathsf{ODecrypt}(\cdot,\cdot),\mathsf{OCorrupt}(\cdot)}(\mathsf{FIND}; \mathsf{EK})$;
$(K_1, H^*) \leftarrow \mathsf{Encaps}(\mathsf{EK}, S)$; $K_0 \xleftarrow{\$} \mathcal{K}$
$b' \leftarrow \mathcal{A}^{\mathsf{ODecrypt}(\cdot,\cdot),\mathsf{OCorrupt}(\cdot)}(\mathsf{GUESS}; state, K_b, H^*)$;
if $\exists i \in S : (i, H^*) \in \mathcal{Q}_D$ or $S \cap \mathcal{Q}_C \neq \emptyset$ then return 0;
else return b';

$\mathrm{Exp}^{\mathsf{ano\text{-}acca}\text{-}b}_{\Pi,\mathcal{A}}(\kappa, N)$
$(\{\mathsf{usk}_i\}, \mathsf{EK}) \leftarrow \mathsf{Setup}(1^\kappa, N)$; $\mathcal{Q}_C \leftarrow \emptyset$; $\mathcal{Q}_D \leftarrow \emptyset$;
$(state, S_0, S_1) \leftarrow \mathcal{A}^{\mathsf{ODecrypt}(\cdot,\cdot),\mathsf{OCorrupt}(\cdot)}(\mathsf{FIND}; \mathsf{EK})$;
$(K, H^*) \leftarrow \mathsf{Encaps}(\mathsf{EK}, S_b)$;
$b' \leftarrow \mathcal{A}^{\mathsf{ODecrypt}(\cdot,\cdot),\mathsf{OCorrupt}(\cdot)}(\mathsf{GUESS}, state; K, H^*)$;
if $\exists i \in \mathcal{S}_0 \triangle S_1 : (i, H^*) \in \mathcal{Q}_D$ or $(\mathcal{S}_0 \triangle S_1) \cap \mathcal{Q}_C \neq \emptyset$
then return 0; else return b';

Figure 2.12: Security games for ANOBE

Default As already mentioned, in this thesis, we focus on public-key key encapsulation system with possibly stateful decoders.

2.6.2 Anonymous Broadcast Encryption

Anonymous broadcast encryption (ANOBE) allows to address a message to a subset of the users, without revealing this target set even to users who successfully decrypted the message. We define an ANOBE as a key encapsulation mechanism (KEM), following the definitions found in [LPQ12].

Definition 2.56 (Anonymous broadcast encryption)**.** An *anonymous broadcast encapsulation* scheme for a key space $\mathcal{K} = \{\mathcal{K}_\kappa\}_{\kappa\in\mathbb{N}}$ is a 3-tuple of PPT algorithms $(\mathsf{Setup}, \mathsf{Encaps}, \mathsf{Decaps})$:

- $\mathsf{Setup}(1^\kappa, N)$, where κ is the security parameter, and N the number of users, generates N user secret keys $\{\mathsf{usk}_i\}_{i\in[N]}$, and an encryption key EK.
- $\mathsf{Encaps}(\mathsf{EK}, S; r)$ takes as input the encryption key EK, the target set $S \subset \{1, \ldots, N\}$, and some random coins r (which are sometimes omitted). It outputs a session key $K \in \mathcal{K}_\kappa$, and an encapsulation H of K;
- $\mathsf{Decaps}(\mathsf{usk}_i, c)$ takes as input a decryption key and a ciphertext c. It outputs the session key K, or the error symbol $\perp$.

For correctness, we require that for all $\kappa, N \in \mathbb{N}, (\mathsf{EK}, \mathsf{usk}_1, \ldots, \mathsf{usk}_N) \in [\mathsf{Setup}(1^\kappa, N)], S \subseteq [N], i \in S$ if $(H, K) \leftarrow \mathsf{Encaps}(\mathsf{EK}, S)$, then $K \leftarrow \mathsf{Decaps}(\mathsf{usk}_i, H)$ with overwhelming probability.

We choose to define confidentiality and anonymity in two separate experiments.

Definition 2.57 (Confidentiality)**.** An anonymous broadcast encryption scheme Π is $(t, N, q_C, q_D, \varepsilon)$-IND-ACCA secure (indistinguishability under adaptive corruption and chosen-ciphertext attacks) if for all $\kappa \in \mathbb{N}$ in the security game presented in figure 2.12, the advantage $\mathrm{Adv}^{\mathsf{ind\text{-}acca}}_\Pi(\kappa, t, N, q_C, q_D)$ of any t-time adversary $\mathcal{A}$ corrupting at most q_C users ($\mathsf{OCorrupt}$ oracle), and asking at most q_D decryption queries ($\mathsf{ODecrypt}$ oracle), is bounded by ε:

$$\mathrm{Adv}^{\mathsf{ind\text{-}acca}}_\Pi(\kappa, t, N, q_C, q_D) = \max_{\mathcal{A}}\{\Pr[\mathrm{Exp}^{\mathsf{ind\text{-}acca}\text{-}1}_{\Pi,\mathcal{A}}(\kappa, N) = 1] - \Pr[\mathrm{Exp}^{\mathsf{ind\text{-}acca}\text{-}0}_{\Pi,\mathcal{A}}(\kappa, N) = 1]\}.$$

This definition includes IND-ACPA (for Chosen-Plaintext Attacks) when $q_D = 0$, and thus we denote the advantage $\mathrm{Adv}^{\mathsf{ind\text{-}acpa}}_\Pi(\kappa, t, N, q_C)$. When no corruption is allowed, we denote the advantage $\mathrm{Adv}^{\mathsf{ind\text{-}cpa}}_\Pi(\kappa, t, N)$.

Definition 2.58 (Anonymity)**.** Let $\triangle$ denote the symmetric difference of two sets: $S \triangle T \stackrel{\mathsf{def}}{=} S \cup T \setminus S \cap T$. We say that an anonymous broadcast encryption (ANOBE) scheme Π is $(\tau, N, q_C, q_D, \varepsilon)$-ANO-ACCA secure (anonymity against adaptive corruption and chosen-ciphertext attacks) if for all $\kappa \in \mathbb{N}$, in the security game presented in figure 2.12, the advantage $\mathrm{Adv}_{\Pi}^{\mathsf{ano\text{-}acca}}(\kappa, t, N, q_C, q_D)$, of any t-time adversary $\mathcal{A}$ corrupting at most q_C users (OCorrupt oracle), and asking at most q_D decryption queries (ODecrypt oracle), is bounded by ε:

$$\mathrm{Adv}_{\Pi}^{\mathsf{ano\text{-}acca}}(\kappa, t, N, q_C, q_D) = \max_{\mathcal{A}}\{\Pr[\mathrm{Exp}_{\Pi,\mathcal{A}}^{\mathsf{ano\text{-}acca\text{-}1}}(\kappa, N) = 1] - \Pr[\mathrm{Exp}_{\Pi,\mathcal{A}}^{\mathsf{ano\text{-}acca\text{-}0}}(\kappa, N) = 1]\}.$$

This definition includes ANO-ACPA (for Chosen-Plaintext Attacks) when $q_D = 0$, and thus we denote the advantage $\mathrm{Adv}_{\Pi}^{\mathsf{ano\text{-}acpa}}(\kappa, t, N, q_C)$. When no corruption is allowed, we denote the advantage $\mathrm{Adv}_{\Pi}^{\mathsf{ano\text{-}cpa}}(\kappa, t, N)$.

2.7 Traitor Tracing

In this section, we define traitor-tracing (TT) and message-based traitor tracing schemes and the building blocks we will use in their construction. Traitor tracing schemes are encryption schemes that offer additional protection against "traitors", users who build and distribute decoders that can decrypt messages. Message-based traitor tracing schemes offer protection against a different kind of traitor, who redistributed the decrypted message instead of building a decoder. We follow an approach similar to [NSS99] by defining first a two-user primitive which we then extend to the multi-user case using fingerprinting codes.

We first state the *marking assumption*, which provides a way to embed a bit in a message block. This will be applied to blocks we protect, whereas no bit will be embedded in non-protected blocks. Then, from the decoded message, the authority will be able to extract the bits involved in the decryption keys in the pirate decoder, unless the decoder drops the protected blocks. We will thus need the property that nobody can detect which blocks are protected so that if the pirate decoder decides to drop some blocks, the choice will be independent from the protection of the blocks. We will show that we can build such a *message-traceable encryption* from a *2-user anonymous broadcast encryption* scheme. Eventually, from all the bits extracted from the protected blocks (and erasures in case of dropped blocks), using the tracing algorithm of a *fingerprinting code*, we can trace back some of the traitors.

2.7.1 Traitor Tracing

Definition 2.59 (Traitor tracing scheme)**.** A *traitor tracing scheme* (TT) is a 4-tuple of PPT algorithms (Setup, Encrypt, Decrypt, Trace):

- $\mathsf{Setup}(1^\kappa, N)$, where κ is the security parameter, and N is the number of users, generates the global parameters param of the system, an encryption key EK, a tracing key TK, and N user secret keys $\{\mathsf{usk}_i\}_{i\in[N]}$;
- $\mathsf{Encrypt}(\mathsf{EK}, m)$ takes as input the encryption key EK and a message m and outputs a ciphertext c;
- $\mathsf{Decrypt}(\mathsf{usk}_i, c)$ takes as input a decryption key usk_i and a ciphertext c, it outputs a message m, or the error symbol $\perp$.
- $\mathsf{Trace}^{D(\cdot)}(\mathsf{TK})$ takes as input the tracing key TK and has oracle access to a decoder D which decrypts ciphertexts. It outputs a user identity $i \in [N]$.

The correctness requirement is that for all $\kappa, N \in \mathbb{N}, (\mathsf{EK}, \mathsf{TK}, \{\mathsf{usk}_i\}_{i\in[N]}) \in [\mathsf{Setup}(1^\kappa, N)], i \in [N]$ we have $\mathsf{Decrypt}(\mathsf{usk}_i, \mathsf{Encrypt}(\mathsf{EK}, m)) = m$ with overwhelming probability.

Note that every user can decrypt all messages, as opposed to broadcast encryption. Traitor tracing schemes where the sender can decide to revoke users, which are then unable to decrypt a ciphertext, are called *trace-and-revoke schemes*. TT schemes where both EK and TK must be kept secret are called secret key TT schemes. If EK but not TK can be public, we say that the TT scheme is public-key, if both can be public, the scheme is a publicly traceable TT scheme.

Security notions are similar to those for (non-dynamic) BE. The requirements on the Trace-algorithm vary. It is usually assumed that the decoder D is *resettable* or *stateless*, meaning that its output can only depend on the current input, not on previous queries. This models software decoders. Hardware decoders, which could have protected memory and so be *stateful*, could refuse to decrypt once the queries deviate from regular encryptions, which indicate that it is being traced. In our definition, we depicted *black-box* tracing, where the tracing algorithm only has oracle access to the decoder. Tracing can be made easier by letting the tracing algorithm examine the decoder and look at the keys it uses, which is known as *white-box* tracing.

2.7.2 Marking Content

In order to trace from the message content itself, we need to be able to distribute different versions of a message to different users in an undetectable way. One way is to use watermarks. Another way could exploit different camera shots (angle and distance) of the same scene in a movie [BS98]. We abstract away from the concrete way to create versions and use the *marking assumption* that has been introduced by [BS98] to abstract from concrete schemes and has become standard since [KP10]. In the following we assume that, given two blocks m_0 and m_1,

- we can double m_b (for a random $b \in \{0,1\}$) into two *equivalent* messages m_b^0 and m_b^1
- when a user receives m_0' and m_1' (such that $m_b' \in \{m_b^0, m_b^1\}$ and $m_{\bar{b}}' = m_{\bar{b}}$, where $\bar{b} = 1 - b$), he cannot guess b.

This essentially means that it is possible to mark a message to protect it, but it is not possible to tell apart protected and unprotected blocks.

In addition, we also assume *robustness* with respect to a symmetric, reflexive relation $\approx_\rho$: for two equivalent blocks $m^0 \approx_\rho m^1$, when the user receives m^b, and tries to alter it (but without changing the meaning or content), he has only a negligible chance to output $m' \approx_\rho m^b$ that is closer to $m^{\bar{b}}$ than to m^b. This reflects that the user cannot change a watermark while preserving the message.

The robustness and the marking assumption guarantee that

- a protected block is indistinguishable from an unprotected block;
- when a user has access to one version of the protected block only, we can learn from its output which bit was embedded: the *detected bit*;
- when a user has access to both versions of the protected block, we either detect from its output one explicit bit as above, or we note that both versions have been used: in either case we can output one bit, associated to at least one version of the block available to the user.

Of course, the user can drop some blocks, but this impacts the quality of the message: we will assume that at most a fraction η of the blocks are dropped.

2.7.3 Fingerprinting Codes

We focus on binary codes, defined over the alphabet $\{0,1\}$. A *word* of length n is a string $w \in \{0,1\}^n$. An (n, N)-code is a set $\Gamma = \{w_1, \ldots, w_N\} \subseteq \{0,1\}^n$. The words which make up a code are called *codewords*. Fingerprinting codes [BS98] allow an authority to trace a subset of the users (the traitors) that colluded to produce a word (the pirate word, which is not necessarily a codeword) from the codewords they were given. This of course depends on the way traitors can derive words from their codewords: for any set of codewords $w_1, \ldots, w_t \in \Gamma$, the *feasible set* is the set of the words that can be derived from them: $\mathcal{FS}(w_1, \ldots, w_t) = \{w \in \{0,1\}^n \mid \forall i \in [n]\ \exists j \in [t] : w[i] = w_j[i]\}$.

The intuition behind this definition stems from the marking assumption: When a user with identity id is given a series of data blocks that contain watermarks corresponding to a codeword w_{id}, and he colludes with a user id' whose blocks are marked with $w_{\mathsf{id}'}$, they are unable to detect or modify the marks at the positions where their codewords agree. At the positions where their marks differ, they can remove the marks, but when the tracing authority recovers the codeword, it can put either 0 or 1 at these positions, since a mark can only be removed when the colluding users' codewords contain both 0 and 1 at this position.

Definition 2.60. A *t-fingerprinting code* $\mathcal{T}$ for $\mathcal{FS}$ is defined by a pair of algorithms $(\mathsf{Gen}, \mathsf{Trace})$, where

- $\mathsf{Gen}(N, \varepsilon)$ takes as input the number N of codewords to output and an error probability ε, it outputs a tracing key tk and a code $\Gamma \subset \{0,1\}^n$ of size N.
- $\mathsf{Trace}(\mathsf{tk}, w')$ takes as input the tracing key tk and a word $w' \in \mathcal{FS}(C)$, where C is a collusion of at most t codewords, and outputs a codeword w.

The running time of both algorithms must be polynomial in $N \log(1/\varepsilon)$, and the tracing algorithm should not be wrong too often: with probability less than ε, $w \notin C$.

More precisely, a t-fingerprinting code for $\mathcal{FS}$ guarantees that

- given $(\Gamma, \mathsf{tk}) \leftarrow \mathsf{Gen}(N, \varepsilon)$, with $\Gamma \subset \{0,1\}^n$ of size N
- for any collusion $C \subset \Gamma$ of size at most t, for any $w \in \mathcal{FS}(C)$, $\mathsf{Trace}(\mathsf{tk}, w)$ outputs a word in C with probability $1 - \varepsilon$.

Boneh and Franklin constructed the simple fingerprinting code $\Gamma_0(n, d)$, whose codewords are the rows in a permutation of the matrix $W(n, d) = (w_{i,j})_{i \in [n], j \in [d(n-1)]}$ with $w_{i,j} = 1$ if $i \leq \lceil j/d \rceil$, else 0, where

$\mathrm{Exp}^{\mathsf{ind\text{-}cca}\text{-}b}_{\Psi,\mathcal{A}}(\kappa, N, t, \varepsilon)$
$(\{\mathsf{usk_{id}}\}, \mathsf{EK}, \mathsf{TK}) \leftarrow \mathsf{Setup}(1^\kappa, N, t, \varepsilon)$; $\mathcal{Q}_D \leftarrow \emptyset$;
$(state, m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{ODecrypt}(\cdot,\cdot)}(\mathsf{FIND}; \mathsf{EK})$;
$c^* \leftarrow \mathsf{Encrypt}(\mathsf{EK}, m_b)$;
$b' \leftarrow \mathcal{A}^{\mathsf{ODecrypt}(\cdot,\cdot)}(\mathsf{GUESS}; state, c^*)$;
`if` $c^* \in \mathcal{Q}_D$ `then return` 0 `else return` b';

$\mathsf{ODecrypt}(\mathsf{id}, c)$
$\mathcal{Q}_D \leftarrow \mathcal{Q}_D \cup \{c\}$;
$m \leftarrow \mathsf{Decrypt}(\mathsf{usk_{id}}, c)$;
`return` m;

Figure 2.13: IND-CCA for message-traceable encryption

the same permutation π is applied to each line. In this code, the i-th codeword is $w_i = \pi(0^{d(i-1)}1^{d(n-i)})$.

$$W(4,3) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \Gamma(4,3) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

For $n \geq 3$, $\Gamma_0(n, d)$ is an n-fingerprinting code with error probability ε if $d = 2n^2 \log(2n/\varepsilon)$.

More efficient constructions of such codes can be found in [Tar08]: the resulting code length for a t-collusion-resistant fingerprinting code for N users with error probability ε is $\mathcal{O}(t^2 \log(N/\varepsilon))$, for any t. He also gives a lower bound, which is met by his construction. It is is thus asymptotically optimal, but contains a large constant factor of 100. In her PhD thesis, Charpentier [Cha11] lists further improvements, which reduced the constants hidden in the $\mathcal{O}$-notation.

Robust fingerprinting codes The t-fingerprinting codes we defined are sufficient to deal with adversaries that construct messages from the data available to the colluding users, possibly erasing watermarks when several versions of a marked block are available. The techniques fail against a stronger adversary, that does not deliver a full message, but drops some blocks of the message. By erasing parts of a message, he will also erase some watermarks, without allowing any conclusion about which bits the codewords contain at the place corresponding to the dropped blocks. This necessitates a stronger definition of a feasible set: $\mathcal{FS}^\star(w_1, \ldots, w_t) = \{w \in \{0,1\}^n \mid \forall i \in [n] \ : (w[i] = \star) \vee (\exists j \in [t] : w[i] = w_j[i])\}$.

Robust fingerprinting codes were constructed by Safavi-Naini and Wang [SNW03b] and Sirvent [Sir07]. Nuida [Nui10] gives the most efficient construction to date. He presents a generic conversion from a fingerprinting code to a robust fingerprinting code, which consists of repeating the code bits and adding dummy 0-bits. The code length for robustness parameter $0 < \delta < 1$, which is an upper bound on the fraction of marks that cannot be recovered, is asymptotically $\mathcal{O}((t \log t)^2/(1-\delta)^2 \log(N/\varepsilon))$, where the constant hidden by the $\mathcal{O}$-notation is roughly 21.

2.7.4 Message-Traceable Encryption

A message-traceable encryption scheme is a multi-cast encryption scheme which allows all the registered users (with a legitimate secret key) to decrypt a ciphertext. In addition, from the decrypted content, it is possible to derive the key (or even the keys) used for the decryption. In the following description, we focus on static schemes (the maximum number of users is set from the beginning):

Definition 2.61 (Message-traceable encryption)**.** A *message-traceable encryption scheme* (MT) is a 4-tuple of PPT algorithms ($\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Trace}$):

- $\mathsf{Setup}(1^\kappa, N, t, \varepsilon)$, where κ is the security parameter, N the number of users, t the maximal size of a collusion, and ε the error probability of the tracing algorithm, generates the global parameters param of the system (omitted in the following), N user secret keys $\{\mathsf{usk_{id}}\}_{\mathsf{id} \in [N]}$, an encryption key EK, and a tracing key TK.
- $\mathsf{Encrypt}(\mathsf{EK}, m)$ takes as input the encryption key EK and a message m and outputs a ciphertext c.
- $\mathsf{Decrypt}(\mathsf{usk_{id}}, c)$ takes as input a decryption key $\mathsf{usk_{id}}$ and a ciphertext c, it outputs a message m, or the error symbol $\perp$.
- $\mathsf{Trace}(\mathsf{TK}, c, m)$ takes as input the tracing key TK, a ciphertext c and the decrypted message m and returns an index $\mathsf{id} \in [N]$ of a user secret key $\mathsf{usk_{id}}$.

The correctness requirement is that for all $\kappa, N \in \mathbb{N}, (\mathsf{EK}, \mathsf{TK}, \{\mathsf{usk}_i\}_{i \in [N]}) \in [\mathsf{Setup}(1^\kappa, N)], i \in [N]$ we have $\mathsf{Decrypt}(\mathsf{usk}_i, \mathsf{Encrypt}(\mathsf{EK}, m)) = m$ with overwhelming probability.

$\mathrm{Exp}^{\mathsf{trace}}_{\Psi,\mathcal{A}}(\kappa, N, t, \varepsilon)$
$(\{\mathsf{usk}_{\mathsf{id}}\}, \mathsf{EK}, \mathsf{TK}) \leftarrow \mathsf{Setup}(1^\kappa, N, t, \varepsilon)$; $\mathcal{Q}_C \leftarrow \emptyset$;
$(state, C) \leftarrow \mathcal{A}(\mathsf{FIND}; \mathsf{EK})$;
$m^\star \xleftarrow{\$} \mathcal{M}$; $c^\star \leftarrow \mathsf{Encrypt}(\mathsf{EK}, m^\star)$;
$m \leftarrow \mathcal{A}(\mathsf{GUESS}; state, c^\star, \{\mathsf{usk}_{\mathsf{id}}\}_{\mathsf{id}\in C})$;
$T \leftarrow \mathsf{Trace}(\mathsf{TK}, c^\star, m)$;
`if` $m = \perp$ `or` $m \not\approx_\rho m^\star$ `then return` 0;
`if` $T \cap C = \emptyset$ `then return` 1 `else return` 0;

Figure 2.14: Traceability

Security notions As for any encryption scheme, the first security notion to define is indistinguishability against chosen-ciphertext attacks (IND-CCA), whose security game is presented in figure 2.13. Of course, to make tracing possible, the encryption algorithm will possibly derive several equivalent versions of the message m_b to be encrypted, which will decrypt to slightly different messages depending on the key used to decrypt. For this reason, we allow the adversary to choose which decryption key should be used by the decryption oracle, hence the additional input id.

Definition 2.62 (Confidentiality)**.** A message-traceable encryption scheme Ψ is $(\tau, N, t, \varepsilon, q_D, \nu)$-IND-CCA secure (indistinguishability against chosen-ciphertext attacks) if for all $\kappa \in \mathbb{N}$, in the security game presented in figure 2.13, the advantage $\mathrm{Adv}^{\mathsf{ind\text{-}cca}}_{\Psi}(\kappa, \tau, N, t, \varepsilon, q_D)$ of any τ-time adversary $\mathcal{A}$ asking for at most q_D decryption queries (ODecrypt oracle) is bounded by ν.

$$\mathrm{Adv}^{\mathsf{ind\text{-}cca}}_{\Psi}(\kappa, \tau, N, t, \varepsilon, q_D) = \max_{\mathcal{A}}\{\Pr[\mathrm{Exp}^{\mathsf{ind\text{-}cca\text{-}1}}_{\Psi,\mathcal{A}}(\kappa, N, t, \varepsilon) = 1] - \Pr[\mathrm{Exp}^{\mathsf{ind\text{-}cca\text{-}0}}_{\Psi,\mathcal{A}}(\kappa, N, t, \varepsilon) = 1]\}$$

This definition includes IND-CPA (for Chosen-Plaintext Attacks) when $q_D = 0$, and thus we denote the advantage $\mathrm{Adv}^{\mathsf{ind\text{-}cpa}}_{\Psi}(\kappa, \tau, N, t)$.

We now formalize the additional security notion of *traceability*: after having received at most t secret keys (the collusion C of traitors), the adversary asks for a ciphertext $c^\star$ of a random message $m^\star$, and outputs a plaintext m that should be equivalent to $m^\star$. The tracing algorithm should then output one of the traitors, otherwise the adversary has won the game. We use the relation $m \approx_\rho m'$ from section 2.7.2 to denote that two messages are "similar" in the current context. If the adversary sends a random message (hence $m \not\approx_\rho m^\star$) or alternatively outputs an empty message, we say the adversary lost the game:

Definition 2.63 (Traceability)**.** A message-traceable encryption scheme Ψ is $(\tau, N, t, \varepsilon, \nu)$-traceable if for all $\kappa \in \mathbb{N}$, in the security game presented in figure 2.14, the success probability $\mathsf{Succ}^{\mathsf{trace}}_{\Psi,\mathcal{A}}(\kappa, \tau, N, t, \varepsilon)$ of any τ-time adversary asking for at most t secret keys is bounded by ν.

$$\mathsf{Succ}^{\mathsf{trace}}_{\Psi}(\kappa, \tau, N, t, \varepsilon) = \max_{\mathcal{A}}\{\Pr[\mathrm{Exp}^{\mathsf{trace}}_{\Psi,\mathcal{A}}(\kappa, N, t, \varepsilon) = 1]\}.$$

Chapter 3

State of the Art

Contents

Before we start developing new schemes, we want to examine the schemes that already exist and compare their efficiency and security. Efficiency is measured by the length of ciphertexts as well as decryption and encryption keys, while security is measured by the security notion achieved and the hardness assumption that is necessary, as well as the tightness of the reduction (where a reduction is given in the paper). A ciphertext usually consists of three parts: an id header that identifies the target set, a key header that encapsulates a symmetric key, and the encryption of the actual message under this symmetric key. Since only the key header depends on the BE scheme, "ciphertext length" usually refers to the length of the key header. We usually give the lengths of keys and ciphertexts in group elements

Private-key	|KH|	|DK|	|EK|	Security	Assumption
CS [NNL01]	$r \log \frac{N}{r}$	$1 + \log N$	$2N - 1$	a1 CCA1	–
SD [NNL01]	$2r - 1$	$\frac{1}{2} \log^2 N + \tilde{\mathcal{O}}(1)$	1	a1 CCA1	PRNG
LSD [HS02]	$d(2r - 1)$	$\mathcal{O}(d \log^{1+\frac{1}{d}}(N))$	1	a1 CCA1	PRNG
SD-MK [Asa03]	$2r - 1$	$\frac{1}{2} \log^2 N + \tilde{\mathcal{O}}(1)$	1	a1 CCA1	RSA
CS-TOWP [NK05]	$r \log \frac{N}{r}$	1	1	key intrac	TOWP
SD-TOWP [Asa04]	$2r - 1$	$\frac{1}{2} \log^2 N + \tilde{\mathcal{O}}(1)$	1	key intrac	TOWP
CS-Rabin [AK05]	$r \log \frac{N}{r}$	1	1	key intrac	TOWP
CS-ary [Asa02]	$r(\frac{\log(N/r)}{\log a} + 1)$	1	1	key intrac	RSA
GR-CS [GR04]	$r(\frac{\log(N/r)}{\log a} + 1)$	1	1	key intrac	RSA
KCT [WNR04]	$4r$	$2 \log N$	1	key intrac	CROWHF
SSD [GST04]	$4r - 2$	$2 \log N$	1	key intrac	CROWHF
LSIC$[k]^{acc}$ [AI07]	$2rd$	1	$\mathcal{O}(N^2)$	key intrac	RSA

Table 3.1: Comparison of subset cover-based BE schemes

or other objects instead of bits, which hides a factor that depends on κ in a way that varies according to the type of group as explained in section 2.3.4. The number of group elements will often depend on the total number of users N, on s, the size of the target set S, or on $r = |R|$, the number of revoked users.

The length of the key header is denoted by |KH|, the length of the decryption and encryption keys by |DK| and |EK|.

3.1 Evolution of Tree-based BE Schemes

Fiat and Naor [FN94] formalize BE and introduce two private-key BE schemes, one using OWFs and hash trees, the other based on root extraction modulo a composite. In the first scheme, a 2-expanding PRG is used to construct a hash tree from a randomly chosen root. Each user u is assigned a leaf and receives as decryption key the $\log N$ roots of the subtrees obtained by removing the path between u and the root from the tree. He can therefore derive all keys but the one corresponding to his leaf. The symmetric key is then the XOR of all the leaves of the revoked users.

In the second scheme, the centre chooses a random RSA modulus M, and a secret element $g \in \mathbb{Z}_M^\times$. Each user u is assigned a public prime p_u, and given g^{p_u} as his key. By the RSA assumption (section 2.3.2), it is hard to recover g without knowing the factorization of M. The exponent for a given target set S is $e = \prod_{u \in S} p_u$, and the symmetric key is g^e. Their basic schemes are not collusion-resilient, but require no key header to be transmitted as the session key is implied by the id header. Then the authors present two ways to increase the collusion resistance of their scheme. In "one-level schemes", to achieve t-collusion resiliency the message M is split into $l = t \log N$ parts as $M = \bigoplus_{i=1}^{l} m_i$, where $l - 1$ of the m_i can be chosen uniformly at random. Then several instances of a 1-resilient BE scheme can be run in parallel to obtain t-resiliency. To increase the efficiency, they propose "multi-level schemes" which increase the collusion resistance of schemes. This results in a (t, n) collusion-resistant scheme where the length of the key header is $\mathcal{O}(t^2 \log^2 t \log n)$ and the length of the decryption key is $\mathcal{O}(t \log t \log n)$.

We now give an overview of several later tree-based broadcast encryption schemes. Their most important characteristics are summarized in table 3.1. All schemes enjoy full collusion resistance. The "security" column lists the security property, either CCA with adaptive corruption, but the adversary has to guess after receiving the challenge ciphertext without making further queries to any oracle, or key intractability, meaning the adversary is unable to compute the secret keys of any non-corrupted user.

Remark 3.1. *Key intractability does not guarantee that keys are indistinguishable from random keys. We consider this notion to be too weak, since it does not allow replacing keys of uncorrupted users with random and unrelated keys in game-hopping proofs (the schemes do not compose well).*

3.1.1 Logical Key Hierarchy

We first describe two access control schemes and a multicast encryption scheme that are the basis for several newer BE schemes. Hierarchical access control schemes maintain keys in such a way that users can compute the keys of all users below them in the hierarchy. Multicast encryption schemes are

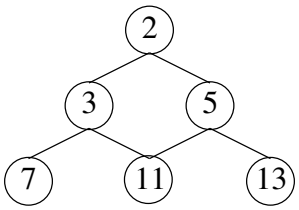

Figure 3.1: In [AT83], nodes in a poset-tree are assigned primes.

designed to maintain a long-term group key for a changing set of users, where the key must be updated every time a user leaves the group.

Akl and Taylor [AT83] describe a hierarchical access control system for a partially ordered set (poset) with access relation "$\leq$", where $u_i \leq u_j$ means that user u_j should be able to derive the key of user u_i. Each node u_i in the access tree is assigned a small prime p_i, as shown in figure 3.1, and a key K is chosen at random. Then they define the exponent $e_i \coloneqq \prod_{u_n \not\leq u_i} p_n$ and give the key K^{e_i} to user u_i. In the example, to compute the exponent for node 2, which was assigned the prime 3, one would compute $e_2 = 2 \cdot 5 \cdot 13 = 130$, while the root node would compute the empty product $e_1 = 1$. If $u_i \leq u_j$, then the set $\{u_n : u_n \not\leq u_j\} \subseteq \{u_n : u_n \not\leq u_i\}$. Therefore $e_j | e_i$ iff $u_i \leq u_j$, so $\frac{e_i}{e_j}$ is an integer and user u_j can compute $K_i = K_j^{e_i/e_j}$. In our example, $u_2 \leq u_1$, and clearly $1|130$.

Chick and Tavares modify this scheme in [CT90] to obtain master keys for a set of N service keys (which are partially ordered themselves). The idea is that it should be easy to calculate a service key from a master key iff the service key is in the subset associated to the master key. In their scheme, only the services are assigned primes. Define $T \coloneqq \prod_{i=1}^{N} p_i$ and for each service i, $t_i \coloneqq \prod_{S_n \leq S_i} p_n$. Then define the service keys $SK_i \coloneqq K^{T/t_i}$ and the master keys $MK_i \coloneqq K^{T/v_i}$ where $v_i \coloneqq \prod_{SK_n \leq MK_i} p_n$. The service keys can be calculated from each other as in the Akl-Taylor scheme, and because the fraction v_j/t_i is an integer for $MK_j \leq SK_i$, $MK_i^{v_j/t_i} = SK_i$. New services can be added after setup without modifying existing keys, if the service is not subordinate to any other service.

LKH Wallner et al. [WHA99] and Wong et al. [WGL00] independently proposed a multicast encryption scheme based on "tree key graphs" that is known as *logical key hierarchy*. They distribute keys as in the CS method detailed in section 3.1.2: In a system with N users, they construct a tree with N leaves, and for each node in the tree, a key is drawn at random. A user is given the keys corresponding to "his" leaf and all its ancestors. If a user u is revoked, all users update the keys they have in common with the revoked user u. To accomplish this, the centre chooses a new key for each node v on the path from u to the root. Let $p(v)$ denote the parent of v, $s(v)$ the sibling of v. Then the centre encrypts the key associated to $p(u)$ with the secret key associated with $s(u)$, because $s(u)$ is the only node that should know this key. Going up, for all nodes v above $p(u)$, the centre encrypts the key associated with v twice, using the keys associated with its children (one of them is old and uncompromised, the other one is new and contained in the previous encryption). In this way, the whole update takes $1 + 2\log N$ messages.

The transmission complexity of the update is reduces to $1 + \log N$ in [CGI+99] by using a PRNG. A random seed r is encrypted with the key of $s(u)$. Then a 2-expanding PRNG with right output half R and left output half L is used to derive $R(r), R(R(r)), \ldots$. Then $L(r)$ is the new key associated with $p(u)$, and it can only be derived by $s(u)$. $R(r)$ is encrypted with the key of $s(p(u))$, and $L(R(r))$ is the new key of $p(p(u))$, $R(R(r))$ is encrypted with the key of $s(p(p(u)))$, and the key $L(R(R(r)))$ is associated with $p(p(p(u)))$ and so on.

Tradeoffs between communication, user storage and centre storage can be found in [CMN99], along with upper and lower bounds.

3.1.2 The Subset Cover Framework

Some of the best-known private-key BE schemes are based on the *subset-cover* (SC) framework defined in [NNL01]. In this framework, long-term symmetric keys L_i are assigned to sets S_i of users, and given to each user in S_i. To broadcast a message to a receiver set S, S is partitioned into disjoint subsets S_i, and the session key is encrypted with each corresponding L_i. The centre then broadcasts the indices

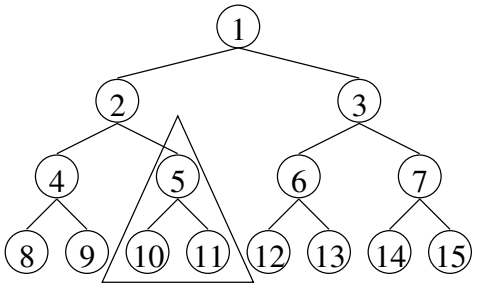

Figure 3.2: In this CS tree, two users can read messages encrypted to node 5.

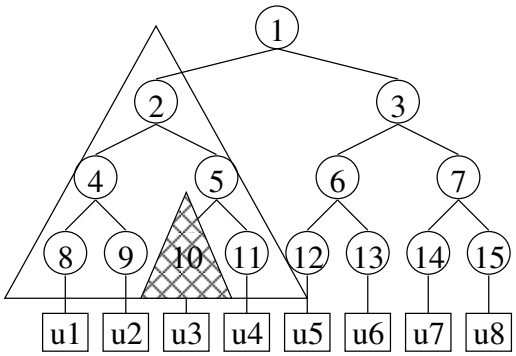

Figure 3.3: In SD, a set is the difference between two subtrees.

i of the targeted sets S_i and the encryptions of the session key under L_i. The header length therefore depends on the number of subsets that make up S. The paper describes two possible instantiations of the SC framework: The complete subtree method and the subset difference method. These instantiations are specified by the collection of subsets, the key assignment, and the way to cover the target set with subsets.

CS In the *complete subtree* (CS) method, the subsets S_i correspond to all complete subtrees of the user tree. The user tree is a balanced binary tree that has the users as leafs. In the original scheme, each node v_i of the tree is the root of the set S_i that contains all its descendants. Each v_i is assigned an independent and random key L_i. Then each user stores the $1 + \log N$ keys on the path to the root node. In the example in figure 3.2, user 3 at leaf number 10 receives the keys of nodes 1, 2, 5, and 10. To encrypt a message to users 3 and 4, the centre will use the key assigned to node 5.

To cover a target set S, the sender chooses the maximal sets S_i that do not contain a revoked user. This can be accomplished by coloring the leafs of the revoked users and all their ancestors black. The black nodes form the Steiner tree $ST(R)$ of the set R of revoked users. The set S is then covered by choosing the non-black subtrees. By a short analysis, it can be shown that for r revoked users, there are at most $r \log \frac{N}{r}$ of them, which is also the length of the header. The (secret) encryption key is of length $2N - 1$, one key for each node. If information-theoretic security is given up, this could reasonably reduced to storing just a single secret key from which all parent labels can be computed. Each user must store his $\log N$ ancestors in the tree in addition to his secret key, and perform a search of his ancestors in the identification header, and one decryption to retrieve the session key.

SD In the *subset difference* (SD) method, a subset $S_{i,j}$ is the subtree rooted at v_i minus the subtree rooted at v_j (it is required that v_i is an ancestor of v_j). A leaf u is in $S_{i,j}$, iff v_i is an ancestor of u, but v_j is not. Each $S_{i,j}$ is associated with a key $L_{i,j}$ and a user is given all keys corresponding to the subsets it is a member of. In the example in figure 3.3, the depicted set $S_{2,10}$ contains users 1, 2, and 4, but not 3.

The keys $L_{i,j}$ are computed in a way similar to the OWF-based scheme from [FN94]. First, each node v_i is assigned a uniformly independent label Lab_i. Given a 3-expanding PRNG G, the output of $G(Lab_i)$ is split into G_L, G_M, G_R. G_L (G_R) is the label of the left (right) child, G_M is the key of the node. Let $Lab_{i,j}$ be the label of node v_j derived from Lab_i. Then we define the key $L_{i,j}$ associated with

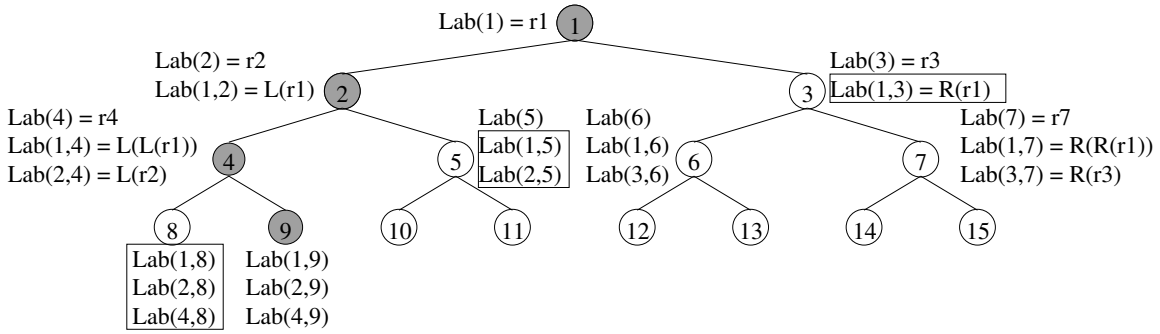

Figure 3.4: In this SD tree, user 2 (at leaf 9) receives the boxed labels.

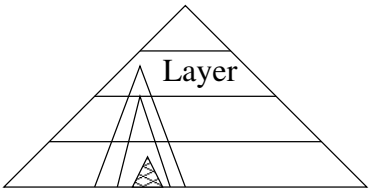

Figure 3.5: Every SD set can be represented as the union of two useful sets.

set $S_{i,j}$ to be $G_M(Lab_{i,j})$. This key can be computed from any label $Lab_{i,k}$ where v_k is a descendant of v_i and an ancestor of v_j.

For every subtree T_i rooted at v_i, a leaf $u \in T_i$ should be able to compute $L_{i,j}$ iff v_j is not an ancestor of u. We depict this graphically in figure 3.4. All ancestors of u in T_i are colored grey. The grey nodes b are those nodes for which u is not in $S_{i,b}$. Therefore we give to user u the labels of all non-grey nodes that have a grey parent. This is repeated for all $1 + \log N$ trees T_i that contain u.

The number of labels stored by each user is $\frac{1}{2}\log^2 N + \frac{1}{2}\log N + 1$. The sender needs to store all $2N - 1$ parent labels assigned to each node, but this could reasonably reduced to storing just a secret key from which all parent labels can be computed (e. g. as the encryption of their identifier). Any cover can contain at most $2r - 1$ subsets. Each user must store the $\log N$ identifiers of his ancestors, so he can find the subset he is a part of. If the revoked users are chosen at random, experiments show a bound of $1.25r$.

LSD The *layered subset difference* (LSD) method from [HS02] is an improvement of the SD method. For the basic method, the main observation is that $S_{i,j} = S_{i,k} \cup S_{k,j}$ if k is on the path from i to j. The root and every level of depth $k \cdot \sqrt{\log(N)}$ are denoted "special levels", the levels between them are called "layers". "Useful sets" are those $S_{i,j}$ where i is in a special level or i and j are in the same layer. Then every set $S_{i,j}$ is either useful or the union of two useful sets, as depicted in figure 3.5. With this technique, the size of the decryption keys shrinks to $\mathcal{O}(\log^{3/2}(N))$ at the cost of increasing the ciphertext by at most a factor of 2 to $4r - 2$. If the revoked users are chosen randomly, experiments give a bound of $2r$. Since the LSD method uses only a subset of the sets defined in the SD method, the other properties such as length of the encryption key, collusion resistance, and security are preserved.

The authors also present a generalized version of the basic scheme by introducing several degrees of specialness for the levels and representing each set as the union of several useful sets. A set $S_{i,j}$ is now useful if j is in the next 1-more special level from i or if i is more special than j. This change yields further asymptotic improvement: For d "degrees of specialness", the length of the decryption key is in $\mathcal{O}(d \log^{1+\frac{1}{d}} N)$ at the cost of increasing the length of the key header to at most $d(2r-1)$. The generalized version is described in more detail in subsection 3.5.1.

3.1.3 Improvements

Several papers were written to improve the characteristics of the previous methods. The common denominator of these efforts is the derivation of keys belonging to nodes of the tree by applying the inverse of a trapdoor-OWF to the root key. Then users can compute the keys of their ancestor nodes by applying the OWF to the value they were given. This is very effective for the CS method, but less so

when applied to the SD or LSD methods, where subsets can consist of several subtrees. The reason for that is that in the CS method, keys are computed only bottom-up by the users, but in the SD method, the values assigned to a subset are computed top-down when the subset consists of several trees. For example, in figure 3.4 all users in the subtree rooted at node 3 will compute the label $Lab_{1,8}$ from the label $Lab_{1,2}$ given to them. Clearly, $Lab_{1,2}$ must not be computable from $Lab_{1,8}$, so if it is computed from another label, the downward derivation must be left intact.

SD-MK [Asa03] is another improvement by Asano based on the master key technique from [CT90]. In the SD method, each user is assigned $\frac{1}{2}\log^2 N + \frac{1}{2}\log N + 1$ labels. Asano proposes to derive those $\log N$ labels $Lab_{i,j}$ where j is a child of i and the one that covers the whole subtree from a master label, thereby subtracting $\log N$ from the size of the decryption key. The derivation step is done exactly as in [CT90] and described above. The method requires $2N - 1$ public primes of which $1 + \log N$ must be stored (or computed) by the receivers. Storing a prime costs roughly $\log N$ bits per prime, computing one can be done in $\mathcal{O}(\log^4 N)$ time. After that, the multiplication of $\log N$ primes costs about $\mathcal{O}(\log^4 N)$, and the exponentiation $\bmod M$ around $\mathcal{O}(\log^2 N \log^2 M)$.

The LSD method can be improved in the same way, because all labels $Lab_{i,j}$ where j is a child of i are defined in LSD as well. Therefore the the size of the decryption key is again reduced by $\log N$. The authors claim key indistinguishability for their scheme with a short argument based on the pairwise independence of the output of the hash function H.

CS-TOWP Nojima and Kaji [NK05] give an improvement of the CS method, which uses a trapdoor one-way permutation (TOWP) f to reduce the size of the secret key stored by each user to $\mathcal{O}(1)$. First, a public label l_i is associated to each node i in the tree. Then the centre assigns the root node a randomly chosen key. If the key of a node i is K_i, its child j is assigned the key $K_j = f^{-1}(K_i \oplus l_j)$. In this way, each node is assigned a key, and the users are given the key assigned to their leaf. A node j can then compute the key of its parent i as $K_i = f(K_j) \oplus l_j$. The header size is the same is in the standard CS method, but the users need to store only a single secret key of constant size. Each user needs to store the labels l_j, but they can be generated by applying a hash function to the user id. He also needs to store a description of the public TOWP (typically a RSA exponent). The time complexity for decryption is $\log N$ evaluations of the TOWP and the hash function to derive the labels.

A disadvantage of this scheme is that it only achieves the security notion of *key intractability*, not *key indistinguishability*. A user can distinguish the key of any node in the tree (which the session keys are) from a random key by computing the key of the root node starting with his key, then compute the key of the root node starting from the other key. In the paper the hash function is modeled as a RO, although the authors speculate that this might not be necessary.

SD-TOWP [Asa04] is a modification of the SD method, where Asano applied the TOWP modification due to Nojima and Kaji. In the same paper, he proposes a LSD-TOWP method. Similar to the SD-MK method, the $\log N + 1$ labels chosen at random by the centre are replaced by a pseudorandom construction. Each user has to store $\frac{1}{2}\log^2 N - \frac{1}{2}\log N + 1$ labels and the public RSA exponent. The construction is described in full in section 3.5.1.

Rabin trees are used by Asano and Kamio to further reduce the computational cost of the Nojima-Kaji CS-TOWP construction. The most efficient TOWP is based on RSA and has a computational cost of $\mathcal{O}(\log e \log^2 m)$ per exponentiation where e is the exponent and m the modulus. One way to reduce the computational cost is to use a modified Rabin tree, which reduces the time complexity for decryption to $\log N$ squarings [AK05]. To ensure that the root of a value is again a quadratic residue, a salt value is appended to the user label and this value is hashed into the group. To compute the labels, each user has to store $\log N$ salt values.

***a*-ary trees** are used to reduce the ciphertext length in [Asa02]. At the same time, the master key technique [CT90] is used to reduce the length of the decryption keys. Table 3.1 describes only method 1 of the paper, but under the assumption that primes are stored instead of computed on the fly. Storage of the primes costs $\mathcal{O}\left((\frac{2^{a-1}-1}{\log a}\log N + 1)(\log N + a + \log(\log N + a)\right)$ bits. Method 2 trades a longer decryption key for a reduced computational complexity at the receiver, but with both methods, receivers have a computational cost that is exponential in a.

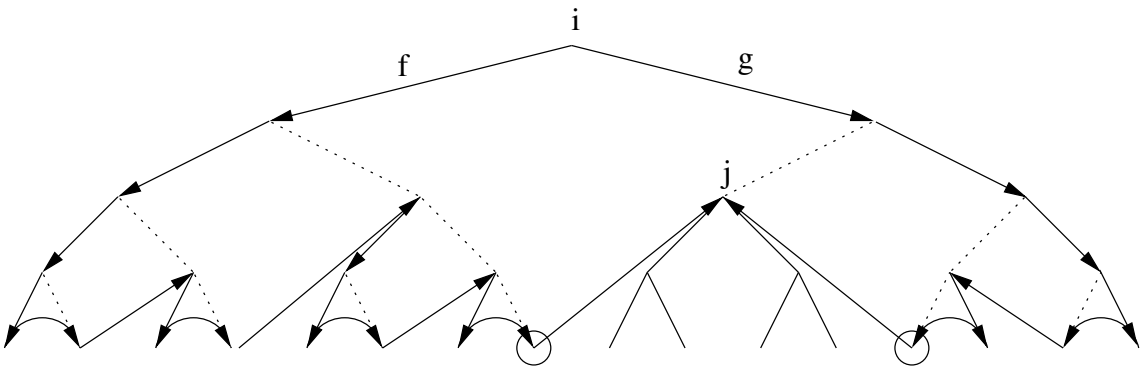

Figure 3.6: Key derivation paths for the subset $S_{i,j}$ in SSD

Gentry and Ramzan [GR04] use RSA-based access control to improve and generalize Asano's a-ary construction. They separate the problem of constructing an efficient scheme into two parts. The first part is to generate a combinatorially optimal poset for a given BE problem, the second to generate an optimal set of keys for a given poset. For the second problem, they use the accumulator-based solution of [AT83]. To solve the first problem, they use a variation of the LSD technique of [HS02]. The length of the key header is $r(\log \frac{N}{r} + 1)$ as for Asano's scheme. Each user needs to store one key. An educated guess of the storage required for the primes is $\log N$ times the computational cost: $\mathcal{O}\left(((2a - \log a - 1)\log N)(\log N + a + \log(\log N + \log a)\right)$ bits as in the a-ary scheme. The cost for key derivation is given as $(2a - \log a - 1)\log N$ exponentiations. Using bounds due to Luby and Staddon [LS98] for the tradeoff between λ_{DK} and λ_{KH}, Gentry and Ramzan can even show some optimality in that respect. The scheme is described in more detail in section 3.5.2.

Key chain trees were independently used by Goodrich, Sun, and Tamassia [GST04] and Wang, Ning, and Reeves [WNR04] to combines the efficiency of CS and SD: Users need to store $\mathcal{O}(\log N)$ keys and ciphertexts are of size $\mathcal{O}(r)$. Wang et al. apply the hash chains to a tree with users at the leaves. For each subtree S_i, they need two key chains, which allows them to remove any number of successive users from S_i. This allows them to cover the target set with subsets of the form $S_i \setminus [a, b]$, where $[a, b] \subset \mathcal{S}_i$, resulting in key headers of size $2r$. To reduce the computational load on the receivers from $\mathcal{O}(n)$ to $\mathcal{O}(\sqrt{n})$ steps, they use a layered scheme with key header length $4r$.

Goodrich et al. applied the same idea to the SD construction. The drawback of the SD method is that users have to store $\mathcal{O}(\log^2 N)$ keys, intuitively they store keys for subsets $S_{i,j}$ and there are $\log N$ candidates for both i and j. The decryption key size can be reduced by employing bidirectional hash chains, where key derivation is based on a collision-resistant one-way hash function (CROWHF), so that for a given i, two keys are sufficient to derive all keys for sets $S_{i,j}$ for any j. The idea is to use two hash chains to traverse the tree depth-first, once using a function f and a left preorder, and a second time using a function g and a right preorder. Each node b is associated with two secret values $L(b), R(b)$ derived from the secret of the subtree's root by repeated application of the hash function, is given the secret value $f(L(b))$ and $g(R(b))$ of its successor in each direction as the secret key. To encrypt a session key to a subtree $S_{i,j}$, the session key is encrypted twice, once each with the key of the left and the right predecessor of j (circled in figure 3.6). The left-preorder keys are used as the common key for the left subtree of $S_{i,j}$, and right-preorder keys are used for the right subtree, and children of j are unable to compute either. To decrypt, a user first finds whether he is in the left or right subtree, then repeatedly applies either f or g to his key (as shown in figure 3.6), until he derives the key of j's predecessor, which was used to encrypt the session key.

This way, each user needs to store only $2 \log N$ secret values, but the ciphertext has only twice the length of an SD ciphertext (since for each subset, two encryptions are necessary). The paper proposes to use independent random seeds for the key of each node i, but they could as well be generated from a small seed using a secure PRNG. Due to the way the keys are generated, the scheme can only claim key intractability, for which a brief argument is sketched in the paper.

They also give a way to decrease the computational load of a user at the cost of longer ciphertexts and keys by dividing the user tree into layers in the way of the LSD scheme. They call this variant "stratified subset difference" (SSD).

Graph decomposition is used by Attrapadung and Imai [AI07] to give a generic and elegant description of constructions in the subset-cover framework. They separate the task of constructing a set system (such as CS or SD) that gives a good key-header size from the task of finding a graph decomposition of this set system that allows for short keys. The authors propose subset incremental chain (SIC) and layered SIC (LSIC) as set systems. The graph decompositions are the PRG-based tree decomposition and the RSA-based chain decomposition. For example, the LSIC$[k]^{acc}$-construction has ciphertexts consisting of $2rd$ elements for a variable d which determines the computational cost $\tilde{\mathcal{O}}(\frac{1}{d}n^{1/d})$ of the receiver. The size of a user secret key is constant, but encryption keys are large (quadratic in N).

3.1.4 Public-Key Subset Cover

The SC framework has been extended to the public-key world. One approach implements the tree structure using (H)IBE (identity-based encryption), the other uses ABE (attribute-based encryption) and is described in section 3.6.2.

The SC framework is extended to the public-key setting in [DF03a]. Dodis and Fazio use IBE as a primitive for the CS and HIBE for the SD method. To model the CS method, each node in the tree is given a name, and each user is given the decryption key for all his predecessors in the tree. Key header length and the key sizes must be multiplied by the respective values of the IBE scheme used.

To model the SD method, each node is given a name prefixed by the name of its parent. and a secret key P_i is derived for each node v_i using the MSK. These P_i are the analogues of the Labels Lab_i chosen at random for each node. For each subset $S_{i,j}$, a key $P_{i,j}$ is derived from P_i using the name of v_j. The $P_{i,j}$ are the analogues of the $Lab_{i,j}$ derived from the Lab_i. The private key $L_{i,j}$ is then extracted from $P_{i,j}$ using the name of v_j appended by a special terminator symbol that does not appear in any name. The LSD method can be instantiated in the same way because it uses only a subset of the SD subsets.

Due to the limitations of HIBE schemes at the time this paper was written, the construction was quite inefficient. The HIBE scheme used by the authors had ciphertext length proportional to the depth of the hierarchy, and so the SD scheme required $\mathcal{O}(r \log N)$-length ciphertexts. Substituting a HIBE scheme with constant-size ciphertext such as [BBG05] reduces the ciphertext length of this construction to $\mathcal{O}(r)$.

3.2 Broadcast Encryption from Polynomial Interpolation

Many BE schemes based on polynomial interpolation are trace-and-revoke schemes, which additionally offer tracing capabilities. Tracing is describe in section 3.9.

The first reference to the problem of broadcast encryption can be found in a 1991 paper by Berkovits [Ber91]. He derives a one-time broadcast encryption scheme for N users from a $(N+1, 2N+1)$ secret sharing scheme, using as an example Shamir's secret sharing scheme, which is based on polynomial interpolation. In the setup phase, he distributes points to all N users. To broadcast, he finds a polynomial of degree N that passes through the point $(0, K)$ and all the points of the users in the target set, but not through the points of users in the revoked set. The ciphertext then contains N additional points on the curve, which all users in the target set can combine with their own secret point to interpolate the polynomial.

This one-time BE scheme was generalized to revoke at most t users in [NP01]. The underlying idea is to obtain a t-collusion resistant BE scheme for N users by doing a $(t+1, N+t)$-secret sharing, and revealing t shares, which can include the shares of up to t revoked users. Each user in the target set can combine his share with the t public ones to recover the secret, which will serve as a symmetric key. Even if all revoked users collude, they have only t shares and cannot recover the secret. This construction yields an information-theoretically secure one-time BE scheme.

The centre draws a random polynomial f of degree t and gives a different point $f(u)$ on this polynomial to each user u. To distribute the random session key, the centre broadcasts t points on this polynomial to all users. These t points contain the points of up to t revoked users. Each non-revoked user does a polynomial interpolation using his point and the t points in the ciphertext to find the session key $f(0)$. Each revoked user finds that his point is in the t broadcasted points and that t points are not sufficient to reconstruct f. Several properties of PI-based BE schemes are already obvious from this sketch: Secret keys are constant-size, the ciphertext is linear in t, which provides an interesting tradeoff, and the scheme can be at most t-collusion resistant, with catastrophic failure if more than t users collude to recover the polynomial, i. e. the master secret key. Furthermore, the scheme is dynamic, as new shares can be generated when a user joins.

The first public-key BE schemes were based on PI and proposed independently by Naor and Pinkas [NP01], and Tzang and Tzeng [TT01]. Both schemes extend the above one-time BE scheme to allow multiple broadcasts by randomizing the polynomial for each broadcast. The random factor has to be included in the broadcast, and to avoid derandomization by curious users, the interpolation is done "in the exponent" of elements of a group in which DDH is hard. The NP scheme also supports *self-enforcement*, where a user risks leaking sensitive data when he shares his secret key, and traitor tracing. Both papers do not define security, but both schemes are IND-SCPA secure under the DDH assumption in the standard model. There is a second scheme in [TT01], which claims (static) IND-CCA2 security. Dodis and Fazio [DF03b] remark that this scheme loses its CCA security if only a single user is corrupted, even if this user is revoked.

Dodis and Fazio [DF03b] improved on the previous constructions. In their scheme, the encryption key contains $3t+5$ group elements and the description of a CRHF, the master secret key and user decryption keys consist of six group elements, and the ciphertext of $2t+3$ group elements plus a MAC tag. The scheme has adaptive IND-CCA2 security (where the adversary can query a corruption oracle before the challenge phase) under the DDH assumption. It is of course only t-collusion resilient.

Yoo et al. build on the scheme by Naor and Pinkas to construct a BE scheme with no limitation on the size of the target set [YJCK04]. They divide the user set into m partitions, and each user receives a point on each of w random polynomials of increasing degree. This makes it possible to use for each partition a polynomial whose degree is just high enough to revoke the users in this partition, keeping the total size of the key header small, while the partition size bounds the computational overhead of the users because they have to interpolate polynomials of relatively small degree.

3.3 Algebraic Constructions of Broadcast Encryption Schemes

While the constructions based on trees and secret sharing are conceptually more abstract, there are constructions that directly exploit properties of algebraic structures. In section 3.3.1, we consider a variety of pairing-based schemes that include very efficient constructions. Section 3.3.2 highlights some recent developments in BE based on lattices.

3.3.1 Pairing-based Broadcast Encryption

BGW The first BE scheme with constant-size ciphertext was described by Boneh, Gentry, and Waters in [BGW05]. They assume a symmetric pairing in a prime-order group. Since the scheme is not dynamic, parameters can depend on the number of users N. We call the first scheme BGW_1. The encryption key and the decryption key of a user both consist of $2N+1$ group elements, but only one group element of the decryption key needs to be kept secret. The key header (ciphertext) consists of two group elements. The scheme is statically IND-CPA secure in the standard model under the N-BDHE assumption, and the reduction is tight.

Note that the authors do not distinguish between encryption and decryption keys, but between public and private keys, where the decryption algorithm takes both the private and the public key as input. In our notation, the encryption key is the public key, but the decryption key consists of both the public and the private key, which is all the information necessary to decrypt a ciphertext.

This allows a slight modification of the scheme, where the elements of the public key which are necessary to decrypt are appended to the ciphertext instead of asking the user to store them. This results in a scheme BGW_1', where a user decryption key is only two group elements, and the ciphertext consists of $2+|S|$ group elements.

To balance the length of the key header against the length of the keys, the authors propose to split the user set into A sets of N/A users and run A instances of the scheme in parallel, one for each set. For $A \approx \sqrt{N}$, this gives a system BGW_2, where the ciphertext and all keys consist of $\mathcal{O}(\sqrt{N})$ group elements. Security is then based on the $\sqrt{N}$-BDHE assumption, and the reduction is tight.

Using the same trick as for the first scheme, the decryption key can be reduced to two group elements. We call this scheme BGW_2'.

The paper also contains a CCA secure scheme, which we denote BGW_3. The construction is based on the CHK-transform (sec. 2.5.6) and uses a SUF-CMA secure signature scheme or MAC.

An extension of the BGW scheme was presented by Boneh and Waters in [BW06]. The scheme uses a symmetric pairing in a composite-order group. The asymptotic efficiency of the scheme is the same as for BGW, except that the secret part of the decryption key is also of size $\mathcal{O}(\sqrt{N})$. The scheme allows tracing, and is adaptive-1 IND-CPA secure in the standard model under three non-q-type assumptions.

GW Gentry and Water presented an adaptively IND-CPA secure (where the adversary can corrupt users before seeing the challenge ciphertext) BE scheme in [GW09]. They proceed by constructing a scheme that is semi-statically secure, meaning that before setup, the adversary must commit to a set of users he will not corrupt. Their construction uses a symmetric pairing in a prime-order group and security is based on the $2N$-BDHE assumption. The public key consists of $N + 2$ group elements in $\mathbb{G}$ and 1 element from $\mathbb{G}_T$, the master secret key is a single group element. Each user secret key consists of N group elements, and a ciphertext is only 2 group elements.

Then they give a transformation to an adaptively secure scheme where every user is associated with two identities, but only knows the secret key of one of them. The transformation doubles the ciphertext sizes and appends an additional tag of $|S|$ bits. In the ROM, the additional tag can be avoided.

LSW Lewko, Sahai, and Waters proposed the first interpolation-based revocation schemes with no bound on the number of users that can be revoked. Their constructions are based on symmetric bilinear maps in prime-order groups [LSW10].

By layering two secret-sharing schemes, the number of revoked users can be chosen at the time of encryption. To revoke r users, they split a secret into r shares. Each share is then split again into two shares, and the ciphertext is constructed in such a way that for each user in the revoked set, there is one share he cannot obtain, so he cannot decrypt the ciphertext, and no two users can combine their shares.

The first scheme is dynamic and has a public key that consists of four group elements from $\mathbb{G}$ and an element from $\mathbb{G}_T$, a master secret key that consists of two elements from $\mathbb{Z}_p$, user secret keys that consists of three group elements, and ciphertexts that consist of an element from $\mathbb{G}_T$ and $2|R| + 1$ elements from $\mathbb{G}$. It is selectively IND-CPA secure under a new q-type assumption, which is shown to hold in the generic bilinear group model.

The second scheme is not dynamic, its public key consists of 11 group elements from $\mathbb{G}$ and one element from $\mathbb{G}_T$ plus a description of the total number of users in the system, the master secret key consists of six group elements and the public key, and user secret keys consist of eight group elements. Ciphertexts consist of one element from $\mathbb{G}_T$ and $2|R| + 7$ elements from $\mathbb{G}$. The scheme is adaptively IND-CPA secure (where the adversary can corrupt users before seeing the challenge ciphertext) under the DBDH and DLIN assumptions, and the reduction to DLIN loses a factor of $N \cdot |R|$.

3.3.2 Lattice-based Broadcast Encryption

In 2002, Boneh and Silverberg [BS02] proposed to base BE on multilinear maps, which were not known to exist at that time.

To broadcast to N users, the scheme relies on symmetric N-multilinear maps. The encryption key consists of the group generator, a random seed of length $\log^2 \kappa$, where $\kappa > N$ is the security parameter, and a number in $[1, \mathrm{ord}(G) - 1]$. a user secret key consists of two group elements and the same random seed. The scheme does not need to transmit a key header, the description of the target set is sufficient. The scheme is IND-CPA, where the adversary cannot corrupt users before the challenge phase, but automatically corrupts all users not in the target set in phase 2, under the the Diffie-Hellman inversion assumption (given g, g^b, it is hard to compute $e(g, \ldots, g)^{1/b}$) in the ROM.

Recently, an analogue of multilinear maps from lattices has been proposed, based on ideal lattices [GGH13] and over the integers [CLT13]. The proposals differ from actual multilinear maps ([BS02, def. 2.2] for the symmetric case) in several aspects: The map is not deterministic but *noisy*, and the groups have an additional structure, they are *graded*. For a regular l-linear map, we would expect to have l groups of the same order and a map that maps a vector consisting of one element from each of the l groups to an element from a target group. In a l-graded encoding scheme for a ring R, we let S_i^α be the set of encodings of a value $\alpha \in R$ at level i. This set corresponds to an element g^α in a cyclic group of a regular multilinear map, and α has several encodings due to the noise inherent in the construction. For correctness, it is important that the encoding is unique, i. e. all S_i^α are disjoint. As usual, we can add and subtract elements on the same level without increasing the level: For $u \in S_i^\alpha, v \in S_i^\beta$, we have $u + v \in S_i^{\alpha+\beta}, u - v \in S_i^{\alpha-\beta}$. The evaluation of the multilinear map corresponds to a multiplication of elements, which also results in an increase of the level, but can be done stepwise: For $u \in S_i^\alpha, v \in S_j^\beta$, we have $uv \in S_{i+j}^{\alpha\beta}$, with the restriction that $i + j \leq l$. This definition models a symmetric map, where all encodings are in the same group. It can be generalized to asymmetric maps by letting the indices be vectors of integers instead of integers, and requiring that all components of the vector are at most l. Analogues of hardness assumptions can be defined in this setting, but it should be noted that the

entity constructing the multilinear map needs to choose a secret element that makes it easy to break these hardness assumptions.

The scheme from [BGW05] can be extended [Bon13] using $(\log N)$-linear maps achieves ciphertexts of size $\mathcal{O}(\log N)$, while encryption and decryption keys have size $\mathcal{O}(\log^2 N)$, and the secret part of the decryption key is $\mathcal{O}(\log N)$.

3.4 Variants of Broadcast Encryption

We treat dynamic broadcast encryption, where the number of users does not have to be fixed at setup, in section 3.4.2. Section 3.4.3 covers identity-based broadcast encryption, where the set of user identifiers can be exponential in the security parameter.

3.4.1 Forward-Secure Broadcast Encryption

Forward secure BE schemes guarantee the secrecy of messages encrypted to a user, even if his decryption key is compromised later. This is accomplished by periodically updating user decryption keys, while leaving the encryption key unchanged. The maximum number of time periods is denoted by S.

Yao, Fazio, Dodis, and Lysyanskaya [YFDL04] used a forward secure HIBE to construct a forward secure BE scheme. They use the same subset-difference based construction as [DF03a], and give the length of user secret keys as $\mathcal{O}(\log^3 N \log T)$, of the ciphertext as $\mathcal{O}(r \log N \log T)$ and of the encryption key as $\mathcal{O}(r \log N + \log T)$ The authors claim that their construction is generalized IND-ACCA secure and can be made IND-ACCA secure using the approach in [DK05].

Attrapadung, Furukawa, and Imai [AFI06] introduced a new primitive called *hierarchical identity-coupling broadcast encryption* (HICBE), which combines HIBE and BE. From this, they construct BE with keyword-search, and two forward secure BE schemes. The schemes require pairings on prime-order groups, but can work with both symmetric and asymmetric pairings. Both schemes have public keys of length $\mathcal{O}(N + \log T)$ and user secret keys of length $\mathcal{O}(\log^2 T)$. The first scheme has ciphertexts of size $\mathcal{O}(\log T)$, while the second one achieves constant-size ciphertexts. Both schemes are selectively IND-CPA secure (the adversary must commit to the target set before the setup) under the N-BDHE assumption (sec. 2.3.2) in the standard model.

3.4.2 Dynamic Broadcast Encryption

Dynamic broadcast encryption was introduced by Delerablée, Paillier, and Pointcheval in [DPP07]. According to their definition, a BE scheme is dynamic, if

1. The setup is independent of the expected number of users.
2. The ciphertext length is independent of the number of users.
3. The decryption keys of users in the system are unaffected by the joining of new users.
4. The public key is modified only incrementally (the complexity of the process is constant) when a new user joins.

DPP The first dynamic broadcast encryption scheme was proposed by Delerablée, Paillier, and Pointcheval in [DPP07]. The construction requires a pairing in prime-order groups, but it does not matter whether the pairing is symmetric or not. Revocation works by assigning a value x_u to each user u and forcing him to multiply a group element by $\frac{1}{x_r - x_u}$ for each revoked user r. This means that every revoked user has to divide by 0 during decryption, which lets the procedure fail.

The scheme is statically IND-CPA secure and (t, N)-collusion resistant in the standard model under the (t, N)-GBDHEassumption (def. 2.16), where the reduction is tight. The security guarantee is actually stronger than static security: The adversary is allowed to corrupt users immediately before they join. This notion will be explored in chapter 4. The master secret key and the user secret keys consist of a scalar value and one group element from each of the base groups $\mathbb{G}_1, \mathbb{G}_2$, the encryption key consists of one group element from each of the base groups and one from the target group $\mathbb{G}_T$. For each user, a scalar, an element from the base group $\mathbb{G}_2$ and an element from the target group is added to the encryption key. The ciphertext consists of one element from each of the base groups plus a scalar and an element from $\mathbb{G}_2$ for each revoked user.

If the encryption key is kept secret, its size can be reduced to four group elements and a scalar, and the description of a hash function. Another modification of the public-key scheme makes it non-dynamic, but allows achieving constant-size ciphertexts at the expense of linear size decryption keys.

Scheme	|KH|	|DK|	|EK|	Security	Assumption
Naive	$\mathcal{O}(s)$	$\mathcal{O}(1)$	$\mathcal{O}(N)$	IND-SCPA	OWF
BGW_1	$\mathcal{O}(1)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	IND-SCPA	N-BDHE
BGW_1'	$\mathcal{O}(s)$	$\mathcal{O}(1)$	$\mathcal{O}(N)$	IND-SCPA	N-BDHE
BGW_2	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(\sqrt{N})$	IND-SCPA	$\sqrt{N}$-BDHE
BGW_2'	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(1)$	$\mathcal{O}(\sqrt{N})$	IND-SCPA	$\sqrt{N}$-BDHE
BGW_3	$\mathcal{O}(1)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	IND-SCCA	$N+1$-BDHE
GW	$\mathcal{O}(s)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	ad1 IND-CPA	N-BDHE
GW	$\mathcal{O}(1)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	ad1 IND-CPA	N-BDHE, ROM
BW	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(\sqrt{N})$	ad1 IND-CPA	$\sqrt{N}$-BDHE
LSW	$\mathcal{O}(r)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	ad1 IND-CPA	DBDH, DLIN
DPP	$\mathcal{O}(r)$	$\mathcal{O}(1)$	$\mathcal{O}(N)$	dyn IND-SCPA	GBDHE
DPP	$\mathcal{O}(1)$	$\mathcal{O}(N)$	$\mathcal{O}(N)$	IND-SCPA	GBDHE

Table 3.2: Comparison of public-key BE schemes

3.4.3 Identity-based Broadcast Encryption

Identity-based broadcast encryption (IBBE) extends BE in the same way that identity-based encryption (sec. 2.5.4) extends PKE. Delerablée proposed the first ID-based BE scheme in [Del08]. The construction is based on pairings in prime-order groups, but it does not matter whether the pairing is asymmetric or not. The construction has constant-size decryption keys and ciphertexts and the public key is linear in the maximal size of the target set. The construction achieves a variant of adaptive IND-CPA security (where users can be corrupted only before the challenge phase and only when they join) under the GBDHE assumption in the random oracle model.

Sakai and Kasahara proposed an IBBE scheme in 2007 without a security proof, which was given in the same year by Sakai and Furukawa [SF07].The scheme also has constant-size decryption keys and ciphertexts, and the public key is linear in the maximal size of the target set. The scheme is fully adaptive IND-CPA secure (where the adversary can corrupt users before and after the challenge phase) in the generic bilinear group model with random oracles.

Gentry and Waters propose an IBBE scheme in [GW09]. Their scheme has ciphertexts of size $\mathcal{O}(\sqrt{|S|})$ and decryption keys of size $\mathcal{O}(\sqrt{\max|S|})$ Their scheme has adaptive IND-CPA security (where the adversary can corrupt users before the challenge phase) under an assumption that is related to BDHE. The scheme achieves constant-size ciphertexts in the ROM.

Attrapadung and Libert [AL12] construct the first IBBE with constant-size ciphertext that is adaptively secure in the standard model under the DBDH and DLIN assumption. The master secret key is also of constant size, but the encryption key and user secret keys are linear in N. They also present an ID-based revocation scheme with constant-size ciphertext that is semi-statically secure under the DBDH and DLIN assumptions in the standard model. Again, the MSK is constant-sized while EK and the usk's are linear in N.

3.5 The State of the Art in Broadcast Encryption

In this chapter, we describe the current state of the art in tree-based broadcast encryption. LSD-TOWP [Asa04] is an enhancement of the LSD scheme which uses a trapdoor one-way permutation (TOWP) to reduce the length of the user decryption key by $\log N$ while keeping the length of the key header linear in the number of revoked users. GR-CS [GR04] is a variant of the CS method by Gentry and Ramzan which has constant-size user decryption keys. To conclude, we give concrete values for the transmission, storage, and computational complexity of both schemes assuming a user set of size 2^{30}.

3.5.1 Layered Subset Difference with TOWP

The length of the decryption key in the basic LSD scheme [NNL01] is reduced by Asano using TOWP [Asa04]. We start from a description of the less complicated SD-TOWP and add the layers later.

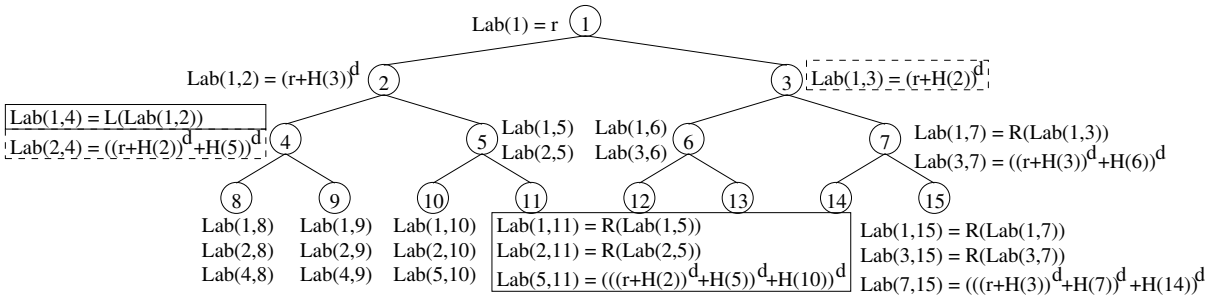

Figure 3.7: User 3 (at leaf 10) receives the boxed labels

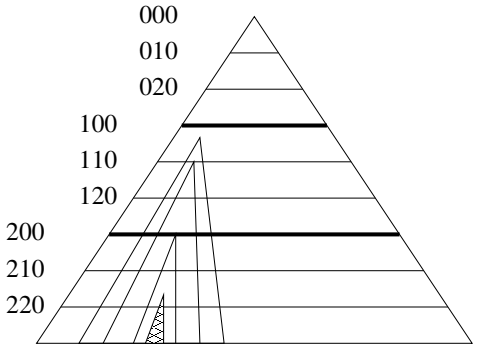

Figure 3.8: In LSD, levels ending in 00 are more special than levels ending in 0.

SD-TOWP

The centre first defines a binary tree with N leaves, where N is a power of 2. All nodes are numbered from 1 to $2N-1$ and the N users u_i are assigned to the leaves.

Let $S_{i,j}$ be the subtree rooted at i minus the subtree rooted at j, as for the SD method. When i is the parent of j, the subset is called *special*. Each user is a member of $\log N$ special subsets. Let S_{full} be the full tree.

The centre chooses a RSA key pair $ek = (e, M), dk = (d, \varphi(M))$, a symmetric encryption algorithm with key length λ, and two independent hash functions $H_1 : \mathbb{Z}_M^\times \to \{0,1\}^\lambda$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_M^\times$. Then he assigns a random value $x_1 \in Z_M^\times$ to the root, and generates a binary tree by setting $x_i = (x_{par(i)} + H_2(i))^d \mod M$. ($par(i)$ returns the parent $\lfloor \frac{i}{2} \rfloor$ of a node.) The keys are then generated as $Lab_{full} = H_1(x_1)$, $Lab_{j,2j} = H_1(x_{2j+1})$, and $Lab_{j,2j+1} = H_1(x_{2j})$. Switching the siblings allows us to take the values that are derived from each other and place them in the tree so that they hang off the same path to the root. Because a user should know all the keys that hang off his path to the root, he will be able to compute all special labels from the lowest one.

Using a 3-expanding PRNG $G : \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$, the centre generates the rest of the labels $Lab_{i,j}$ as in the original SD method. For convenience, we name the right third of G's output R, and the left third L.

The centre gives to each user i the non-special labels hanging off his path to the root, as in the original SD scheme, the key corresponding to the full tree, and the value $x_{sib(i)}$ (the lowest special label that was assigned to his sibling). This is depicted in an example for 8 users in figure 3.7. From the figure we can see how the special labels in the dashed boxes can be derived from the label $Lab_{5,11}$ by exponentiating with the public e and adding the hash values of the publicly known node numbers.

LSD-TOWP

Now we add the layers. We start with the basic LSD scheme, and designate every level that is a multiple of $\sqrt{\log N}$ as a "special level". Then we give to the users only those keys belonging to special subsets. A subset$S_{i,j}$ is special if i is in a special layer ($i = l\sqrt{\log N}$ for any integer l) or if i and j are in the same layer (they are "sandwiched" between the same special levels, so $i \div \sqrt{\log N} = j \div \sqrt{\log N}$ where $\div$ is the operator that returns the integer part of a division). Any subset from the SD method can be written as the union of two special subsets.

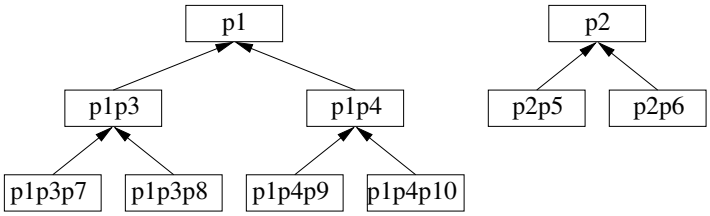

Figure 3.9: The e-values for a simple poset. Arrows denote the direction of key derivation.

General LSD Let the number of levels in the tree be l. We give each level a number with d digits in base $\sqrt[d]{\log N}$. The root is in level $0 \dots 00$, the next level is $0 \dots 01$, and so on until the last level $(d-1) \dots (d-1)(d-1)$. Levels that end in k 0's are called k-special. We illustrate this for $d = 3$ and $N = 2^{27}$ in figure 3.8. Users are given keys for those subsets where j is the next-more special node from i, or where i is more special than j.

Remark 3.2. *Asano designated those labels $Lab_{i,j}$ as special for which i is the parent of j. They are special in the sense that the set $S_{i,j}$ consists of a single tree. This construction does not work when i is not the parent of j, because the set $S_{i,j}$ consists of disconnected subtrees. Therefore, some users will try to derive the label going "downward" from another label they have received using the pseudorandom function. This means we cannot replace this derivation using another one based on a TOWP.*

3.5.2 Accumulator-based

Gentry and Ramzan separate the problem of constructing a BE scheme in the subset cover framework into the combinatorial problem of constructing a subset-cover, and the computational problem of deriving keys for the subsets from a single master key. The subset-cover problem is solved by introducing layers, similar to the LSD method, while the key derivation is done with the accumulator-based method of Akl and Taylor [AT83].

Constructing a Subset-Cover

In the SC framework, there are a number of subsets $S_i, i \in I$ such that any subset of the user set U can be represented as a union of S_i's. The subset-cover problem is to find for a target set S a set $J \subset I$ such that $T = \bigcup_{i \in J} S_i$ and $|J|$ is minimal.

First, we set up a binary tree and choose a depth b. Then we stratify the tree into blocks containing b levels and set $a = 2^b$. This a is the parameter that regulates a trade-off between the length of the key header and the number of exponentiations the receiver hast to perform. Then we generate keys $k_{i,j}$ for all subset differences of nodes within the same block (including the borders). Unlike the LSD method, this way of layering aims to keep the distance between i and j short (at most b). A user u is then given all $k_{i,j}$ for which i is an ancestor of u, but j is not. Each user receives at most $(2a - \log a - 2) \log_a N$ keys.

This is similar to the LSD method, but while LSD minimizes the length of the user secret key, the goal of GR is to minimize the number of keys that can be derived from a user secret key, which minimizes the computational cost at the receiver.

Key Derivation

Given a partially ordered set (poset) $(S_i, \subseteq)$, Akl and Taylor use RSA to generate a corresponding set of keys k_i such that a key k_j can be derived from another key k_i ($k_i \to k_j$) exactly if $S_i \subseteq S_j$. This is to enable single users to compute the keys to the groups they are a member of from their keys.

To implement this, choose an RSA modulus M, a master key $K \in \mathbb{Z}_M^\times$, and a prime p_i for each S_i. For each S_i, calculate exponent $e_i = \prod_{k_i \to k_j} p_j$. The key of a node will then be $k_i = K^{\frac{1}{e_i}}$, so that k_j with $k_i \to k_j$ can be computed from k_i by raising k_i to the power of the primes associated with the nodes between them. The following procedure is equivalent. First choose all largest subsets S_j in the sense that $S_j \subseteq S_i \Rightarrow S_j = S_i$. These largest S_j will be the roots of the forest implied by the partial order, because their keys do not imply any other keys. Set $k_j = K^{1/p_j}$. Then for all neighbors S_i of such an S_j with $k_i \to k_j$, compute $k_i = k_j^{1/p_i} = K^{\frac{1}{p_i p_j}}$. This makes clear that k_j can be computed from k_i as $k_j = k_i^{p_j}$. A simple example is depicted in figure 3.9.

The cost to compute a key from another depends on the number of nodes on the way between them. In the worst case, a modular exponentiation has do be computed for each inverted prime in the exponent of a key to reach one of the roots.

3.5.3 Concrete Efficiency

Now that we have described both schemes and their asymptotical complexity, we want to evaluate their concrete performance assuming a user set of size 2^{30} and a security level of 128 bits. This corresponds to an RSA key length of 3248 bits [ECR12, tab. 7.2], which we round to 3072.

Practical performance of LSD-TOWP We first construct the plain LSD scheme. The basic LSD scheme would require each users to store $2\log^{3/2} N = 2 \cdot 30^{3/2} = 329$ labels. If we use the generalized version with 3 degrees of specialness, we need a layer depth of $\lceil \sqrt[3]{30} \rceil = 4$. For the full tree with $4^3 = 64$ levels, we expect the length of the user secret key to be in the order of $3 \cdot 30^{4/3} = 280$ labels. If we increased the number of degrees of specialness to 4, we would need $4 \cdot 30^{5/4} = 280$ labels for the full tree with $3^4 = 81$ levels, but with an increased complexity.

So we choose 3 degrees with layer depth 4, and number the levels of the tree in 4-ary, starting with 000, 001, 002, 003, 010, ..., so the last level will be number $132 = 1 \cdot 16 + 3 \cdot 4 + 2 = 30$. The only 2-special levels are 000 and 100, and the six 1-special levels are 010, 020, 030, 110, 120, and 130. For our tree with 30 levels, a users secret key comprises 139 keys. The reason for the difference to the expected result is the less than optimal utilization of the parameters ($\sqrt[3]{30} \sim 3.1$, and the tree could support 64 levels instead of the 30 we use). This means that for a key length of 128 bits (which is more than $2\log N$), a user secret key will have 17792 bits, or about 2.2 kB. If we apply the TOWP construction, we will save 30 labels, but we will have to use labels in $\mathbb{Z}_M^\times$, where M is a RSA modulus. A RSA modulus of corresponding security level will have 3072 bits, so our user decryption key will be longer by a factor of 19.

The key header will have a length of $6r$ encryptions of the session key for r revoked users, but as described in [HS02], the average number will be lower. In experiments with the basic scheme, instead of $4r - 2$, the scheme used only $2r$ keys, so we can reasonably hope for a value around $3r$ encryptions, or $384r$ bits. The id header needs $2\log 2N = 62$ bits per encryption to identify the sets $S_{i,j}$. Then the total communication complexity will be $446r$ bits for both headers.

The computational complexity at the receiver is determined by the number of calls to the PRNG he has to make to derive the key. In the worst case, he has to derive the label $Lab_{1,j}$ where j is a leaf from the label $Lab_{1,2}$ or $Lab_{1,3}$, which takes $\log N = 30$ calls.

We now detail the subsets for which a user receives labels. First we color the leaf of the user and all nodes on the path to the root grey. The user will receive labels $Lab_{i,j}$ for all nodes j that have a grey parent. There are five kinds of subsets $S_{i,j}$ for which the users will receive labels:

1. When j is 1-more special than i
   (a) When i is not special and j is 1-special ($i \in xyz, j \in x(y+1)0$, with $z \neq 0$)
   (b) When i is 1-special and j is 2-special ($i \in 0y0, j \in 100$, with $y \neq 0$)
2. When j is less special than i
   (a) When i is not special and j are in the same layer ($i \in xyz, j \in xyz'$ with $z \neq 0, z' > z$)
   (b) When i is 1-special, and j is in the same 1-layer ($i = xy0, j = xy'z$ with $y \neq 0, y' \geq y$)
   (c) When i is 2-special, and j is in the same 2-layer ($i = x00, j = x'yz$ with $x' \geq x$).

Let us now compute the exact number of labels for each type of subset.

1. There are 24 subsets of this kind.
   (a) In each of the seven complete layers, there are 3 such subsets, so 21 in total.
   (b) There are three such subsets.
2. There are 105 subsets of this kind.
   (a) There are eight layers: 00z, 01z, 02z, 03z, 10z, 11z, 12z, and 13z. In each of the seven complete layers there are the following three subsets: $i \in 1, j \in 2, 3$, and $i \in 2, j \in 3$. In the last layer there is only one subset: $i \in 131, j \in 132$. So in total there are $7 \cdot 3 + 1 = 22$ subsets in this category.

(b) For $i \in 0y0$ there are $y \cdot 4 - 1$ subsets $0y'z$. so $11 + 7 + 3 = 21$ in total For $i \in 1y0$ there are $y \cdot 4 - 2$ subsets $1y'z$. So in this category there are $11 + 7 + 3 + 10 + 6 + 2 = 39$ subsets in total.

(c) For $i \in 000$ there are 30 subsets. For $i \in 100$ there are 14 subsets. So there are 44 in total.

This gives a total of 139 labels.

Practical performance of GR04 The user secret key of this scheme is an element in $\mathbb{Z}_M^\times$, where M is a RSA modulus, which will have 3072 bits at the 128-bit security level, which beats the LSD scheme by a factor of 5.8. However, the users have to perform exponentiations $\bmod M$ instead of calls to a PRNG to derive the key and need to store large primes as additional information. In addition, the transmission complexity is higher, and there is a tradeoff between the transmission complexity and the computational complexity at the receiver that is a parameter of the system.

The length of the key header is $r(\frac{\log N - \log r}{\log a} + 1) \log M = r\frac{30-(\log r - \log a)}{\log a} 3072$ bits. We choose $\log a = 5$, which gives us $(7 - \frac{\log r}{5})3072r$ bits, or $(21504 - 614.4 \log r)r$ bits for the key header length, which is $56 - 1.6 \cdot \log r$ times of what was required for LSD. We obtain $(2a - \log a - 1)\frac{\log N}{\log a} = (64 - 5 - 1)\frac{30}{5} = 348$ exponentiations as the receivers computational cost.

Practical performance of BGW Since we assume a security level of 128 bits (and we assume for simplicity that a 3072-bit RSA modulus will have the same security level), then as discussed in sec. 2.3.4, we can assume that elements in $\mathbb{G}$ have 256-bit representations and elements in $\mathbb{G}_T$ have 3072-bit representations if we relied on the DDH assumption. Since we use a q-type assumption, for $q = 2^{30}$, we will have to increase the size of $\mathbb{G}$ by 30 bits, leaving us with $12 \cdot 286 = 3432$ bits as the size of $\mathbb{G}_T$-elements.

For the BGW_1-construction, this gives us a key header of $2 \cdot 286 = 572$ bits, but a decryption key size of $(2^{31} + 1)286\text{bits} \approx 71.5$ GB.

The BGW_2-scheme only relies on the $\sqrt{N}$-BDHE assumption, so we can have 271-bit $\mathbb{G}$-elements and 3252-bit $\mathbb{G}_T$-elements. A ciphertext will be $2^{16} \cdot 271$ bits, or 2.1 MB, with key sizes about the same.

Practical performance of BW It is interesting to compare this to the [BW06] scheme, which uses composite order groups and no q-type assumption. In this setting, elements of $\mathbb{G}$ have 3072 bits, and elements of $\mathbb{G}_T$ have 6144 bits. A ciphertext consists of 2^{15} elements of $\mathbb{G}_T$ and $6 \cdot 2^{15}$ elements of G for a total size of $6 \cdot 2^{25} + 18 \cdot 2^{25} = 3 \cdot 2^{28}$ bits or 96 MB, 46 times larger than for the BGW_2-scheme.

Practical performance of GW To find out how much we pay for adaptive security, we turn to the [GW09] scheme, which uses prime order groups and the $2N$-BDHE assumption. In this setting, elements of $\mathbb{G}$ have 287 bits, and elements of $\mathbb{G}_T$ have 3444 bits. A ciphertext has 4 group elements plus $|S|$ bits, or at most $2^{30} + 4 \cdot 287$ bits, and we see that the tag of 128 MB clearly dominates the ciphertext length. For small S, the scheme can be very efficient.

Practical performance of DPP The security of DPP is based on a q-type assumption: (t, N)-GBDHE for (t, N)-collusion resistance. We have $q \approx N + t$, so we increase the size of $\mathbb{G}$-elements to 287 bits, resulting in 3444-bit $\mathbb{G}_T$ elements. Key header length is $2 \cdot 287 + (287 + 3444)r = 574 + 3731r$ bits, or 0.455 kB per revoked user. This is more efficient than a BGW_2-key header as long as at most 4760 ($\approx 2^{12}$) out of 2^{30} users are revoked.

3.6 Broadcast Encryption Schemes with Advanced Functionalities

In section 3.6.1, we present anonymous broadcast encryption schemes, which hide not only the content, but also the target set. We also take a quick look at broadcast attribute-based encryption in section 3.6.2.

3.6.1 Anonymous Broadcast Encryption

In some settings, even the information which users are in the target set of a broadcasted message can be sensitive. Anonymous broadcast encryption (sec. 2.6.2) aims to hide the set of receivers of a ciphertext as well as the message. This means that the division of the ciphertext into target set header and key header no longer applies, since the information about the target set is implicitly contained in the key header.

Barth, Boneh, and Waters [BBW06] propose a generic construction of an anonymous (called private in this paper) broadcast encryption scheme based on an IND-CCA secure key private PKE scheme. Because the PKE ciphertexts are tied together with a one-time signature, the construction achieves anonymity under selective corruption (the two target sets S_0, S_1 of equal size are fixed before setup, and all users in $S_0 \cap S_1$ are corrupted) and chosen-ciphertext attacks, but confidentiality is neither defined nor proved. Efficiency is as for the naive BE scheme. They also give a way to speed up decryption in the ROM by avoiding trial decryptions. Each user secret key is extended with a random scalar u, and the corresponding group element g^u is added to the encryption key. The broadcaster draws a random scalar r, prepends g^r to the ciphertext, and labels each ciphertext component encrypted to (ek, g^u) with $H(g^{ur})$. A decrypting user computes $H(g^{ur})$ and only attempts to decrypt the component labeled with this value.

Krzywiecki, Kubiak, and Kutyłowski [KKK06] and Krzywiecki, Kutyłowski and Nikodem [KKN08] construct an anonymous BE scheme based on polynomial interpolation. The size of the user secret keys and the ciphertext is linear in the maximal number of revoked users. No formal security definition or analysis is given.

Krzywiecki and Kutyłowski [KK11] construct another anonymous BE scheme on the assumption that a physically unclonable function (PUF) instantiates a random oracle, but do not provide a formal security analysis.

Libert, Paterson, and Quaglia [LPQ12] provide a security definition for ANOBE that combines anonymity and confidentiality, and where the two challenge target sets may be required to be of equal size. They then show that the scheme from [BBW06] has adaptive CCA security (anonymity and confidentiality) in the standard model even when using the tag-based hint system to speed up decryption and remark that the PKE scheme used in the construction in [BBW06] needs to be weakly robust as well as key-private. They proceed by giving two concrete constructions, one based on a hybrid encryption scheme, the other from a multi-trusted authority IBE scheme.

Fazio and Perera [FP12] define *outsider-anonymity*, where anonymity of the receivers is guaranteed only against parties that cannot decrypt a ciphertext. This relaxation allows them to construct a scheme based on anonymous IBE and the construction from [DF03a] of CS-based BE from IBE. The scheme is as efficient as all CS constructions (a decryption key of $\mathcal{O}(\log N)$ group elements and a ciphertext of $\mathcal{O}(r \log \frac{N}{r})$ group elements), and achieves adaptive CCA security (outsider anonymity and confidentiality).

3.6.2 Attribute-based Broadcast Encryption

Attribute-based broadcast encryption schemes (ABBE) combine attribute-based encryption (sec. 2.5.5) and broadcast encryption in such a way that a user can decrypt a ciphertext if his attributes fulfill some policy and he is not in the set of revoked users.

Lubicz and Sirvent [LS08] combine attribute-based encryption with the CS method to obtain a public-key broadcast encryption scheme that allows efficient selection of certain user groups using attributes. The authors propose to assign users to the leaves of a binary tree, and a new attribute to each node in the tree. Then the subsets from the SD method can be realized by setting $S_{i,j}$ as the policy $(i \wedge \neg j)$. To increase efficiency, additional attributes can be defined to address users based on "natural" groups they form. To define the policy, AND and NOT can be used. OR can be achieved by concatenation.

The construction is based on symmetric pairings on prime-order groups. The size of a user's decryption key is linear in the number of attributes associated with it, while the size of a ciphertext is linear in the number of attributes used in the access policy (which can be upper-bounded by $\mathcal{O}(r)$ if each user has a unique attribute and the subset-difference technique is used). The size of the encryption key is linear in the total number of attributes used in the system, which will be greater than N if the recommendation of assigning a unique attribute to each user is followed. The scheme is proven selectively IND-CPA secure (the adversary must choose the target set before setup) under an extension of the GDHE assumption, which is shown to hold in the GGM.

Attrapadung and Imai [AI09] propose several Broadcast KP-ABE and Broadcast CP-ABE schemes. They combine the KP-ABE from [GPSW06] and the CP-ABE from [Wat11] with the BE schemes from [BGW05] and [LSW10], resulting in four different ABBE schemes. The performance of the schemes is comparable to the underlying ABE schemes, with the BE part either increasing the size of the encryption key or the size of the ciphertext. The schemes are shown to be selectively IND-CPA secure (the adversary

has to commit to the target set before setup) under the strongest assumption of the ones underlying the constituent schemes.

3.7 Predicate Encryption

Broadcast encryption functionality can be obtained from encryption schemes that provide a more powerful functionality. *Predicate encryption* [BSW11] is a generalization of public-key encryption where the decryption algorithm evaluates a (polynomial-time computable) predicate P of the user secret key k and some index y that is part of the ciphertext, and a ciphertext associated to an index y can only be decrypted by keys k for which $P(k, y) = 1$. In predicate encryption even a user that decrypts successfully learns nothing about the index y associated to a decrypted ciphertext, except that his key k, $P(k, y) = 1$. Predicate encryption schemes that reveal the index to a user that successfully decrypts are called *predicate encryption with public index*.

One form of predicate encryption that admits transmitting to several users at the same time is wildcarded IBE (WIBE) [ACD+06, ABC+11], a variant of HIBE (sec. 2.5.4), where the identity vector y to which a ciphertext is encrypted can contain wildcards $\star$ and the predicate P is defined by $P(x, y) = 1$ iff $|x| \leq |y|$ and for $i = 1, \ldots, |x| : (x_i = y_i) \vee y_i = \star$.

In inner-product encryption (IPE), user keys and ciphertexts are associated to vectors x and y, and the predicate is orthogonality: $P(x, y) = 1 \Leftrightarrow x \cdot y = 0$, where $\cdot$ denotes the inner product. Decryption of a ciphertext associated to a vector y is only successful if the secret key used to decrypt is associated to a vector x such that $x \cdot y = 0$. From an IPE scheme, an identity-based broadcast encryption (IBBE) scheme can be constructed. In [AL12], Attrapadung and Libert proposed the first IPE scheme that has constant-size ciphertexts (where the description of the vector is not considered part of the ciphertext), requires only a constant number of pairing evaluations, and is adaptively IND-CPA secure in the standard model under the DLIN and DBDH assumption.

Non-zero inner-product encryption (NIPE) is the complement to IPE, where decryption of a ciphertext associated to a vector y is only successful if the secret key used to decrypt is associated to a vector x such that $x \cdot y \neq 0$. From a NIPE scheme, an identity-based revocation (IBR) scheme can be constructed. In [OT11], Okamoto and Takashima proposed the first NIPE scheme that has constant-size ciphertexts or constant-size secret keys (where the description of the vector is not considered part of the ciphertext or the keys) and is adaptively IND-CPA secure in the standard model under the DLIN assumption. They also construct an IPE scheme with constant-size ciphertexts that is adaptively IND-CPA secure under only the DLIN assumption.

Attrapadung et al. [AHL+12] present several selectively secure constructions with constant-size ciphertexts, at the expense of having large (quadratic) private keys. They give a generic transformation of IBBE schemes with very special properties (the scheme must be in a prime-order group with a symmetric bilinear map and have a linearity property of the private keys and the ciphertext) to KP-ABE schemes with monotonic access structures, and a specific construction with non-monotonic access structures from an identity-based revocation scheme.

3.8 Combinatorial Traitor Tracing Schemes

Combinatorial TT schemes rely on the assignment of a subset of symmetric keys to users, so that when a pirate decoder is found, traitors can be traced by analyzing the keys found in the decoder. Almost all combinatorial TT schemes are tree-based.

The first traitor tracing scheme was combinatorial: TT was introduced by Chor, Fiat, and Naor [CFN94], who constructed a t-resilient TT scheme based on symmetric encryption. To obtain a 1-resilient TT scheme for user identifiers id of length n, they generate $2n$ secret keys $\{s_i^b\}_{i \in [n], b \in \{0,1\}}$ and assign each user id the keys $s_i^{\mathsf{id}[i]}$. To send a key K, they do an n-out-of-n secret sharing of K and encrypt the i-th share with both s_i^b. This allows white-box tracing of a single traitor from the keys in the decoder. Based on this, the authors propose several constructions, which are later extended in Chor, Fiat, Naor, and Pinkas [CFNP00], based on the use of hash functions to assign keys to users. The construct "one-level" schemes, where a hash function maps the users to their keys, and "two-level" schemes, where a hash function first maps users to a set of size t (for a t-resilient scheme), and a second hash function assigns the keys. They also distinguish between "open" schemes, where the hash functions are public, "secret" schemes, where they are kept secret, and "threshold" schemes, which require that a pirate decoder decrypt with a probability greater than a threshold fixed in advance.

The subset cover (SC) framework presented in [NNL01] is a framework for broadcast encryption.

input : A partition P
output: $\emptyset$ if P is disabling or a subset S_i containing a traitor
if $\Pr[D(Enc(EK, P, M)) = M] < p$ **then**
    return $\emptyset$
else `// binary search`
    $f := 0$;
    $l := |P|$;
    while $l - f > 1$ **do**
        $m := \lfloor \frac{f+l}{2} \rfloor$;
        if $p_f - p_m > p_m - p_l$ **then**
            $l := m$
        else
            $f := m$
        end
    end
    return S_l
end

Algorithm 1: SubTrace

The authors also describe a tracing algorithm with relaxed requirements. Given a resettable black-box decoder that decrypts messages sent to all users with probability at least p, the tracing procedure either outputs the identity of one of the traitors, or a subset partition that can be used to broadcast to legitimate users but does not allow the box to decrypt with probability greater than p.

A prerequisite that a scheme has to fulfill to be traceable in this way is the *bifurcation property*, which is that it is possible to partition any subset into two sets of roughly equal size. The *bifurcation value* is defined as the relative size of the largest subset in such a partition and measures its imbalance. Both methods proposed in the paper have this property. In the complete subtree (CS) method each subset is a subtree that can be split into two subtrees of equal size. This gives a bifurcation value of $\frac{1}{2}$. In the subset difference (SD) method, each subset $S_{i,j}$ is the difference of the two subtrees rooted at nodes i and j. The worst case is when i is a grandparent of j. Then the two resulting subtrees are rooted at j's brother and j's uncle, giving a bifurcation value of $\frac{2}{3}$.

The bifurcation property guarantees the existence of a *Split* procedure that takes as input a set and outputs two subsets of roughly equal size.

We define a subset tracing procedure *SubTrace* that takes as input a partition $P = S_1, \ldots, S_m$ of the user space and outputs either a message that P disables the decoder or a subset S_i that contains a traitor.

1. Test if the box decodes a message to P with probability greater than p. If not, output P as the partition that disables the decoder. If yes, continue and find a subset S_i that contains a traitor.
2. This is done using a binary search. Substitute the keys sent to $S_1 \ldots S_{m/2}$ with a random string of the same length and check if the box decrypts with a significant probability of success. If yes, also substitute the keys sent to $S_{m/2} \ldots S_{3m/4}$, if not substitute only the keys sent to $S_1 \ldots S_{m/4}$. Continue partitioning until a set S_i is found that surely contains a traitor.

To make it easier to describe the subset tracing, we define the probability p_j as the probability of the decoder to decrypt the message after the first j key encryptions have been replaced with encryptions of a random key.

The tracing procedure *Trace* now works as follows. It is initialized with a partition $P = S_1, \ldots, S_m$. *SubTrace*$(S_1, \ldots, S_m)$ returns either a message that P disables the decoder, or a set S_j that contains a traitor. If S_j contains only one user, output this user as the traitor. If not, split S_j into two subsets of roughly equal size, and run *SubTrace* on the new partition $P' = S_1, \ldots, Split(S_j), \ldots, S_m$. The running time of *Trace* is $t \log \frac{N}{t}$, where N is the number of users, and t is the number of keys used in the construction of the pirate decoder.

Parallel Tracing To trace users from more than one pirate decoder, the tracing algorithm is run in parallel on all the decoders, all initialized with the same set. The first box that detects a traitor gives this information to the other tracing procedures so they can exclude him. If a tracing procedure discovers a partition that disables its box, it forwards this partition to all other procedures.

```
input  : A partition P
output: Either a disabling partition or a traitor
while true do
    S_i ⟵ SubTrace(P);
    if S_i == ∅ then
        exit "P is disabling partition"
    else if |S_i| == 1 then
        {u} ≔ S_i;
        exit "u is a traitor"
    else
        (S_i^l, S_i^r) ⟵ Split(S_i);
        P' ≔ P \ S_i;
        P ≔ P' ∪ {S_i^l, S_i^r};
    end
end
```

Algorithm 2: Trace

3.8.1 Improvements

Because the tracing algorithm described requires $t \log \frac{N}{t}$ iterations, and in each iteration the number of subsets in the partition increases by one, tracing t traitors can result in up to $t \log \frac{N}{t}$ subsets, and in messages of length $t \log \frac{N}{t}$. Because the CS method has a message length of $r \log \frac{N}{r}$ for r revoked users, the tracing algorithm can trace up to r traitors. (If we trace more traitors a decoder will know he is being traced, because the message length will be greater than for any regular message.) For the SD method, the message length is only $2r - 1$, which allows less users to be traced. A modification of the tracing algorithm will decrease the number of subsets in a partition to $5t + 1$ and so allow up to $\frac{r}{5}$ users to be traced.

The authors introduce *frontier subsets* that contain those sets where traitors are suspected. The set of frontier subsets F is empty at the beginning, and in each iteration we set $F \coloneqq F \cup \mathit{Split}(S_j)$, where S_j is the subset output by *SubTrace*. Furthermore, if S_j was in the result of an earlier split, $S_j, S'_j \leftarrow \mathit{Split}(S_i)$, they remove the other set from the frontier $F \coloneqq F \setminus S'_j$. Then they compute a cover C of all receivers not in F and define the new partition as $P \coloneqq F \cup C$.

Using the SD method, C can be covered using at most $3t - 1$ subsets, and F contains at most $2t$ subsets, so the number of sets required to cover the receivers is $5t - 1$.

3.8.2 Tracing for Other Schemes in the SC Framework

Tracing works for the CS and SD methods, because the adversary cannot distinguish encryptions of the session key $E_{K_u}(K)$ from encryptions of a random key $E_{K_u}(R)$. The reason for this is that the user secret keys K_u are indistinguishable from random keys for the adversary, and so the session key is encrypted under a pseudorandom key.

There are modifications of the CS and SD methods that are designed to make them more efficient. However, in several cases these modifications result in schemes that are no longer key indistinguishable, but only key intractable. In this case, it is not clear that the indistinguishability of the whole scheme is preserved.

3.8.3 Public-Key Variants

Naor et al. point out that the CS method can be transferred to the public-key world by replacing each secret key with a public/private key pair from a PKE (public-key encryption) scheme. This results in a large public key that consists of $2N - 1$ public keys of the basic PKE scheme. They point out that an IBE (identity-based encryption) scheme would mitigate this problem [NNL01, sec. 4.3]. Dodis and Fazio expand on this idea and describe in detail the use of IBE to construct a public-key CS scheme and of HIBE (hierarchical IBE) to obtain a public-key SD scheme [DF03a]. Tracing is done in the same way as in the secret-key scheme.

3.8.4 Pirates 2.0

Billet and Phan [BP09] pointed out an attack that had not been considered in the security model. Since user secret keys consist of many subkeys, a traitor does not have to contribute his complete key to a pirate decoder. If he only contributes keys that are associated to nodes high up in the tree, then there will be many other users who could have contributed the same keys, and he has a guaranteed level of anonymity. This guaranteed anonymity makes it possible to coordinate the sharing in public and might conceivably lower the inhibition of users to contribute keys to the pirate decoder, since they can be sure that they will not be found out. The only way for the centre to counteract this is to encrypt using only keys that are located at low levels of the tree and which are shared by only a few users, which results in large ciphertexts. The same holds for code-based schemes, where user keys also consist of many subkeys. The authors show that due to the relatively little space of 1 MB assigned to the master key block, this is a realistic attack on the AACS scheme.

D'Arco and Perez del Pozo [DdP11] discussed countermeasures against this attack. By keeping the location of keys in the tree secret, public collaboration threatens anonymity, but private collaboration remains a problem. They therefore proposed combining a vulnerable scheme with other schemes.

3.9 Traitor Tracing from Polynomial Interpolation

Naor and Pinkas [NP01] combine traitor tracing with the *self enforcement* introduced in [DLN96], where a user decryption key contains private information such as his credit card number. Since the only keys that colluding users can produce are linear combination of their keys, the scheme allows tracing in the following forms.

- white-box: Given a key, the identities of up at most $t/2$ colluding traitors can be exposed.
- confirmation: For any subset of at most t users, it can be tested whether their keys were used to construct the pirate decoder.

For t-collusion resilience, a polynomial of degree t must be used. This results in a ciphertext of $t+1$ group elements, a user decryption key of a single group element, and an encryption key of $t+1$ scalars (the description of the polynomial).

The scheme is IND-CPA secure under the DDH assumption in the standard model.

3.10 Code-based Traitor Tracing

The basic idea of code-based TT is to extend a scheme for two users that is resilient against a single traitor to N users by combining l instances of the scheme using a binary code with codewords of length l. This design principle was used by Naccache, Shamir, and Stern [NSS99] in a more generic context. They construct a *copyrighted* symmetric encryption scheme, where users have different decryption keys that can all decrypt normal ciphertexts, but for which special ciphertexts can be constructed that allow tracing.

Gafni, Staddon, and Yin [GSY99] discuss how to add revocation to a tracing scheme or tracing to a revocation scheme. They show that if a user key contains at least $4t \log N$ subkeys, then the scheme can be made t-traceable at the cost of increasing the total number of keys and the ciphertext size by a factor of $2t^2$.

Kiayias and Yung [KY02c] constructed the first black-box TT scheme with constant transmission rate, based on collusion-secure fingerprint codes. They follow the code-based design paradigm, first constructing a copyrighted PKE and extending it to multiple users with a collusion-secure code.

In [KY02b], Kiayias and Yung propose a way to convert a tracing procedure against stateless decoders in a tracing procedure against stateful decoders by using watermarks.

Phan, Safavi-Naini, and Tonien [PSNT06] construct a black-box publicly traceable scheme based on IPP codes. The construction uses a target collision-free hash function, a PKE and a symmetric encryption scheme.

Fazio, Nicolosi, and Phan [FNP07] construct the first black-box traceable scheme with transmission rate 1, based on collusion-secure fingerprinting codes. The scheme uses an all-or-nothing transform (AONT) to force the decoder to decrypt all ciphertext blocks.

Billet and Phan [BP08] constructed a scheme with constant-size ciphertexts based on collusion-secure robust fingerprinting codes that can deal with erasures (sec. 2.7.3). To add δ-robustness (for $\delta \in (0,1)$) to a fingerprinting code, they repeat bits $\frac{1}{1-\delta}$ times. Ciphertexts consist of $2u$ symmetric ciphertexts and u random values that are logarithmic in the length of the used code, where u is a constant that depends on the minimal decryption probability of a pirate decoder and the allowed fraction of deletions.

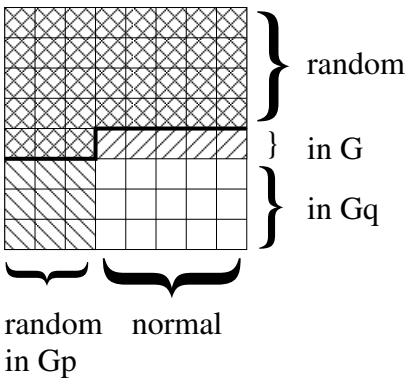

Figure 3.10: Abstract view of the BSW scheme.

Boneh and Naor [BN08] also construct a scheme with constant-size ciphertexts based on collusion-secure fingerprinting codes. They obtain a t-collusion resistant traitor tracing scheme from an IND-CPA secure PKE scheme and a t-collusion resistant fingerprinting code. To trace imperfect decoders that fail to decrypt with some probability, they use robust fingerprinting codes. The ciphertext consists of two PKE ciphertexts.

3.11 Algebraic Constructions of Traitor Tracing Schemes

Kurosawa and Desmedt [KD98] construct a scheme where tracing is white-box, i. e. based on examination of the keys in the pirate decoder. The security of the scheme is based on DDH.

BF Boneh and Franklin [BF99] proposed a public-key TT scheme based on a *linear space tracing* code and the *representation problem* for a group element. The centre chooses a random element y with a known representation $y = \prod h_i^{\alpha_i}$ with respect to the public basis $h_1, \ldots, h_{2t}$. He also constructs a code with N codewords, which are vectors of integers of length $2t$: $\Gamma = \{\gamma_1, \ldots, \gamma_N\} \subset \mathbb{Z}^{2t}$ and assigns a code word to each user. The secret key of a user i is a scalar θ_i such that $d_i = \theta_i \cdot \gamma_i$ is a representation of y w.r.t. the basis $h_1, \ldots, h_{2t}$. If DLOGis hard, then from less than $2t$ secret keys, a traitor can construct only representations that are convex combinations of these secret keys.

This approach allows them to trace in three ways:

- white-box: Given one representation of y used by the pirate decoder, a tracing algorithm can determine one of the keys used to construct this representation in time $\mathcal{O}(N \log N \log \log N)$.
- black-box single-key: If the decoder contains only a single representation of y (which can be derived from up to t secret keys), then black-box tracing is possible. This requires the tracer to be able to compute discrete logs, e. g. because composite-order groups are used and he knows the secret factorization of the group order.
- confirmation: If the tracing authority suspects that the decoder was constructed using the keys of a specific set, it can confirm in a black-box way that the decoder was constructed using a subset of the suspected keys. Since there are $\binom{N}{t}$ possible sets of size t, this does not lead to an efficient black-box tracing algorithm.

The maximal number of traitors t must be set before setup, and must fulfill $N \geq 2t + 2$. The public key consists of a single group element and $2t$ scalars in $\mathbb{Z}_q$, a user decryption key consists of a single group element and a codeword of $2t$ bits, which need not be kept secret. A ciphertext consists of $2t + 1$ group elements. The scheme is IND-CPA secure under the DDH assumption. It can be modified to achieve IND-CCA security under the same assumption at the cost of adding a single group element to the ciphertext, two group elements to the encryption key and four scalars to each secret key.

Kiayias and Yung [KY02a] propose a public-key white-box traceable TT scheme, which is *frameproof* (the group manager cannot construct a pirate decoder that implicates an innocent user). This requires an interactive join-procedure, where the user can influence his key. User decryption keys are constant, while ciphertext and encryption key length is $2t + 2$ group elements.

Chabanne, Phan, and Pointcheval [CPP05] introduce the concept of *public traceability* and achieved transmission rate 1, but did not support black-box tracing.

BSW Boneh, Sahai, and Waters [BSW06] construct a fully collusion-resistant black-box traceable TT scheme from a *private linear broadcast encryption scheme* (PLBE). A PLBE scheme has two defining properties:

- linear broadcast: Users are numbered from 1 to N, and the sender can choose any set $[i, N]$ as the target set.
- private: No user has any information about the target set, besides whether he is in the target set or not.

While encrypting to $[1, N]$ can be done with a public encryption key, encrypting to other target set requires the secret tracing key. This allows a fairly simple tracing procedure based on three observations:

1. When the target set is $[1, N]$, the decoder will decrypt the ciphertext (otherwise it is not useful).
2. When the target set is $[N + 1, N] = \emptyset$, the decoder cannot decrypt the ciphertext (due to the security of the scheme).
3. The decision of the decoder to decrypt can only differ between messages encrypted to $[i, N]$ and $[i + 1, N]$ if he knows the key of user i (because the scheme is private).

Tracing a stateless decoder is then done by simply feeding the pirate decoder with ciphertexts with target sets $[1, N], \ldots, [N + 1, N]$ and observing when he stops decrypting. The running time can be improved by doing a binary search instead.

The procedure can be extended to decoders that decrypt even ciphertexts for $[1, N]$ only with a certain (nonnegligible) probability, by repeating the queries for each target set a sufficient number of times. The traitor is then identified by observing a nonnegligible drop in the decryption probability. The running time of the tracing algorithm is then in $\tilde{\mathcal{O}}(N^2)$.

The construction of the PLBE scheme is based on pairings on composite-order groups. If $\mathbb{G}$ is a group of order pq, with an order p subgroup $\mathbb{G}_p$ and an order q subgroup $\mathbb{G}_q$, then for any $g_p \in \mathbb{G}_p, g_q \in \mathbb{G}_q$, $e(g_p, g_q) = 1$. This cancellation property is used to introduce noise only in a well-controlled fraction of the ciphertexts, as depicted in fig. 3.10. First, the users are ordered in a square and identified by their coordinates (x, y). The ciphertext contains components corresponding to rows and columns in the square. If a row is completely revoked, then the y-component corresponding to this row will be random, preventing users in this column from decrypting. In the one row where some users are in the target set and some are not, the y element will be in the full group $\mathbb{G}$. Then the x-components corresponding to the revoked users will contain random elements in $\mathbb{G}_p$, which will prevent the revoked users from decrypting. To cancel the random noise in the x-columns, the y elements corresponding to rows where all users are in the target set will be in $\mathbb{G}_q$.

The IND-CPAsecurity of the scheme relies on the decision 3-party Diffie-Hellman assumption, the traceability relies in addition to this on the subgroup decision assumption in $\mathbb{G}$, and the bilinear subgroup decision assumption in $\mathbb{G}_T$.

The ciphertext contains $5\sqrt{N}$ elements in $\mathbb{G}$ and $\sqrt{N}$ elements in $\mathbb{G}_T$. User decryption keys are a single element in $\mathbb{G}$ and the public encryption key consists of $3 + 3\sqrt{N}$ elements in $\mathbb{G}$ and $\sqrt{N}$ elements in $\mathbb{G}_T$.

BW Boneh and Waters [BW06] extend this scheme to a publicly traceable trace-and-revoke scheme (TR). They follow the same design, first defining augmented broadcast encryption (aBE), which is broadcast encryption augmented by the privacy property that is necessary for linear tracing: For a target set $\mathcal{S}$ and an index i, the scheme encrypts to $\mathcal{S} \cap [i, N]$, and for any target set $\mathcal{S}$, a user can only distinguish between encryptions to $\mathcal{S}, i$ and $\mathcal{S}, i+1$ if he knows the i-th decryption key. Tracing proceeds in the same way as in the BSW scheme. The security rests on a modified D3DH assumption and the subgroup and bilinear subgroup decision assumptions.

3.12 The State of the Art in Traitor Tracing

In table 3.3 we compare several black-box traitor tracing schemes with respect to the size of their key header, decryption and encryption key, traceability and the hardness assumption on which security is based. Traceability is given as traitor tracing (TT) or trace and revoke (TR), and the method of tracing: the tracing algorithm tracing outputs either a traitor or a configuration to broadcast which disables the pirate decoder, the tracing algorithm can only confirm a suspicion about a user, or the tracing algorithm traces a user using only minimal access to a decoder, where it only learns whether the decoder decrypted a message correctly or not.

Scheme	|KH|	|DK|	|EK|	Traceability	Assumption
[NNL01]	$2r-1$	$\frac{1}{2}\log^2 N + \tilde{\mathcal{O}}(1)$	1	disable TR	PRNG
[NP01]	$t+1$	1	$t+1$	confirm. TR	DDH
[BSW06]	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(1)$	$\mathcal{O}(\sqrt{N})$	min. acc. TT	D3DH
[BW06]	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(\sqrt{N})$	min. acc. TR	mod. D3DH
[FNP07]	$\mathcal{O}(1)$	$\mathcal{O}(t^2\log(N\varepsilon))$	$\mathcal{O}(t^2\log(N\varepsilon))$	min. acc. TT	DBDH
[BP08]	$\mathcal{O}(1)$	$\mathcal{O}(\frac{(t\log t)^2}{(1-\delta)^2}\log(N/\varepsilon))$	$\mathcal{O}(\frac{(t\log t)^2}{(1-\delta)^2}\log(N/\varepsilon))$	min. acc. TT	IND-CPA SE
[BN08]	$\mathcal{O}(1)$	$\mathcal{O}(\frac{(t\log t)^2}{(1-\delta)^2}\log(N/\varepsilon))$	$\mathcal{O}(\frac{(t\log t)^2}{(1-\delta)^2}\log(N/\varepsilon))$	min. acc. TT	IND-CPA PKE

Table 3.3: Comparison of black-box traitor tracing schemes

We now go into more detail and compare the concrete performance of some schemes, for a user set size of 2^{30} and a security level of 128 bits, which we let correspond to a RSA modulus length of 3072 bits. The size of group elements is derived using the simplifying assumptions from section 2.3.4.

Practical performance of FNP The scheme uses a symmetric pairing in prime-order groups $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, and its security rests on the DBDH assumption, so elements of $\mathbb{G}$ have 256-bit representations and elements of $\mathbb{G}_T$ have 3072-bit representations.

According to [FNP07], for a tracing error probability of 2^{-30} and at most 30 colluding users, the codewords have length $n \approx 5\ 000\ 000$.

The public EK consists of $10n + 3$ elements from $\mathbb{G}$, or $2560n + 768$ bits plus the description of a hash function, and the MSK contains $10n + 2$ scalars from $\mathbb{Z}_q$, for a length of $2560n + 512$ bits, while the user secret key consists of $2n$ scalars ($512n$ bits). The encryption of a message of length $\kappa(n-1)$ bits (80 MB) consists of 2 elements of $\mathbb{G}_T$, 2 elements of $\mathbb{G}$, and $\kappa n + \log n$ bits or $128n + 6679$ bits. This is an overhead of 6807 bits (851 B) for a ciphertext of 80 MB, which gives us a ciphertext rate of approximately 1.

Practical performance of BN The scheme uses an IND-CPA secure PKE scheme, which we instantiate here with EC-ElGamal with 256-bit group elements. We need to use a robust collusion-secure, which will be longer than the code we used for the evaluation of the FNP scheme. Let n be the length of a codeword.

The scheme which is secure against perfect decoders (which correctly decrypt all well-formed ciphertexts) has a public EK consisting of $2n$ PKE public keys, for a length of $512n$ bits, and a MSK of the same length, which consists of $2n$ PKE secret keys. A user secret key consists of n PKE secret keys and the user's codeword for a length of $257n$ bits. The ciphertext of a 128-bit message consists of 2 PKE ciphertexts, or 4 group elements, for a length of 512 bits. This gives us a ciphertext rate of 4. With randomness reuse and hybrid encryption, we could compress the ciphertext to one group element and two symmetric encryptions, or 256 bits, for a ciphertext rate of 2.

Practical performance of BSW The scheme uses a symmetric pairing in a composite-order group, where the group order is a product of two primes. Security rests on the decision 3-party Diffie-Hellman assumption, the subgroup decision assumption, and the bilinear subgroup decision assumption, which we treat as equivalent to DDH and DLOG for the purposes of this comparison. We set the size of $\mathbb{G}$-elements to 3072 bits and of $\mathbb{G}_T$-elements to 6144 bits. The size of an element of the q-order subgroup $\mathbb{G}_q \subset \mathbb{G}$ is 1536 bits.

A public key EK consists of $2\sqrt{N}+1$ elements from $\mathbb{G}_q$, $\sqrt{N}+2$ elements from $\mathbb{G}$, and $\sqrt{N}$ elements from $\mathbb{G}_T$, for a total size of 402 660 864 bits, or 50.3 MB. The MSK contains 3 1536-bit elements and $3\sqrt{N}$ elements from $\mathbb{Z}_n$, for a total length of 301 994 496 bits, or almost 37.8 MB. A user secret key consists of a single element in $\mathbb{G}$, for a length of 3072 bits or 0.4 KB. The ciphertext of a 6144-bit message consists of $5\sqrt{N}$ elements from $\mathbb{G}$, plus $\sqrt{N}$ elements from $\mathbb{G}_T$, for a length of 704 643 072 bits, or 88 MB, which gives us a ciphertext rate of 114 688.

3.13 Message-based Traitor Tracing

Traitor tracing is only concerned with tracing pirate decoders. It offers no protection against users that make symmetric session keys or even the content itself public, because these are the same for all

users and therefore untraceable. Message-based traitor tracing uses watermarks and distributes different versions of the same content to the users, which makes it possible to trace traitors based on the version of the message that is leaked.

Fiat and Tassa [FT99] present a scheme which they call *dynamic traitor tracing*. The code that governs which marked versions are distributed to which users is generated on the fly, and is adapted to the feedback from the pirate network. The authors assume that the tracing authority has real-time access to e. g. a pirated video stream. This makes it possible to adapt to the number of traitors instead of bounding their number during setup. Because traitors which are detected can be revoked, it is possible to trace more than one traitor.

Safavi-Naini and Wang [SNW03a] remark that the FT scheme heavily depends on the delay between broadcasting the content and analysing it to determine that watermark allocation. They propose a *sequential traitor tracing* scheme that still proceeds in rounds, but where the mark allocation is predetermined and not vulnerable to delayed broadcasting on the side of the traitors. Again, when a traitor is discovered, he is revoked and the tracing continues with the goal to trace all traitors.

Tassa [Tas05] constructs another dynamic traitor tracing scheme, where the bit of a codeword assigned to a used depends on the feedback from the previous round. He improves the scheme from [FT99] by combining it with Boneh-Shaw fingerprinting codes.

Jin, Lotspiech, and Nusser [JLN04] propose a scheme based on error-correcting codes and reduce the ciphertext overhead by splitting the tracing between an inner code that allocates watermarks to single movies, and an outer code that coordinates the allocation of codewords to the movies themselves. This means that tracing must take place over several movies, with the guarantee that the set of traitors does not change in between.

Jin and Lotspiech [JL07] build on the [JLN04] scheme by adding a revocation mechanism, which allows to revoke a traced user, and still be able to trace afterwards.

Chapter 4

Security Notions for Broadcast Encryption

Contents

When comparing BE schemes as in chapter 3, we are interested not only in the various indicators of efficiency, but also in the security level the schemes achieve. We found that while it is fairly easy to compare the efficiency of BE schemes, most papers define their own security notions, and it is not immediately clear how they compare. For this reason, we define a more systematic security model for broadcast encryption schemes, and construct a generic security framework for BE. As always in the provable security framework, we take into account oracles to model the means available to the adversary, such as the possibility to join new users, to corrupt users, and to decrypt messages. It is worth noting that small details can have a high impact. For example, the choice of the set of users to which the challenge message is encrypted also plays a role in how the models relate to each other. We investigate the relationships between the different notions, and find that while in a lot of cases we can prove the separations we expected, some cases are more involved and sometimes the relationship depends on the collusion-resistance of the BE schemes. After describing the relationships between notions in our framework, we have a closer look at the security models and the schemes proposed in the literature, and discuss where they are in our framework, which then makes it easy to compare them.

Our results are relevant for existing BE schemes. We can see that some schemes achieve stronger security guarantees than were proved in the original paper. For example, from the proof found in [GW09], it is clear that the two-key transformation actually achieves a stronger form of adaptive security. This further underlines the importance of having clear security definitions.

Related work The first BE scheme to come with a security argument was the subset-cover framework introduced by Naor, Naor, and Lotspiech [NNL01]. The framework uses symmetric keys, where the sender and the receivers share some secrets, so the security proof relies on assumptions about the symmetric primitives (one-way functions and PRGs/stream ciphers). Dodis and Fazio [DF03b] presented the first CCA2 secure public-key trace and revoke (TR) scheme along with a security model covering CCA2

and generalized CCA2, a notion where the adversary is prohibited to ask ciphertexts to the decryption oracle which fulfill some efficiently computable equivalence relation with the challenge ciphertext. When considering possible corruption after the target ciphertext has been sent, one has to deal with forward security. This was done by Yao, Fazio, Dodis, and Lysyanskaya [YFDL04] who first considered forward security for HIBE and then by extension for BE. Boneh, Gentry, and Waters [BGW05] designed a fully collusion-resistant BE scheme and proposed a security model for it, where the adversary can corrupt all the users, except the target users. Thereafter, Boneh and Waters [BW06] presented a fully collusion-resistant TR scheme secure against adaptive attacks. Delerablée, Paillier, and Pointcheval [DPP07] also presented a fully collusion secure dynamic BE scheme (DBE) and presented a new matching security model. More recently, Gentry and Waters [GW09] defined two additional security notions they call "semi-static" and "adaptive", and propose a generic transformation from a semi-static secure scheme into an adaptively secure scheme. They then construct a semi-statically secure scheme to which they apply the transformation.

Organization In section 4.1 we define our security framework. Section 4.2 relates the different security notions to each other. In section 4.3 we embed the existing security models from the literature into our framework. In section 4.4, we describe which security notions have been achieved by existing protocols, taking into account our new findings. To show that our definitions are not empty, we present a construction of a scheme that achieves the strongest security notion in section 4.5.

4.1 Security Notions

Besides the various properties that a broadcast encryption scheme can satisfy, many security notions have been defined to take all the threats into consideration. We will review them, and try to give a cleaner view. As usual, security notions are defined by the goal the adversary want to achieve, and by the means that are available. We first define our framework of security notions, and then compare this with some alternatives defined in the literature.

4.1.1 Standard Security Notions

Since we are studying a KEM (sec. 2.5.3), the goal of the adversary, noted IND for *key indistinguishability*, is to distinguish two keys in a key encapsulation: After having received the public parameters, in the first phase (the FIND-phase) the adversary outputs a target set $\mathcal{S}$; then the challenger runs the key encapsulation algorithm on this set $\mathcal{S}$, which outputs the ephemeral K and its encapsulation H. The challenger then chooses a random key K' and a random bit b and sets $K_b = K$ and $K_{1-b} = K'$. Upon receiving (H, K_0, K_1), the adversary runs the second phase (the GUESS-phase), during which it has to decide whether H encapsulates K_0 or K_1, which means it has to guess the bit b.

Oracles can be available at different periods of time (before Setup, during the FIND-phase, or during both the FIND- and GUESS-phase) which defines several kinds of attacks. Figure 4.1 shows the experiment $\mathrm{Exp}^{\mathsf{ind-dxaycca}z}_{\mathcal{DBE},\mathcal{A}}(\kappa)$, where the oracles OJoin_1, $\mathsf{OCorrupt}_1$ and $\mathsf{ODecaps}_1$ are available during the FIND-phase, and the oracles OJoin_2, $\mathsf{OCorrupt}_2$ and $\mathsf{ODecaps}_2$ are available during the GUESS-phase. According to the exact definition of these oracles, we have an IND-Dyn**x**-Ady-CCA**z** security game, for x-Dynamic (Join), y-Adaptive (Corrupt) and CCA-z (Decaps). If not otherwise specified, use of the variables x, y, z means that they can be replaced by any level defined below.

The Join oracle It can be available at the Setup-time only. In this case, the adversary can make a number of non-adaptive Join-queries, where he receives the results only at the end of the Setup-phase, together with the parameters and MSK, EK. As said above, we then talk about a **static scheme**, and the attack is *s-dynamic* (or Dyn**S**), and both the oracles OJoin_1 and OJoin_2 output $\perp$. The Join-oracle can be available during the first phase only, then $\mathsf{OJoin}_1 = \mathsf{Join}$ but the OJoin_2 oracle outputs $\perp$, and the attack is *1-dynamic* (or Dyn**1**); it can be available always, then $\mathsf{OJoin}_1 = \mathsf{OJoin}_2 = \mathsf{Join}$, and the attack is *2-dynamic* (or Dyn**2**).

The Corrupt oracle Corruptions can be more or less adaptive. Again, the adversary may have to decide before the Setup-time which users will be corrupted. This is a **selective attack** or *s-adaptive* (also denoted Ad**S**), which is meaningful for static schemes (Dyn**S**) only (otherwise there are no users to corrupt during the Setup-phase), and then both the oracles $\mathsf{OCorrupt}_1$ and $\mathsf{OCorrupt}_2$ output $\perp$. It can be available during the first phase only, then $\mathsf{OCorrupt}_1 = \mathsf{Corrupt}$ but the $\mathsf{OCorrupt}_2$ oracle

$\mathrm{Exp}^{\mathsf{ind-d}x\mathsf{ay}\mathsf{cca}z\text{-}b}_{\mathcal{DBE},\mathcal{A}}(\kappa)$
$\mathcal{Q}_C \leftarrow \emptyset$; $\mathcal{Q}_D \leftarrow \emptyset$;
$(\mathsf{MSK}, \mathsf{EK}) \leftarrow \mathsf{Setup}(1^\kappa)$;
$(st, \mathcal{S}) \leftarrow \mathcal{A}^{\mathsf{OJoin}_1(\cdot),\mathsf{OCorrupt}_1(\cdot),\mathsf{ODecaps}_1(\cdot,\cdot,\cdot)}(\mathsf{EK})$;
$(H, K) \leftarrow \mathsf{Encaps}(\mathsf{EK}, Reg, \mathcal{S})$; $K_b \leftarrow K$; $K_{1-b} \xleftarrow{\$} \mathcal{K}$;
$b' \leftarrow \mathcal{A}^{\mathsf{OJoin}_2(\cdot),\mathsf{OCorrupt}_2(\cdot),\mathsf{ODecaps}_2(\cdot,\cdot,\cdot)}(st; \mathcal{S}, H, K_0, K_1)$;
`if` $\exists i \in \mathcal{S}, (i, \mathcal{S}, H) \in \mathcal{Q}_D$ `or` $i \in \mathcal{Q}_C$
`then return` 0;
`else return` b';

where $x \in \{s, 1, 2\}$, $y \in \{0, s, 1, 2\}$, $z \in \{0, 1, 2\}$.

$\mathsf{OJoin}(i)$
$(\mathsf{usk}_i, \mathsf{upk}_i) \leftarrow \mathsf{Join}(\mathsf{MSK}, i)$;
`return` upk_i;

$\mathsf{OCorrupt}(i)$
$\mathcal{Q}_C \leftarrow \mathcal{Q}_C \cup \{i\}$;
`return` usk_i;

$\mathsf{ODecaps}(i, \mathcal{S}, H)$
$\mathcal{Q}_D \leftarrow \mathcal{Q}_D \cup \{(i, \mathcal{S}, H)\}$
$K \leftarrow \mathsf{Decaps}(\mathsf{usk}_i, \mathcal{S}, H)$;
`return` K;

Figure 4.1: $\mathcal{DBE}$: Key privacy (IND-Dyn**x**-Ady-CCA**z**)

outputs $\perp$, and the attack is *1-adaptive* (or Ad**1**). It can be available during the full security game, then $\mathsf{OCorrupt}_1 = \mathsf{OCorrupt}_2 = \mathsf{Corrupt}$, and the attack is *2-adaptive* (or Ad**2**). Eventually, the adversary can have no access at all to the Corrupt oracle: we say the attack is *0-adaptive* (or Ad**0**).

The Decaps oracle As usual for chosen-ciphertext security, the Decaps-oracle can be available or not. It is not available in the CPA (or CCA**0**) scenario, and both the oracles $\mathsf{ODecaps}_1$ and $\mathsf{ODecaps}_2$ output $\perp$; it can be available during the first phase only, then $\mathsf{ODecaps}_1 = \mathsf{Decaps}$ but the $\mathsf{ODecaps}_2$ oracle outputs $\perp$, and the attack is CCA**1**; it can be available during the full security game, then $\mathsf{ODecaps}_1 = \mathsf{ODecaps}_2 = \mathsf{Decaps}$, and the attack is CCA**2**.

For the IND-goal, the natural restriction for the adversary is not to ask for the decapsulation of the challenge header H nor corrupt any user in the target set $\mathcal{S}$.

Remark 4.1. *For private-key schemes, the adversary is granted access to the encapsulation oracle instead of the encryption key. In the rest of the section, we will focus on the public-key setting for dynamic broadcast encryption schemes (noted PKDBE).*

Definition 4.2. A public-key DBE scheme $\mathcal{DBE}$ is $(t, N, q_C, q_D, \varepsilon)$-IND-Dyn**x**-Ady-CCA**z** secure if for all $\kappa \in \mathbb{N}$, in the security game presented in figure 4.1, the advantage $\mathrm{Adv}^{\mathsf{ind-d}x\mathsf{ay}\mathsf{cca}z}_{\mathcal{DBE}}(\kappa, t, N, q_C, q_D)$ of any t-time adversary $\mathcal{A}$ registering at most N users (OJoin oracle), corrupting at most q_C of them (OCorrupt oracle), and asking for at most q_D decapsulation queries (ODecaps oracle), is bounded by ε:

$$\mathrm{Adv}^{\mathsf{ind-d}x\mathsf{ay}\mathsf{cca}z}_{\mathcal{DBE}}(\kappa, t, N, q_C, q_D) = \max_{\mathcal{A}}\{\Pr[\mathrm{Exp}^{\mathsf{ind-d}x\mathsf{ay}\mathsf{cca}z\text{-}1}_{\mathcal{DBE},\mathcal{A}}(\kappa) = 1] - \Pr[\mathrm{Exp}^{\mathsf{ind-d}x\mathsf{ay}\mathsf{cca}z\text{-}0}_{\mathcal{DBE},\mathcal{A}}(\kappa) = 1]\}.$$

4.1.2 Alternatives and Variants

Forward-secrecy For dynamic revocation schemes (the target set is defined by the list of revoked users), new users are by definition included in the target sets of the message headers, even if they did not exist at the time the header was sent. This means the encryption does not provide forward-secrecy. Furthermore, since new users are included in the challenge set S, the adversary is not allowed to corrupt them, so new users are implicitly assumed to be honest. To model forward-secrecy, we can allow corruption of joined users, and in this case the encryption key EK must evolve when a new user joins the system.

For dynamic BE schemes (where the target set is defined by the list of authorized users), the Ad**2** notion provides *forward-secrecy* since any user not in the target set can be corrupted in the second phase.

Target set In the default security game, the adversary chooses the target set $\mathcal{S}$ at the end of the first phase, the FIND-phase, which consists in finding the best $\mathcal{S}$ for winning the game. But some papers in the literature restricted this choice:

- The adversary announces the target set before the setup phase [BGW05]. We call this *selective security*, denoted Targ**S**. This can only happen in static schemes, because the adversary needs to know the set of users to choose the target set from.
- The target set is automatically set to all uncorrupted users at the end of the first phase [DF03b]. We call this *fixed-target-set security*, denoted Targ**F**.

Security	before setup	FIND-Phase	Challenge-Phase	GUESS-Phase
Ad**2**		Corrupt	$\mathcal{S}$	Corrupt
Ad**2**Targ**F**		Corrupt		Corrupt
Ad**1**		Corrupt	$\mathcal{S}$	
Ad**1**Targ**F**		Corrupt		
semi-static	$\mathcal{C}$		$\mathcal{S}$	
static	$\mathcal{C}$			

Table 4.1: Adversarial capabilities

When needed, the default case (the adversary chooses the target set $\mathcal{S}$ at the end of the FIND-phase) is denoted Targ**C**.

Malleability of the id-header By restricting the adversary only to asking queries different from (S, H) to the decapsulation oracle, our security definition implies non-malleability of the id header. If the adversary manages to submit a valid query (i, S', H) with a different target set for the original header, he wins the game. Our definition thus goes beyond the one found in [BGW05], where the restriction is only that the same H must not be queried to the decapsulation oracle. By removing S from the list $\mathcal{Q}_D$, our definition can be weakened so it does not require S to be protected.

Security models in the literature We can now characterize all the security models defined in the literature into our formalism: These notions are summarized in table 4.1, where $\mathcal{S}$ is the target set and $\mathcal{C}$ the set of corrupted users.

- In [YFDL04], the authors defined the full access to the Corrupt oracle, but for a static scheme (no Join oracle). In order to accommodate the forward-secrecy, they included time slots. Disregarding the latter, the security model is similar to IND-Dyn**S**-Ad**2**-CCA**2**-Targ**F**. Essentially the adversary is restricted to corrupting only users from a time slot later than the one the challenge message was sent in. In our model, IND-Dyn**x**-Ad**2**-CCA**z**-Targ**F** security does only make sense for $x = 2$, as otherwise no users can be corrupted in the GUESS-phase (because the target set is fixed to $U \setminus \mathcal{C}$ and the adversary cannot join new users after the challenge phase).
- In [Del08] the authors define a security model for IBBE they call IND-sID-CCA (selective ID CCA security), which is IND-Dyn**S**-Ad**2**-CCA**2**-Targ**S** security in our notation.
- Adaptive access to the Corrupt oracle has been used in [BW06] and [GW09]. In our notation, the authors used IND-Dyn**S**-Ad**1**-CCA**0** security, since no decapsulation queries are available, and corruption only occurs in the FIND-phase.
- As noted above, the fixed-target-set security was introduced in [DF03b], but no Corrupt queries were allowed in the second phase, and the system was static (no Join query). In our formalism, this is IND-Dyn**S**-Ad**1**-CCA**z**-Targ**F**, according to the Decaps-oracle access.
- *Semi-static* security has been introduced in [GW09] in order to build a generic conversion to adaptive-1. In this setting, the adversary must announce the set of corrupted users before the setup phase, as we defined as *selective attack*. In our notation, this is IND-Dyn**S**-Ad**S**-CCA**0** security. [1]
- In the *static* model due to [BGW05], the adversary also has to announce its target set before the setup phase (selective attack). In our notation, this is IND-Dyn**S**-Ad**S**-CCA**2**-Targ**F** security with fixed target set. The authors also define a CPA version. [2]

Collusion resistance We can also distinguish between two types of collusion-resistance: full collusion-resistance, where there is no limit on the number of Corrupt-queries, and t-collusion-resistance, where the number of queries is bounded by t (which can depend on the number of users N). With our parameters, we implicitly consider all the cases.

1. More precisely, in the *semi-static* version of the experiment, the adversary must commit to a set $\tilde{\mathcal{S}}$ before the setup phase. He is allowed to corrupt any user not in $\tilde{\mathcal{S}}$ after the setup phase, and must choose a challenge set $\mathcal{S} \subseteq \tilde{\mathcal{S}}$. An equivalent formulation is that the adversary chooses the set $\mathcal{C}$ of users to corrupt before the setup phase (because he can corrupt all users not in $\tilde{\mathcal{S}}$), but chooses $\mathcal{S}$ at the challenge phase. This formulation is only equivalent for fully collusion-resilient schemes, but it is for these schemes that the notions were designed.

2. In the *static* version of the experiment [BGW05], the adversary has to announce the set $\mathcal{S}$ of users he wants to attack before the setup phase. He then receives the private keys of all users not in $\mathcal{S}$ after the setup phase. An equivalent definition is that he chooses the set $\mathcal{C}$ of corrupted users, and the $\mathcal{S}$ is fixed to be all the users except $\mathcal{C}$. To allow the adversary to choose the target set, the adversary announces both $\mathcal{C}$ and $\mathcal{S}$ before the setup phase. This definition where the adversary chooses both $\mathcal{C}$ and $\mathcal{S}$ can also be used in not fully collusion secure schemes and is the one considered in this section.

4.2 Relationship between the Security Notions

In this section, we shed light on the relationships between the security notions we defined in the last section. We start in section 4.2.1 with the hierarchy of Decaps-oracles, where we expect no surprises. In section 4.2.2, we explore the Join-oracle, of which we defined three different versions: For the passive version, which takes no input, all notions are equivalent; for the active version, which takes input and outputs a user tag, we can separate all notions. For the IBBE version, which takes an arbitrary string as input, but does not output a user tag (the upk is the identity of the user), we can show equivalences and separations based on the availability of a Corrupt-oracle. In section 4.2.3, we address the Corrupt-queries, and gaps appear according to the number of such queries, and thus the level of collusion-resistance. In section 4.2.4, we examine the various ways in which the target set can be chosen.

4.2.1 Separating CPA and CCA

We remember the well-known separation between CPA (CCA**0**), CCA**1**, and CCA**2** security for PKE from [BDPR98]. The same separation applies in the case of broadcast encryption, first because if we set $\mathsf{KeyGen}(1^\kappa)$ to
$(\mathsf{MSK}, \mathsf{EK}) \leftarrow \mathsf{Setup}(1^\kappa)$; $(\mathsf{usk}_1, \mathsf{upk}_1) \leftarrow \mathsf{Join}(\mathsf{MSK}, 1)$; $\mathsf{dk} \stackrel{\mathsf{def}}{=} \mathsf{usk}_1, \mathsf{ek} \stackrel{\mathsf{def}}{=} \mathsf{EK}||\mathsf{upk}_1$, we obtain a single-user KEM scheme. But for completeness, and as a warm-up, we give a proof of the relationship for BE, whatever the size of the target set, part of which leans closely on the proof in [BDPR98].

Theorem 4.3. *The following implications are strict:*

$$\textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{y}\textit{-CCA2} \Rightarrow \textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{y}\textit{-CCA1} \Rightarrow \textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{y}\textit{-CCA0}.$$

Proof. The implications are clear, since having access to an additional oracle always strengthens the adversary.

CCA0 vs. CCA1 For this separation, we can use a simpler proof because we do not have to worry about non-malleability. We modify an IND-Dyn**x**-Ad**y**-CCA**0** secure BE scheme Π into a scheme Π' that is still IND-Dyn**x**-Ad**y**-CCA**0** but obviously not IND-Dyn**x**-Ad**y**-CCA**1**:

- $\Pi'.\mathsf{Encaps}(\mathsf{EK}, Reg, S)$:
  $(H, K) \leftarrow \Pi.\mathsf{Encaps}(\mathsf{EK}, Reg, S)$;
  `return` $(0||H, K)$.
- $\Pi'.\mathsf{Decaps}(\mathsf{usk}_{\mathsf{id}}, S, H)$:
  parse H as $b||H'$;
  `if` $b = 0$ `then` $K \leftarrow \Pi.\mathsf{Decaps}(\mathsf{usk}_{\mathsf{id}}, S, H')$; `return` K; `fi`;
  `if` $b = 1$ `then return` $\mathsf{usk}_{\mathsf{id}}$; `fi`;

Π' is clearly not IND-Dyn**x**-Ad**y**-CCA**1**, because one call to the decapsulation oracle reveals the secret key. Π' remains IND-Dyn**x**-Ad**y**-CCA**0**, because the only difference is in the decapsulation oracle, which is not used. Furthermore, everything the adversary interacts with, EK and the OJoin and OCorrupt oracles, is unchanged and the only task of the simulator is to prepend a 0 to the challenge key header.

CCA1 vs. CCA2 This separation can use malleability with a bit b as above, except that b is disregarded in the decapsulation process:

- $\Pi'.\mathsf{Encaps}(\mathsf{EK}, Reg, S)$:
  $(H, K) \leftarrow \Pi.\mathsf{Encaps}(\mathsf{EK}, Reg, S)$;
  `return` $(0||H, K)$.
- $\Pi'.\mathsf{Decaps}(\mathsf{usk}_{\mathsf{id}}, S, H)$:
  parse H as $b||H'$;
  $K \leftarrow \Pi.\mathsf{Decaps}(\mathsf{usk}_{\mathsf{id}}, S, H')$;
  `return` K.

This construction is not IND-Dyn**x**-Ad**y**-CCA**2** secure, because on receiving a challenge $0||H$, the adversary can query $1||H$ to the ODecaps, which returns K. It is IND-Dyn**x**-Ad**y**-CCA**1** secure, because all possible queries in the FIND-phase can be perfectly simulated by removing b. □

4.2.2 Separating Notions of Dynamicity

In this section, in order to compare the Join-oracle access, we also have to consider the three versions of the Join-algorithm, as defined in section 2.6.1: *passive*-Join, if it takes no input; *active*-Join, if it takes an input; *ID-based*-Join, if the output tag upk is the input identity.

Easy Implications

There is a clear hierarchy on the Join oracle access: at the setup time only, in the first phase, or in both phases.

Theorem 4.4. *The following implications hold for all versions of the* Join oracle: IND-Dyn**2**-Ad**y**-CCA**z** $\Rightarrow$ IND-Dyn**1**-Ad**y**-CCA**z** $\Rightarrow$ IND-Dyn**S**-Ad**y**-CCA**z**.

Passive Join

This is a standard definition in the literature. Interestingly enough, in this context all the notions are equivalent, since the adversary cannot influence the output. [3]

Theorem 4.5. *If* Join *takes no input,we have the following equivalences*

$$\textit{IND-Dyn2-Ady-CCAz} \Leftrightarrow \textit{IND-Dyn1-Ady-CCAz} \Leftrightarrow \textit{IND-DynS-Ady-CCAz}.$$

Proof. Because of the trivial implications, it remains to show that Dyn**S** $\Rightarrow$ Dyn**2**. Given a successful Dyn**2**-adversary $\mathcal{A}_d$ that makes N_1 queries to the Join-oracle in phase 1, and N_2 queries to the Join-oracle in phase 2, we construct a successful Dyn**S**-adversary $\mathcal{A}_s$ that joins $N = N_1 + N_2$ users before the setup phase. Because the Join-oracle takes no input, its behavior is exactly the same in phase 1 and phase 2. Therefore $\mathcal{A}_s$ can store the results and then answer all Join-queries made by $\mathcal{A}_d$ later. □

Active Join with Large Input

If the Join-algorithm is interactive or takes input from the adversary (that can be sufficiently large, i. e. $|\mathcal{UI}|$ is superpolynomial), the adversary can influence the Join-process:

Theorem 4.6. *If* Join takes input and outputs a public tag, the following implications are strict

$$\text{IND-Dyn}\mathbf{2}\text{-Ad}\mathbf{y}\text{-CCA}\mathbf{z} \Rightarrow \text{IND-Dyn}\mathbf{1}\text{-Ad}\mathbf{y}\text{-CCA}\mathbf{z} \Rightarrow \text{IND-Dyn}\mathbf{S}\text{-Ad}\mathbf{y}\text{-CCA}\mathbf{z}.$$

Proof. We first study Dyn**S** vs. Dyn**1**, and then Dyn**1** vs. Dyn**2**.

DynS vs. Dyn1 Let Π be a IND-Dyn**S**-Ad**y**-CCA**z** secure BE scheme. We construct a scheme Π' as follows:

- $\Pi'.\mathsf{Setup}(1^\kappa)$:
  $(\mathsf{EK}', \mathsf{MSK}') \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$;
  $r \xleftarrow{\$} \mathcal{UI}$;
  $\mathsf{EK} \stackrel{\mathsf{def}}{=} \mathsf{EK}'||r, \mathsf{MSK} \stackrel{\mathsf{def}}{=} \mathsf{MSK}'||r$;
  `return` $(\mathsf{EK}, \mathsf{MSK})$.
- $\Pi'.\mathsf{Join}(\mathsf{MSK}, \mathsf{id})$:
  $(\mathsf{usk}'_{\mathsf{id}}, \mathsf{upk}'_{\mathsf{id}}) \leftarrow \Pi.\mathsf{Join}(\mathsf{MSK}', \mathsf{id})$;
  `if` $\mathsf{id} = r$, `then` $\mathsf{upk}'_{\mathsf{id}} \stackrel{\mathsf{def}}{=} \mathsf{upk}'_{\mathsf{id}}||\mathsf{MSK}$; `fi`;
  `return` $(\mathsf{usk}'_{\mathsf{id}}, \mathsf{upk}'_{\mathsf{id}})$.

Π' is not IND-Dyn**1**-Ad**y**-CCA**z** secure, because if the adversary has access to a OJoin oracle, he can query OJoin(r) and get MSK. Intuitively, Π' is still IND-Dyn**S**-Ad**y**-CCA**z** secure, because the users are joined randomly and $|\mathcal{UI}|$ is super-polynomial: user r is in the user set with negligible probability.

3. It is interesting to note that the equivalence is for our above notions only: for passive-Join, a query in the first phase is strictly more useful than a query in the second phase. As a consequence, if we consider in details the number of queries in each phase, as done in [PP04] for the encryption and decryption oracles, we can show that $\mathsf{Dyn}(N_1 + N_2, 0) \rightarrow \mathsf{Dyn}(N_1, N_2) \rightarrow \mathsf{Dyn}(0, N_1 + N_2)$, and these implications are strict. However, in the above theorem, we do not fix the number of queries.

Dyn1 vs. Dyn2 We modify a Dyn**1** secure BE scheme Π in such a way that the Join-oracle effectively doubles as a Decaps-oracle. This means that an id-string fulfills two roles: It is interpreted as a user identifier and as the input to a Decaps-oracle.

– $\Pi'.\mathsf{Join}(\mathsf{MSK}, \mathsf{id})$:
$(\mathsf{usk}'_{\mathsf{id}}, \mathsf{upk}'_{\mathsf{id}}) \leftarrow \Pi.\mathsf{Join}(\mathsf{MSK}, \mathsf{id})$;
parse id as (i, S, H); (possible if we assume S and H to have a fixed length)
$(\mathsf{usk}_i, \mathsf{upk}_i) \leftarrow \Pi.\mathsf{Join}(\mathsf{MSK}, i)$; (note that i is a prefix of id)
$K \leftarrow \Pi.\mathsf{Decaps}(\mathsf{usk}_i, S, H)$;
`if` $K = \perp$ `then` $K \xleftarrow{\$} \mathcal{K}$; `fi`;
$\mathsf{upk}_{\mathsf{id}} \stackrel{\mathsf{def}}{=} \mathsf{upk}'_{\mathsf{id}} || K$;
`return` $(\mathsf{usk}'_{\mathsf{id}}, \mathsf{upk}_{\mathsf{id}})$.

Π' is not IND-Dyn**2**-Ady-CCA**z** secure, because the adversary can use $\mathsf{OJoin}(y)$ to decrypt the challenge message. Intuitively, Π' is still IND-Dyn**1**-Ady-CCA**z** secure, because the adversary has only a negligible chance to guess the challenge key header H (which encapsulates a key of high entropy) before the challenge phase. □

Identity-based

In this case, the Join-algorithm only outputs a user secret key $\mathsf{usk}_{\mathsf{id}}$ (because $\mathsf{upk}_{\mathsf{id}} = \mathsf{id}$), so the OJoin-oracle only returns the id that it was given as input, making it effectively useless. In addition, for ID-based schemes, we assume that all users exist from the beginning, because we can encrypt messages to them before their secret keys have been created.

This means that a OJoin-oracle is useless to the adversary, but to keep notation consistent, we define that for IBBE schemes it is always available (Dyn**2**).

4.2.3 Separating Forms of Corruption

Theorem 4.7.

$$\textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{2}\textit{-CCA}\boldsymbol{z} \Rightarrow \textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{1}\textit{-CCA}\boldsymbol{z} \Rightarrow \textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{0}\textit{-CCA}\boldsymbol{z},$$
$$\textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{1}\textit{-CCA}\boldsymbol{z} \Rightarrow \textit{IND-Dyn}\boldsymbol{S}\textit{-Ad}\boldsymbol{S}\textit{-CCA}\boldsymbol{z} \Rightarrow \textit{IND-Dyn}\boldsymbol{S}\textit{-Ad}\boldsymbol{0}\textit{-CCA}\boldsymbol{z},$$

and for BE schemes that are not fully collusion secure all implications are strict.

Proof. The implications are clear, since having access to an oracle never makes an adversary weaker. The separations follow from lemmas 4.8, 4.9 in conjunction with theorem 4.6, 4.10, 4.11, and 4.12 □

Separation of no Corruption from Selective Corruption

Recall that for Ad**S**, the only version of Dyn that makes sense is Dyn**S** (section 4.1.1).

Lemma 4.8. *IND-Dyn**S**-Ad**0**-CCA**z** $\not\Rightarrow$ IND-Dyn**S**-Ad**S**-CCA**z**.*

Proof. Let Π be a BE scheme that is Ad**0** secure (no corruption). We construct a scheme Π' that is still secure in this model, but not Ad**S** secure (selective corruption). We only change the Join-algorithm.

– $\Pi'.\mathsf{Join}(\mathsf{MSK}, \mathsf{id})$: $(\mathsf{usk}_{\mathsf{id}}, \mathsf{upk}_{\mathsf{id}}) \leftarrow \Pi.\mathsf{Join}(\mathsf{MSK}, \mathsf{id})$; `return` $(\mathsf{usk}_{\mathsf{id}} || \mathsf{MSK}, \mathsf{upk}_{\mathsf{id}})$.

The scheme remains secure against adversaries that do not corrupt any user, because they will never see any $\mathsf{usk}_{\mathsf{id}}$'s, which is all that changed. In case of corruption, the adversary learns MSK. □

Separation of Selective Corruption from 1-Adaptive Corruption

In a model with selective corruption, the adversary must announce the set $\mathcal{C}$ of corrupted users before seeing the encryption key. To make a difference, we would have to give some information on the subset of the users to corrupt in the encryption key, then embed some information in the user secret keys using a secret sharing scheme to make sure all of the identified users (special users) have to be corrupted.

We need to make sure that the subset is hard to guess. This is the case for IBBE, where the size of the set $\mathcal{UI}$ is exponential and any user is hard to guess. In case $\mathcal{UI}$ is of polynomial size, the size of the subset must not be too small, otherwise all of the special users will be corrupted even by a selective-corruption adversary with significant probability. If t is the number of special users, there are $\binom{N}{t}$ ways of choosing them. For IBBE, $\binom{|\mathcal{UI}|}{1}$ is already exponential in the security parameter. To make

the binomial be super-polynomial for a polynomial-size N, we need t to be non-constant. Due to the symmetry $\binom{N}{t} = \binom{N}{N-t}$, we also need $N-t$ to be non-constant.

To be sure that the adversary corrupts at most t users, we an use a t-collusion secure scheme. In the following, we thus focus on t-collusion secure schemes, where t must be non-constant and less than the total number of users minus a non-constant number.

Lemma 4.9. *For a t-collusion secure scheme (for t and $N-t$ non-constant numbers),*

$$\textit{IND-Dyn}\mathbf{S}\textit{-Ad}\mathbf{S}\textit{-CCAz} \not\Rightarrow \textit{IND-Dyn}\mathbf{S}\textit{-Ad}\mathbf{1}\textit{-CCAz}.$$

Proof. Let Π be a BE scheme that is $\mathsf{Ad}\mathbf{S}$ secure. We construct a BE scheme Π' that is still $\mathsf{Ad}\mathbf{S}$ secure but not $\mathsf{Ad}\mathbf{1}$ secure.

- $\Pi'.\mathsf{Setup}(1^\kappa, N)$:
  $(\mathsf{MSK}', \mathsf{EK}') \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$;
  Choose a random subset $I \subset U$, with $|I| = t$;
  Use a t-out-of-t secret sharing scheme: $\bigoplus_{i\in I} s_i = \mathsf{MSK}'$;
  $\mathsf{EK} \stackrel{\mathsf{def}}{=} \mathsf{EK}'||I, \mathsf{MSK} \stackrel{\mathsf{def}}{=} \mathsf{MSK}'||I||\{s_i\}_{i\in I}$;
  `return` $(\mathsf{MSK}, \mathsf{EK})$.
- $\Pi'.\mathsf{Join}(\mathsf{MSK}, \mathsf{id})$:
  parse MSK as $\mathsf{MSK}'||I||\{s_i\}$;
  $(\mathsf{upk}'_{\mathsf{id}}, \mathsf{usk}'_{\mathsf{id}}) \leftarrow \Pi.\mathsf{Join}(\mathsf{MSK}', \mathsf{id})$;
  `if` $\mathsf{id} \notin I$, $s_{\mathsf{id}} \stackrel{\$}{\leftarrow} \{0,1\}^k$; `fi`;
  $\mathsf{usk}_{\mathsf{id}} \stackrel{\mathsf{def}}{=} \mathsf{usk}'_{\mathsf{id}}||s_{\mathsf{id}}$;
  `return` $(\mathsf{upk}'_{\mathsf{id}}, \mathsf{usk}_{\mathsf{id}})$.

The scheme is insecure under $\mathsf{Ad}\mathbf{1}$ attacks, because the adversary extracts I from EK, then corrupts all users in I and computes the MSK. Intuitively, it remains $\mathsf{Ad}\mathbf{S}$ secure because the adversary must choose the users he corrupts before setup is called. Because we use a secret sharing scheme, he cannot learn any additional information unless he corrupts all users in I, which happens with negligible probability. This is true if the adversary does not corrupt almost all the users, hence the restriction to t-collusion secure schemes, with t no too big. □

Separation of no Corruption from 1-Adaptive Corruption

Since for selective corruption we could only consider static schemes, we give a separation lemma for any kind of dynamicity between no corruption and 1-adaptive corruption.

Lemma 4.10. $\textit{IND-Dyn}\mathbf{x}\textit{-Ad}\mathbf{0}\textit{-CCAz} \not\Rightarrow \textit{IND-Dyn}\mathbf{x}\textit{-Ad}\mathbf{1}\textit{-CCAz}.$

The proof is the same as the proof of lemma 4.8.

Separation of 1-Adaptive Corruption from 2-Adaptive Corruption

Lemma 4.11. *For a t-collusion secure scheme (for t and $N-t$ non-constant numbers),*

$$\textit{IND-Dyn}\mathbf{x}\textit{-Ad}\mathbf{1}\textit{-CCAz} \not\Rightarrow \textit{IND-Dyn}\mathbf{x}\textit{-Ad}\mathbf{2}\textit{-CCAz} \text{ for } z \in \{0,1\}.$$

Proof. We modify a $\mathsf{Ad}\mathbf{1}$ secure BE scheme Π as above with a secret sharing, but of the ephemeral key at the challenge phase:

- $\Pi'.\mathsf{Encaps}(\mathsf{EK}, S)$:
  $(H', K) \leftarrow \Pi.\mathsf{Encaps}(\mathsf{EK}, Reg, S)$;
  Choose a random subset $I \subset U$, with $|I| = t$;
  $\forall i \in I : (H_i, K_i) \leftarrow \Pi.\mathsf{Encaps}(\mathsf{EK}, \{i\})$;
  Set $K_0 = K \bigoplus_{i\in I} K_i$;
  `return` $(H'||K_0||\{H_i\}_{i\in I}, K)$.

The Decaps-algorithm just uses H', and drop the rest of the ciphertext. Π' is not $\mathsf{Ad}\mathbf{2}$ secure, since the adversary can corrupt all users in I after receiving the challenge. Π' is still $\mathsf{Ad}\mathbf{1}$ secure because the adversary cannot guess I before seeing the challenge, under the restriction that the number of corrupted users is not too big. □

As noted, the proof requires t and $N-t$ to be non-constant. But we can also note that it does not work in the CCA**2**-setting, because on the one hand the scheme is malleable, and on the other hand the adversary could simply query the H_i's to the Decaps-oracle.

Lemma 4.12. *For a t-collusion secure scheme (for t and $N-t$ non-constant numbers),if SUF-CMA secure MAC, IND-CCA2 secure symmetric encryption and homomorphic OWF exist,*

$$\textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{1}\textit{-CCA}\boldsymbol{2} \not\Rightarrow \textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{2}\textit{-CCA}\boldsymbol{2}.$$

MAC, symmetric encryption, and homomorphic OWF are defined in section 2.4.

Proof. We use a proof similar to the one for NM-CCA1 $\not\Rightarrow$ NM-CCA2 in [BDPR98]. Let t be the maximum number of users the adversary is allowed to corrupt, N the number of users at the end of the game. We assume that both t and $N-t$ are non-constant. Assume that $(\mathsf{Dec}, \mathsf{Enc})$ is an IND-CCA2 secure symmetric encryption, $(\mathsf{GenMac}, \mathsf{VerifMac})$ a SUF-CMA secure MAC, f a homomorphic OWF with $f(x)+f(y)=f(x+y)$ (an example of this, assuming discrete logarithm is hard, is discrete exponentiation). We modify an IND-Dyn**x**-Ad**1**-CCA**2** secure BE scheme Π as follows.

$\Pi'.\mathsf{Setup}(1^\kappa)$:
$(\mathsf{MSK}', \mathsf{EK}') \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$;
`for all` $\mathsf{id} \in U:\ r_{\mathsf{id}} \xleftarrow{\$} \{0,1\}^k$;
$\mathsf{MSK} \stackrel{\mathsf{def}}{=} \mathsf{MSK}'||\{r_{\mathsf{id}}\}_{\mathsf{id}\in U}$;
`return` $(\mathsf{MSK}, \mathsf{EK}')$.

$\Pi'.\mathsf{Join}(\mathsf{MSK}, \mathsf{id})$:
parse MSK as $\mathsf{MSK}'||\{r_{\mathsf{id}}\}$;
$(\mathsf{usk}'_{\mathsf{id}}, \mathsf{upk}'_{\mathsf{id}}) \leftarrow \Pi.\mathsf{Join}(\mathsf{MSK}', \mathsf{id})$;
$\mathsf{usk}_{\mathsf{id}} \stackrel{\mathsf{def}}{=} \mathsf{usk}'_{\mathsf{id}}||r_{\mathsf{id}}||\{f(r_i)\}_{i\in U}$;
(if r_{id} undef., $r_{\mathsf{id}} \stackrel{\mathsf{def}}{=} 0$)
`return` $(\mathsf{usk}_{\mathsf{id}}, \mathsf{upk}'_{\mathsf{id}})$.

$\Pi'.\mathsf{Encaps}(\mathsf{EK}, Reg, S)$:
$(H', K') \leftarrow \Pi.\mathsf{Encaps}(\mathsf{EK}', S)$;
$\mathcal{K}_m, K \xleftarrow{\$} \{0,1\}^k$;
choose $T \subset U, |T| = t$
$C \stackrel{\mathsf{def}}{=} \mathsf{Enc}(K', \mathcal{K}_m||K)$;
$M \stackrel{\mathsf{def}}{=} \mathsf{GenMac}(\mathcal{K}_m, T||C||H')$;
$H \stackrel{\mathsf{def}}{=} 0||T||C||H'||M$;
`return` (H, K').

$\Pi'.\mathsf{Decaps}(\mathsf{usk}_{\mathsf{id}}, S, H)$:
parse $\mathsf{usk}_{\mathsf{id}} = \mathsf{usk}'_{\mathsf{id}}||r_{\mathsf{id}}||\{f(r_i)\}$;
parse $H = b||T||C||H'||M||R$
`if` $(b=0$ and $R=\emptyset)$
or $(b=1$ and $f(R) = \sum_{i\in T} f(r_i))$
$K' \stackrel{\mathsf{def}}{=} \Pi.\mathsf{Decaps}(\mathsf{usk}'_{\mathsf{id}}, S, H')$;
$\mathcal{K}_m||K \stackrel{\mathsf{def}}{=} \mathsf{Dec}(K', C)$;
`if` $\mathsf{VerifMac}(\mathcal{K}_m, T||C||H', M) = 1$
`then return` K
`else return` $\perp$.
`else return` $\perp$.

Π' is not IND-Dyn**x**-Ad**2**-CCA**2** secure, because the adversary can corrupt the right users to retrieve all r_i, and then exploit malleability: he can construct a key header that decrypts to the same key as the challenge. However, Π' is still IND-Dyn**x**-Ad**1**-CCA**2** secure, because the adversary has only a negligible chance to corrupt the right users that he learns only in the challenge phase. The MAC avoids the malleability in this case. □

4.2.4 Choice of the Target Set

Selective Security

Lemma 4.13. *The following implication is strict:*

$$\textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{y}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{C} \Rightarrow \textit{IND-Dyn}\boldsymbol{S}\textit{-Ad}\boldsymbol{y}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{S}.$$

Proof. The implication is clear, as it is always possible to choose the same $\mathcal{S}$ in the challenge phase that has been output before the setup phase. Let Π be a Targ**S** secure BE scheme. We construct a BE scheme Π' that is still Targ**S** secure, but insecure if the adversary is allowed to select the target set during the challenge phase (Targ**C**).

- $\Pi'.\mathsf{Setup}$ works as $\Pi.\mathsf{Setup}$, but appends a randomly chosen subset T of users to EK.
- $\Pi'.\mathsf{Encaps}$ works as $\Pi.\mathsf{Encaps}$, except if the target set S is the set T. In this case, it uses the random coins 0^k (a constant one, publicly known).

Π' is IND-Dyn**x**-Ad**S**-CCA**z**-Targ**S** secure since the adversary has to announce the target set S before seeing EK and he can guess T only with negligible probability. Π' is IND-Dyn**x**-Ad**y**-CCA**z**-Targ**C**-insecure if the adversary can freely choose S after the setup phase, because the adversary receives EK before having to output the challenge set S. Then the challenge K is deterministically chosen. Even if he has to choose $\mathcal{C}$ before setup, he can choose $\mathcal{C} = \emptyset$, so he can choose any S he wants. □

Fixed Target Sets

In our definition the adversary chooses the target set S of the challenge. In the security model in [DPP07], S is automatically the set of all non-compromised users. The same situation appears in [BGW05], where the adversary outputs S before the setup and receives the secret keys for all users in $U \setminus S$. We could reformulate the BGW model so that the adversary outputs the set $\mathcal{C}$ of the keys he wants to know, and S is set to $U \setminus \mathcal{C}$. This formulation is obviously equivalent. We want to investigate the relationship between these two notions.

Note that under the "fixed" definition, the notions IND-Dyn**x**-Ad**1**-CCA**z** and IND-Dyn**x**-Ad**2**-CCA**z** for $x \in \{s, 1\}$ are equivalent since in any case the adversary cannot corrupt users after the challenge phase (all the non-corrupted users at the end of the first phases are in the target set and cannot be corrupted).

Theorem 4.14. *All the following implications are strict*

$$\begin{aligned}
&\textit{IND-Dyn}\boldsymbol{S}\textit{-Ad}\boldsymbol{S}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{C} \Rightarrow \textit{IND-Dyn}\boldsymbol{S}\textit{-Ad}\boldsymbol{S}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{S} \Leftrightarrow \textit{IND-Dyn}\boldsymbol{S}\textit{-Ad}\boldsymbol{S}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{F}\\
&\textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{0}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{C} \Rightarrow \textit{IND-Dyn}\boldsymbol{S}\textit{-Ad}\boldsymbol{0}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{S} \Rightarrow \textit{IND-Dyn}\boldsymbol{S}\textit{-Ad}\boldsymbol{0}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{F}\\
&\textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{0}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{C} \Rightarrow \textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{0}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{F}
\end{aligned}$$

The theorem follows from lemmas 4.13, 4.15, 4.16, 4.17, and 4.18.

Lemma 4.15. $\textit{IND-Dyn}\boldsymbol{S}\textit{-Ad}\boldsymbol{y}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{S} \Rightarrow \textit{IND-Dyn}\boldsymbol{S}\textit{-Ad}\boldsymbol{y}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{F}$
for $y \in \{0, s\}$.

Proof. From an adversary $\mathcal{A}_f$ against the IND-Dyn**S**-Ad**y**-CCA**z**-Targ**F** security of a BE scheme, we build an adversary $\mathcal{A}_s$ against the IND-Dyn**S**-Ad**y**-CCA**z**-Targ**S** security. If the model has no corruption or static corruption, $\mathcal{A}_s$ runs $\mathcal{A}_f$, who outputs $\mathcal{C}$, chooses the same $\mathcal{C}$ and sets his target set $S = U \setminus \mathcal{C}$. □

Lemma 4.16. $\textit{IND-Dyn}\boldsymbol{S}\textit{-Ad}\boldsymbol{S}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{F} \Rightarrow \textit{IND-Dyn}\boldsymbol{S}\textit{-Ad}\boldsymbol{S}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{S}$.

Proof. Given a successful adversary $\mathcal{A}_S$, we construct an adversary $\mathcal{A}_f$ as follows. $\mathcal{A}_S$ outputs his target set S and the set of users to corrupt $\mathcal{C}$ before the Setup phase. $\mathcal{A}_f$ chooses $\mathcal{C}' = U \setminus S$. □

Lemma 4.17. $\textit{IND-Dyn}\boldsymbol{S}\textit{-Ad}\boldsymbol{0}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{F} \not\Rightarrow \textit{IND-Dyn}\boldsymbol{S}\textit{-Ad}\boldsymbol{0}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{S}$.

Proof. In the IND-Dyn**S**-Ad**0**-CCA**z**-Targ**F**-experiment, the target set is always fixed to $S = U$. Given a IND-Dyn**S**-Ad**0**-CCA**z**-Targ**F** secure scheme Π, we construct a IND-Dyn**S**-Ad**0**-CCA**z**-Targ**F** secure scheme Π', which is not IND-Dyn**S**-Ad**0**-CCA**z**-Targ**S** secure. The only change is that if $|S| = 1$, $\Pi'.\mathsf{Encaps}$ sets $K = 0$ (or determines the key in a deterministic way by fixing all random coins e. g. to 0). □

Lemma 4.18. $\textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{0}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{C} \Rightarrow \textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{0}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{F}$ *is a strict implication.*

Proof. The implication is clear, as being able to choose the target set is not weaker than having the target set fixed by the challenger. The nonimplication is proved in the same way as in the proof of lemma 4.17. □

Theorem 4.19. *For fully collusion-resilient BE schemes, the following implications are strict*

$$\begin{aligned}
\textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{y}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{C} &\Leftrightarrow \textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{y}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{F}\\
&\Rightarrow \textit{IND-Dyn}\boldsymbol{S}\textit{-Ad}\boldsymbol{y}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{S} \quad (y \in \{1, 2\})
\end{aligned}$$

The theorem follows from lemmas 4.13 and 4.20. It seem curious at first that the relationship between fixed target set and selective security is inverted for models with no corruption, but in this case the fixed target set means that it is always set to U, while the selective security allows some freedom of the adversary to choose.

Lemma 4.20. *For fully collusion-resistant BE schemes*

$$\textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{y}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{C} \Leftrightarrow \textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{y}\textit{-CCA}\boldsymbol{z}\textit{-Targ}\boldsymbol{F} \quad (y \in \{1, 2\}).$$

Proof. It is clear that if the adversary can choose S freely, he can set it to $U \setminus \mathcal{C}$. Let $\mathcal{A}_c$ be a successful adversary against a BE scheme that can choose his target set S. Then we construct $\mathcal{A}_f$ as follows: $\mathcal{A}_f$ faithfully forwards all queries. When $\mathcal{A}_c$ outputs his challenge target set S, $\mathcal{A}_f$ first issues corrupt queries so that $U \setminus \mathcal{C} = S$, then asks for the challenge and forwards it to $\mathcal{A}_c$. He forwards the guess bit b and wins with the same probability as $\mathcal{A}_c$. □

Note that $\mathcal{A}_f$ corrupts more users, which could reduce the tightness of a security proof, and causes the proof to fail in a t-resilient setting where $t < N-1$ (if $t = N-1$, the scheme is fully collusion-resistant).

In the following, we denote by $\nRightarrow\!\!\!\Leftarrow$ the fact that $\not\Rightarrow$ in both directions.

Theorem 4.21. *For BE schemes where the adversary must leave at least two users uncorrupted, the following implications are strict (where $y \in \{1,2\}$):*

$$\begin{aligned}\mathit{IND\text{-}Dyn}\boldsymbol{x}\mathit{\text{-}Ad}\boldsymbol{y}\mathit{\text{-}CCA}\boldsymbol{z}\mathit{\text{-}Targ}\boldsymbol{C} &\Rightarrow \mathit{IND\text{-}Dyn}\boldsymbol{S}\mathit{\text{-}Ad}\boldsymbol{y}\mathit{\text{-}CCA}\boldsymbol{z}\mathit{\text{-}Targ}\boldsymbol{S}\\ \mathit{IND\text{-}Dyn}\boldsymbol{x}\mathit{\text{-}Ad}\boldsymbol{y}\mathit{\text{-}CCA}\boldsymbol{z}\mathit{\text{-}Targ}\boldsymbol{C} &\Rightarrow \mathit{IND\text{-}Dyn}\boldsymbol{x}\mathit{\text{-}Ad}\boldsymbol{y}\mathit{\text{-}CCA}\boldsymbol{z}\mathit{\text{-}Targ}\boldsymbol{F}\\ \mathit{IND\text{-}Dyn}\boldsymbol{x}\mathit{\text{-}Ad}\boldsymbol{y}\mathit{\text{-}CCA}\boldsymbol{z}\mathit{\text{-}Targ}\boldsymbol{F} &\nRightarrow\!\!\!\Leftarrow \mathit{IND\text{-}Dyn}\boldsymbol{S}\mathit{\text{-}Ad}\boldsymbol{y}\mathit{\text{-}CCA}\boldsymbol{z}\mathit{\text{-}Targ}\boldsymbol{S}\end{aligned}$$

The theorem follows from lemmas 4.13, 4.22, 4.23, and 4.24.

Lemma 4.22. *If the adversary is restricted to leaving at least 2 users uncorrupted, the following implication is strict*

$$\mathit{IND\text{-}Dyn}\boldsymbol{x}\mathit{\text{-}Ad}\boldsymbol{y}\mathit{\text{-}CCA}\boldsymbol{z}\mathit{\text{-}Targ}\boldsymbol{C} \Rightarrow \mathit{IND\text{-}Dyn}\boldsymbol{x}\mathit{\text{-}Ad}\boldsymbol{y}\mathit{\text{-}CCA}\boldsymbol{z}\mathit{\text{-}Targ}\boldsymbol{F} \quad (y \in \{1,2\}).$$

Proof. The implication is clear, as the adversary can always set $S = U \setminus C$. For the reverse direction, we exploit the fact that $|S| > 1$ and modify a scheme Π that we assume to be IND-Dyn**x**-Ad**S**-CCA**z**-Targ**F** secure as follows. If $|S| = 1$, Encaps sets $K = 0$ (or in deterministic way as in the proof of lemma 4.17). □

Lemma 4.23. *If the adversary is restricted to leaving at least 2 users uncorrupted,*

$$\mathit{IND\text{-}Dyn}\boldsymbol{x}\mathit{\text{-}Ad}\boldsymbol{y}\mathit{\text{-}CCA}\boldsymbol{z}\mathit{\text{-}Targ}\boldsymbol{F} \not\Rightarrow \mathit{IND\text{-}Dyn}\boldsymbol{S}\mathit{\text{-}Ad}\boldsymbol{y}\mathit{\text{-}CCA}\boldsymbol{z}\mathit{\text{-}Targ}\boldsymbol{S}.$$

Proof. If $|\mathcal{C}| \leq N-2$ and the target set is fixed to $S = U \setminus \mathcal{C}$, then $|S| \geq N-(N-2) = 2$ Given a IND-Dyn**x**-Ad**0**-CCA**z**-Targ**F** secure scheme Π, we exploit this to construct a scheme Π' that is IND-Dyn**x**-Ad**0**-CCA**z**-Targ**F** secure, but not IND-Dyn**S**-Ad**0**-CCA**z**-Targ**S**. The only change is as in the proof of lemma 4.22: if $|S| = 1$, Π'.Encaps sets $K = 0$. □

We can easily see that the adversary does not get weaker if he can choose the target set freely from the set of uncorrupted users $U \setminus \mathcal{C}$, because he can choose $S = U \setminus \mathcal{C}$ as in the fixed case.

Lemma 4.24.

$$\mathit{IND\text{-}Dyn}\boldsymbol{S}\mathit{\text{-}Ad}\boldsymbol{y}\mathit{\text{-}CCA}\boldsymbol{z}\mathit{\text{-}Targ}\boldsymbol{S} \not\Rightarrow \mathit{IND\text{-}Dyn}\boldsymbol{S}\mathit{\text{-}Ad}\boldsymbol{y}\mathit{\text{-}CCA}\boldsymbol{z}\mathit{\text{-}Targ}\boldsymbol{F} \text{ for } y \in \{1,2\}.$$

Proof. Given a IND-Dyn**S**-Ad**y**-CCA**z**-Targ**S** secure BE scheme Π, we construct another BE scheme Π' that is also IND-Dyn**S**-Ad**y**-CCA**z**-Targ**S** secure, but is IND-Dyn**S**-Ad**y**-CCA**z**-Targ**F**-insecure.

- $\Pi'.\mathsf{Setup}(1^\kappa, N)$: $(\mathsf{EK}', \mathsf{MSK}') \leftarrow \Pi.\mathsf{Setup}(1^\kappa, N)$; chooses $T \subset U$; $\mathsf{EK} \stackrel{\mathsf{def}}{=} \mathsf{EK}'||T$; `return` $(\mathsf{MSK}', \mathsf{EK})$.
- $\Pi'.\mathsf{Encaps}(\mathsf{EK}, S)$: $(H', K') \leftarrow \Pi.\mathsf{Encaps}(\mathsf{EK}', S)$ `if` $S = T$ `then` $K \stackrel{\mathsf{def}}{=} 0$

Π' is still IND-Dyn**S**-Ad**y**-CCA**z**-Targ**S** secure, because the adversary cannot guess T with significant probability. Π' is not IND-Dyn**S**-Ad**y**-CCA**z**-Targ**F** secure, since the adversary can corrupt all users in $U \setminus T$, so that $S = T$. □

4.3 Relationships Between Notions from the Literature

A security notion that our model does not cover is defined in [DPP07]. In this model, the adversary accesses a JoinCorrupted oracle instead of the Corrupt oracle. That means he must decide whether to corrupt a user before the user is joined, but the choice can depend on information gained previously. The model defined in [DPP07] is Dyn**1**, as the adversary has access to a Join oracle before the challenge phase, CCA**0** and Targ**F**, as the challenge set is *fixed* to $S = U \setminus \mathcal{C}$, so it is rather similar to IND-Dyn**1**-Ad**1**-CCA**0**-Targ**F**-model in our framework, except that the Corrupt oracle is replaced with JoinCorrupted. We call it the *partially adaptive* model and denote this by Ad**P1** of the JoinCorrupted-oracle is available only in the FIND-phase, and by Ad**P2** if it is available in both phases. Its relationship to other notions from the literature is depicted in figure 4.2. As in the previous section, we also denote Targ**C** the default case where the adversary can choose S as any subset of $U \setminus \mathcal{C}$.

Theorem 4.25. *We have the following implications*

$$\begin{aligned}&\mathit{IND\text{-}Dyn}\boldsymbol{1}\mathit{\text{-}Ad}\boldsymbol{1}\mathit{\text{-}CCA}\boldsymbol{z}\mathit{\text{-}Targ}\boldsymbol{F} \Rightarrow \mathit{IND\text{-}Dyn}\boldsymbol{1}\mathit{\text{-}Ad}\boldsymbol{P1}\mathit{\text{-}CCA}\boldsymbol{z}\mathit{\text{-}Targ}\boldsymbol{C}\\ \Rightarrow\ &\mathit{IND\text{-}Dyn}\boldsymbol{1}\mathit{\text{-}Ad}\boldsymbol{P1}\mathit{\text{-}CCA}\boldsymbol{z}\mathit{\text{-}Targ}\boldsymbol{F} \Rightarrow \mathit{IND\text{-}Dyn}\boldsymbol{S}\mathit{\text{-}Ad}\boldsymbol{1}\mathit{\text{-}CCA}\boldsymbol{z}\mathit{\text{-}Targ}\boldsymbol{S}\end{aligned}$$

that are all strict (the first one only if t-collusion secure with t and $N-t$ non-constant).

Figure 4.2: Relations between security notions from the literature (all implications are strict, *: for t-collusion secure schemes with t and $N-t$ non-constant)

Proof. We first show the implications.

1. Let $\mathcal{A}_{pac}$ be a partially adaptive choice adversary. From $\mathcal{A}_{pac}$ we construct an adaptive-1 adversary $\mathcal{A}_a$ as follows. $\mathcal{A}_a$ forwards all Join-queries made by $\mathcal{A}_{pac}$, and substitutes each call to JoinCorrupted with Join, then Corrupt. When $\mathcal{A}_{pac}$ outputs his target set S, $\mathcal{A}_a$ corrupts all users not in S.
2. The second implication is clear.
3. From any selective-target-set adaptive-1 adversary $\mathcal{A}_s$ we construct a partially adaptive adversary as follows. $\mathcal{A}_s$ announces S before the setup phase. $\mathcal{A}_{pa}$ joins all users in S and JoinCorrupts all users in $U \setminus S$. He now has enough information to answer all Corrupt queries.

The separations follow from lemmas 4.26, 4.27, and 4.28. □

Lemma 4.26. *For t-collusion secure schemes with t and $N-t$ non-constant,*

$$\textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{P}\boldsymbol{y}\textit{-CCA}\boldsymbol{z} \not\Rightarrow \textit{IND-Dyn}\boldsymbol{x}\textit{-Ad}\boldsymbol{y}\textit{-CCA}\boldsymbol{z} \text{ for } y \in \{1,2\}.$$

The advantage the adversary has when using Corrupt queries is that he can view the system after all users are joined, before he has to decide who to corrupt.

We proceed as in the proof of lemma 4.9, but instead of encoding I in EK, we append a bit to each tag that indicates whether the user is in I. Then, the adversary knows this only after a user has joined.

Proof. Fix a degree of adaptiveness and let Π be a scheme secure in this model. We construct a scheme Π' that is secure in this model, but insecure when the JoinCorrupted oracle is replaced with a Corrupt oracle.

- $\Pi'.\mathsf{Setup}(1^\kappa)$:
  $(\mathsf{MSK}', \mathsf{EK}') \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$;
  $N \stackrel{\text{def}}{=} 2t$;
  determine a random subset $I \subset [N]$ with $|I| = t$;
  $\bigoplus_{i \in I} s_i = \mathsf{MSK}'$;
  $\mathsf{MSK} \stackrel{\text{def}}{=} \mathsf{MSK}' || I || \{s_i\}_{i \in I}$;
  `return` $(\mathsf{MSK}, \mathsf{EK}')$.
- $\Pi'.\mathsf{Extract}(\mathsf{MSK}, \mathsf{id})$:
  parse MSK as $\mathsf{MSK}' || I || \{s_i\}$;
  $(\mathsf{usk}'_{\mathsf{id}}, \mathsf{upk}'_{\mathsf{id}}) \leftarrow \Pi'.\mathsf{Extract}(\mathsf{MSK}', \mathsf{id})$;
  `if` $\mathsf{id} \in I$: $\mathsf{usk}_{\mathsf{id}} \stackrel{\text{def}}{=} \mathsf{usk}'_{\mathsf{id}} || s_{\mathsf{id}}, \mathsf{upk}_{\mathsf{id}} \stackrel{\text{def}}{=} \mathsf{upk}'_{\mathsf{id}} || 1$ `fi`;
  `if` $\mathsf{id} \notin I$: $\mathsf{usk}_{\mathsf{id}} \stackrel{\text{def}}{=} \mathsf{usk}'_{\mathsf{id}} || 0^k, \mathsf{upk}_{\mathsf{id}} \stackrel{\text{def}}{=} \mathsf{upk}'_{\mathsf{id}} || 0$ `fi`;
  `return` $(\mathsf{usk}_{\mathsf{id}}, \mathsf{upk}_{\mathsf{id}})$.

Π' *is not secure under corruption attacks*, because the adversary joins $2t$ users, then corrupts those which have a 1 as the last bit of their upk and computes the MSK.

Π' *is secure under JoinCorrupted attacks*, because the adversary must choose the users he corrupts before he joins them. Therefore, he cannot learn any additional information unless he JoinCorrupts all users in I, which happens with negligible probability. □

Lemma 4.27. *When considering partially adaptive attacks, Targ$\boldsymbol{F}$ $\not\Rightarrow$ Targ$\boldsymbol{C}$.*

Proof. Let Π be a public-key BE scheme secure against partially adaptive attacks where the challenge set is fixed to $U \setminus \mathcal{C}$. Then we construct a BE scheme Π' that is still secure against an adversary whose target set is fixed, but insecure against any adversary who can choose his target set. Let N be the maximum number of users if the scheme is static, otherwise set $N = \kappa$.

- $\Pi'.\mathsf{Setup}(1^\kappa)$:
  $(\mathsf{MSK}', \mathsf{EK}') \leftarrow \Pi.\mathsf{Setup}(1^\kappa)$;
  select a random subset $I \subset [N]$;
  $\mathsf{MSK} \stackrel{\mathsf{def}}{=} \mathsf{MSK}'||I$;
  `return` $(\mathsf{MSK}, \mathsf{EK}')$.
- $\Pi'.\mathsf{Extract}(\mathsf{MSK}, \mathsf{id})$:
  parse MSK as $\mathsf{MSK}'||I$;
  $(\mathsf{usk}'_{\mathsf{id}}, \mathsf{upk}'_{\mathsf{id}}) \leftarrow \Pi'.\mathsf{Extract}(\mathsf{MSK}', \mathsf{id})$;
  `if` $\mathsf{id} \in I$: $\mathsf{upk}_{\mathsf{id}} \stackrel{\mathsf{def}}{=} \mathsf{upk}'_{\mathsf{id}}||1$ `fi`;
  `if` $\mathsf{id} \notin I$: $\mathsf{upk}_{\mathsf{id}} \stackrel{\mathsf{def}}{=} \mathsf{upk}'_{\mathsf{id}}||0$ `fi`;
  `return` $(\mathsf{usk}'_{\mathsf{id}}, \mathsf{upk}_{\mathsf{id}})$.
- $\Pi'.\mathsf{Encaps}(\mathsf{EK}, Reg, S)$:
  $(H, K) \leftarrow \Pi.\mathsf{Encaps}(\mathsf{EK}, S)$;
  `if` $S = I : K = 0^k$ `fi` (I is derived from the upk that are part of Reg)
  `return` (H, K).

Π' *is* ***TargF*** *secure*: Any fixed adversary has to guess S before he starts to corrupt users, because S is determined by his corruptions.

Π' *is not* ***TargC*** *secure*: Any adversary that can freely choose S corrupts no users and sets S to the users determined by the public tags. □

Lemma 4.28. *IND-Dyn**S**-Ad**1**-CCAz-Targ**S*** $\not\Rightarrow$ *IND-Dyn**1**-Ad**P1**-CCAz-Targ**F***.

Proof. We use exactly the same separation as in lemma 4.13. □

We now have almost all the results we need to establish the relationship between the security notions that can be found in the existing literature to fill the picture on figure 4.2. We now complete it.

Theorem 4.29. *The following implication is strict*

$$\textit{IND-Dyn}\mathbf{1}\textit{-Ad}\mathbf{P1}\textit{-CCAz-Targ}\mathbf{C} \Rightarrow \textit{IND-Dyn}\mathbf{S}\textit{-Ad}\mathbf{S}\textit{-CCAz-Targ}\mathbf{C}.$$

Proof. $\Rightarrow$ From any semi-static adversary $\mathcal{A}_s$ we construct a partially adaptive adversary $\mathcal{A}_{pa}$ as follows. $\mathcal{A}_S$ announces N and C before the setup phase. $\mathcal{A}_{pa}$ asks for $\mathsf{JoinCorrupted}$ on all users in C and simply joins all users in $U \setminus C$.

$\not\Leftarrow$ The separation is analogous to the one in lemma 4.9.

□

We relate semi-static security to the version of static security with 1-adaptive corruption defined in [GW09].

Theorem 4.30. *The following implication is strict*

$$\textit{IND-Dyn}\mathbf{S}\textit{-Ad}\mathbf{S}\textit{-CCAz-Targ}\mathbf{C} \Rightarrow \textit{IND-Dyn}\mathbf{S}\textit{-Ad}\mathbf{1}\textit{-CCAz-Targ}\mathbf{S}.$$

Proof. $\Rightarrow$ From any selectively 1-adaptive adversary $\mathcal{A}_a$ we construct a semi-static adversary $\mathcal{A}_s$. $\mathcal{A}_a$ announces N and S before the setup phase. $\mathcal{A}_s$ forwards N and sets $C = U \setminus S$. He now has enough information to answer all $\mathsf{Corrupt}$-queries.

$\not\Leftarrow$ The separation is analogous to the one in lemma 4.13.

□

Theorem 4.31. *IND-Dyn**1**-Ad**P1**-CCAz-Targ**F*** $\not\Leftrightarrow$ *IND-Dyn**S**-Ad**S**-CCAz-Targ**C***.

Proof. $\not\Rightarrow$ To separate the two notions, we construct a BE scheme that is $\mathsf{IND\text{-}Dyn}\mathbf{1}\text{-}\mathsf{Ad}\mathbf{P1}\text{-}\mathsf{CCAz\text{-}Targ}\mathbf{F}$ secure but not $\mathsf{IND\text{-}Dyn}\mathbf{S}\text{-}\mathsf{Ad}\mathbf{S}\text{-}\mathsf{CCAz\text{-}Targ}\mathbf{C}$ secure by exploiting the fact that if the adversary has to leave at least two users uncorrupted, $|S| > 1$. The scheme is exactly the same as the one in lemma 4.22.

	DF03	BGW05	DPP07	Del08	GW09	Naive
Dyn	Dyn**S**	Dyn**S**	Dyn**1**	Dyn**2**	Dyn**S**	Dyn**2**
Ad	Ad**1**	Ad**S**	Ad**P1**	Ad**2**	Ad**2**	Ad**2**
CCA	CCA**2**	CCA**0**	CCA**0**	CCA**0**	CCA**0**	CCA**2**
Targ	Targ**F**	Targ**F**	Targ**F**	Targ**S**	Targ**C**	Targ**C**

Table 4.2: Comparison between schemes.

$\not\Leftarrow$ For this separation, we construct a scheme that is IND-Dyn**S**-Ad**S**-CCA**z**-Targ**C** secure but not IND-Dyn**1**-Ad**P1**-CCA**z**-Targ**F** secure by exploiting the fact that the adversary has to announce the corrupted users before seeing the public key. The scheme is exactly the same as the one in lemma 4.9.

□

4.4 Previous Schemes

Let us now discuss on the previous schemes in order to compare them. Table 4.2 sums up the security levels for each of them.

DF03 Dodis and Fazio [DF03b] proposed the first scheme that is secure against adaptive adversaries. However, their scheme is in the Targ**F** model. Consequently, the scheme can only be Ad**1** secure, because any corrupted user in the second phase is implicitly included in the target set and can thus decrypt. In the DF03 scheme, the bound on the number of revoked users r_{max} must be fixed before the setup and as soon as there are more than r_{max} corrupted users, the scheme can be completely broken, in the sense that the MSK can be recovered. The DF03 scheme can be shown to be Ad**2** secure when the target set is adversarially chosen with the size of the revoked set bounded by r_{max} and the total number of corrupted users in both first and second phases is also bounded by r_{max}.

BGW05 In [BGW05], Boneh, Gentry, and Waters presented new methods for achieving fully collusion-resistant systems with short ciphertexts. However, the scheme is only proved secure in the static model (Ad**S**). As discussed in [GW09], the BGW proof of security requires an "exact cancellation" and there is not an obvious way to prove BGW05 to be semi-statically secure.

DPP07 In [DPP07], Delerablée, Paillier, and Pointcheval proposed a dynamic scheme that is partially adaptive secure.

Del08 The identity-based broadcast encryption in [Del08] deals with 2-adaptive corruption and enjoys constant ciphertext and private key sizes. However, the adversary has to announce its target set before the setup phase which corresponds to our selective security model.

GW09 In [GW09], the authors aim to construct efficient schemes that are adaptively secure and that resist to full collusion. The adaptive security mentioned in the paper correspond to our Ad**1** model. They introduced a two-key transformation that convert a semi-static system of $2N$ users into an Ad**1** secure system of N users. However, their schemes can be easily proved Ad**2** secure. Their schemes are not dynamic.

4.5 A Fully Secure Scheme

We have defined a hierarchy of security notions, but we do not know of any scheme that fulfills the strongest notion. To make sure that the definition is not empty and no such scheme can exist, we need to construct such a scheme.

In [PPS11], we demonstrated the existence of such a scheme by using the naive construction with IND-CCA2 secure PKE and a MAC to tie the component ciphertexts together. Proving that this construction achieves IND-Dyn**2**-Ad**2**-CCA**2** security is not hard. We have opted instead to present a very efficient construction from [PPSS12]. While this construction is very efficient, its security rests on new assumptions. In chapter 5, we present another scheme, which is IND-Dyn**2**-Ad**2**-CCA**2** secure under the DDH assumption in the standard model.

4.5.1 An Efficient Selectively CCA-Secure Broadcast Encryption

We construct a BE scheme based on the BGW scheme [BGW05]. We describe the system for (at most) $N-1$ users to be notationally consistent with the original BGW scheme, the system for N users can be defined accordingly. We define a broadcast encryption scheme hBGW (hashed BGW) in the following. Let $\mathsf{H}_\zeta : \mathbb{G} \mapsto \mathbb{Z}_p$ be a hash family indexed by ζ.

- $\mathsf{Setup}(1^\kappa, N-1)$ picks a random generator $g \in \mathbb{G}$, two random quantities $\alpha, \gamma \in \mathbb{Z}_p$, and a random index ζ for hash function H, computes $v = g^\gamma$, and outputs $\mathsf{MSK} = (\alpha, \gamma)$ and $\mathsf{EK} = (g, v, \zeta)$.
- $\mathsf{Join}(\mathsf{MSK}, i)$ computes $g_k = g^{(\alpha^k)}$ for $k = i, i+1, N+1-i$, and $N+1+i$, and $d_i = g_i^\gamma$, and outputs $\mathsf{usk}_i = d_i$ and $\mathsf{upk}_i = (g_i, g_{i+1}, g_{N+1-i}, g_{N+1+i})$. The secret key sk_i is given to the user, and EK is updated by appending pk_i.
- $\mathsf{Encaps}(\mathsf{EK}, S)$ picks a random $t \in \mathbb{Z}_p$ and sets $K = e(g_{N+1}, g)^t$, which can be computed as $K = e(g_{N+1-i}, g_i)^t$ for any i, computes H as follows, and outputs (H, K).

$$H = \langle g^t, (v \cdot g_1^{\mathsf{H}_\zeta(g^t)} \cdot \prod_{j \in S} g_{N+1-j})^t \rangle.$$

- $\mathsf{Decaps}(\mathsf{usk}_i, S, H)$ parses the header as $H = (C_0, C_1)$, checks if the following equation holds:

$$e(C_1, g) = e(v \cdot g_1^{\mathsf{H}_\zeta(C_0)} \cdot \prod_{j \in S} g_{n+1-j}, C_0),$$

  and if it does, then calculates the session key as follows:

$$K = \frac{e(C_1, g_i)}{e(d_i \cdot g_{1+i}^{\mathsf{H}_\zeta(C_0)} \cdot \prod_{j \in S \setminus \{i\}} g_{n+1-j+i},\ C_0)}.$$

In the following we bring a theorem which states that if the hash function H is a universal one-way hash function (def. 2.24), then the proposed scheme satisfies selective CCA security under the same assumption as that of the original scheme, namely N-BDHE. Intuitively, the main modification we make in (the encryption algorithm of) the original scheme is the introduction of $g_1^{\mathsf{H}_\zeta(g^t)}$. If this element is not present, as it is in the original scheme, given a header $H = (C_0, C_1)$ corresponding to a key K, one can compute the header (C_0^r, C_1^r) that corresponds to the key K^r, and hence the scheme is malleable. We show that a UOWHF is sufficient to eradicate malleability and get CCA security. This modification is inspired by a similar technique in [BMW05] which, in contrast, was shown to be applicable to an *identity-based* scheme. Here we show that a similar idea is applicable to BGW_1. The proof of the following theorem can be found in [PPSS12]. In the proof we use the structure of the keys in the scheme to simulate decryption queries.

Theorem 4.32. *The above scheme is IND-Dyn2-AdS-CCA2 secure if the N-BDHE problem is hard and H is a universal one-way hash function.*

On dynamicity Note that the bound on the number of users in hBGW does not prevent the system from being able to handle more than $N-1$ users. That is, as long as the system "jumps over" the users number N and $N+1$ (i.e., after user number $N-1$, the next user is numbered $N+2$), the system can handle polynomially many users more than $N-1$ and remains secure. The security of the scheme with more than $N-1$ users can be proved based on the following assumption: given the input h, and $\{g_k = g^{\alpha^k}\}$ for $k \in \{N+1-m, \ldots, N+1+m\} \setminus \{N+1\}$ for random $g, h \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_p$, it is hard to decide between $e(g_{n+1}, h)$ and a random $T \in \mathbb{G}_T$. It is not hard to see that this assumption is equivalent to the following assumption: given the input g, h, and $\{g_k = g^{\alpha^k}\}$ for $k \in \{1, \ldots, 2m\} \setminus \{m\}$ for random $h \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_p$, it is hard to decide between $e(g_m, h)$ and a random $T \in \mathbb{G}_T$. Here $m \geq N+2$ is the last user number to join. This assumption is comparable to the m-BDHE assumption. In fact, like the BDHE assumption, it is an instance of the GBDHE assumption. In view of this observation, hBGW is a *dynamic* broadcast encryption in the sense that: (1) the system setup and the ciphertext size are independent of the upper bound on the number of users; (2) a new user can join anytime without incurring modification of other user secret keys; and (3) the encryption key is incrementally updated by an operation of $\mathcal{O}(1)$ complexity.

Comparison The only broadcast encryption scheme in the literature that provides CCA security with constant-size ciphertexts is BGW_3(sec. 3.3.1). It has similar secret and public key sizes as our scheme. However, there are differences in terms of security assumptions and ciphertext size. BGW_3 uses a signature or a message authentication code (MAC) and is proved secure under N-BDHE plus the strong unforgeability (SUF) of the signature or the MAC, whereas hBGW needs N-BDHE plus a universal one-way hash function (UOWHF). In theory, SUF and UOWHF are equivalent (both are equivalent to one-wayness), but in practice, hash functions are generally much more efficient than signatures. In terms of ciphertext size, BGW_3 has a ciphertext whose size is (about) double that of BGW_1's ciphertext: a BGW_3 ciphertext consists of a BGW_1 ciphertext of two $\mathbb{G}$ elements, plus an element in $\mathbb{Z}_p$ and a signature (or a MAC tag). hBGW has the same ciphertext size as that of BGW_1, i.e., only two $\mathbb{G}$ elements. Note that although pk_i in hBGW includes four group elements, since there are some repeating values the final EK includes the three initial values plus only $2N-1$ extra values of g_i.

4.5.2 Achieving Adaptive CCA Security

Since we have a very efficient scheme with asymptotically optimal size secret keys and ciphertexts which is already proved selective CCA secure based on standard assumptions, in this section we try to see how further we can achieve in terms of security by considering reasonable generalizations of some standard assumptions, while retaining the same optimally efficient secret key and ciphertext sizes. We first propose reasonable generalizations of GBDHE and prove that they hold in the generic group model; then we prove that hBGW can be proved IND-Dyn**2**-Ad**2**-CCA**2** secure under these assumptions.

The OBDHE Assumption

We consider extending the GBDHE problem (sec. 2.3.2) assuming that an extra resource is also given: the *Diffie-Hellman computation oracle* $\mathcal{O}^{\text{DH}}_{g,e}$, that takes two inputs $u, v \in \mathbb{G}$ and outputs $w \in \mathbb{G}$ such that $e(u,v) = e(g,w)$.

As for GBDHE, we define the sets of polynomials $P = (p_1, \ldots, p_s)$ and $Q = (q_1, \ldots, q_t)$, with $p_1 = q_1 = 1$, and a polynomial f, where $\forall i,k : p_i, q_k, f \in \mathbb{F}_p[X_1, \ldots, X_n]$. Let $g^P = (g^{p_1}, \ldots, g^{p_s})$. We say that f is independent of (P,Q) if it cannot be written as $f = \sum_{i,j=1}^{s} a_{i,j} p_i p_j + \sum_{k=1}^{t} b_k q_k$ for constants $a_{i,j}$ and b_k.

Definition 4.33 (OBDHE)**.** The *oracle bilinear Diffie-Hellman problem* (OBDHE) is defined as follows: Given the input $g^{P(x_1,\ldots,x_n)}$ and $g_T^{Q(x_1,\ldots,x_n)}$ for random choices of $x_1, \ldots, x_n \in \mathbb{F}_p$, and access to the $\mathcal{O}^{\text{DH}}_{g,e}$ oracle, decide between $g_T^{f(x_1,\ldots,x_n)}$ and a random $T \in \mathbb{G}_T$.

Note that the GBDHE assumption implies that the only elements (dependent on $x_1, \ldots, x_n$ and) in $\mathbb{G}$ that can be computed are those in the form $g^{\sum a_i p_i}$. Also note that if we assume $u = g^{\sigma_u}$ and $v = g^{\sigma_v}$, we will have $w = \mathcal{O}^{\text{DH}}_{g,e}(u,v) = g^{\sigma_u \sigma_v}$. Hence, by providing access to $\mathcal{O}^{\text{DH}}_{g,e}$, basically a number of "free multiplications" in the exponent are given. Let us define $p' = \sigma_u \sigma_v$. If we consider q' queries to $\mathcal{O}^{\text{DH}}_{g,e}$, and the output to the i-th query represented as $w_i = g^{p'_i}$, we can define $P' = (p'_1, \ldots, p'_{q'})$. Our extension of the GBDHE assumption says that it is still hard to solve the GBDHE problem if these "free multiplications" in the exponent do not help breaking the independence property.

The OBDHE assumption says that it is hard to solve the decision (P,Q,f)-OBDHE problem if f is independent of $(P||P', Q)$.

Theorem 4.34. *The OBDHE assumption holds in the generic group model.*

The proof is in [PPSS12]. We prove an upper bound on the success of any generic algorithm trying to solve the OBDHE problem which is negligible if p, the order of $\mathbb{F}_p$, is super-polynomial. In fact, our proof is very similar to that of [BBG05], suggesting that our assumption is a natural and closely-related extension of GBDHE.

It is also worth to note that OBDHE is falsifiable by solving the corresponding $(P||P', Q, f)$-GBDHE problem efficiently.

The GKEA Assumption

We propose the generalized knowledge of exponent assumption (GKEA) as follows and prove that it holds in the generic group model.

In the following we use p to denote a polynomial (suppressing the random variables) and $p(x_1, \ldots, x_n)$ to denote the evaluation of p on the input $(x_1, \ldots, x_n)$. Let $P = (p_1, \ldots, p_s) \in \mathbb{F}_p[X_1, \ldots, X_n]^s$. Let the linear span of P, denoted by $\mathrm{Span}(P)$, be defined as the vector space containing all the polynomials in the form $\sum_{k=1}^{s} a_k p_k$.

Definition 4.35 (GKEA)**.** Let $P = (p_1, \ldots, p_s) \in \mathbb{F}_p[X_1, \ldots, X_n]^s$, where $p_1 = 1$. Let A be an algorithm that given $g^{P(x_1,\ldots,x_n)}$ for a random $(x_1, \ldots, x_n)$, outputs

$$\left((a_k)_{k=1}^{s},\ h,\ h^{q(x_1,\ldots,x_n)} \right), \qquad \text{such that} \quad q(x_1, \ldots, x_n) = \sum_{k=1}^{s} a_k p_k(x_1, \ldots, x_n).$$

Consider the subspace of $\mathrm{Span}(P)$ defined as $V_q = \{r \mid r, rq \in \mathrm{Span}(P)\}$ and let $\{r_i\}_{i=1}^{t}$ be a basis for V_q. Then, there exists an extractor that given the same input as A outputs

$$(b_i)_{i=1}^{t}, \qquad \text{such that} \quad \mathsf{dlog}_g(h) = \sum_{i=1}^{t} b_i r_i(x_1, \ldots, x_n).$$

This assumption basically says that the only way an adversary can produce pairs of the form (h, h^q) is to pick given pairs of the form (h_i, h_i^q) and output $(\prod h_i^{b_i}, \prod (h_i^q)^{b_i})$ for some known values of b_i.

For $P = (1, X)$ and $q(X) = X$, this becomes the original KEA of [Dam92], which basically says that given (g, g^x) the only way an adversary can produce pairs of the form (h, h^x) is to output $(g^b, (g^x)^b)$ for some known value of b. This assumption is referred to KEA1 in [HT98, BP04] and as Diffie-Hellman Knowledge (DHK) in [Den06]. A similar problem is formalized as strong Diffie-Hellman (SDH) in [ABR01].

For $P = (1, X, Y, YX)$ and $q(X, Y) = X$, this becomes the KEA3 assumption of [BP04], which basically says that given (g, g^x, f, f^x) the only way an adversary can produce pairs of the form (h, h^x) is to output $(g^b f^c, (g^x)^b (f^x)^c)$ for some known values of b and c. This assumption is referred to as Extended KEA (XKEA) in [AF07] and as Extended Diffie-Hellman Knowledge (EDHK) in [DP08].

The above two instances of the assumption have already been proved to hold in the generic group model [Den06, AF07, DP08].

We restate a theorem from [PPSS12].

Theorem 4.36. *The GKEA assumption holds in the generic group model.*

Adaptive CCA Security

In this section we prove hBGW adaptive CCA secure under our generalized versions of the BDHE and knowledge of exponent assumptions. To prove IND-Dyn**2**-Ad**2**-CCA**2** security, we basically show that a decryption query by the adversary that contains a valid ciphertext does not increase the (cryptographic) 'knowledge' of the adversary. Also note that since ciphertext validity is publicly verifiable, a decryption query that contains an invalid ciphertext does not increase the adversary's knowledge either. Hence we basically show that a CCA attack against the system is equivalent to a CPA attack, under the GKEA assumption and the hash function being a UOWHF. Furthermore, the access to $\mathcal{O}_{g,e}^{\mathrm{DH}}$ enables answering adaptive corruption queries.

The IND-ACCA security of the scheme rests on the hardness of the OBDHE and the GKEA problems and the fact that H is a UOWHF. Intuitively, selective CPA security stems from the BDHE assumption underlying the OBDHE assumption along with the hash function being a UOWHF; the Diffie-Hellman oracle enables adaptive security; and the CCA security is achieved from the GKEA assumption along with the hash function being a UOWHF. The following theorem is from [PPSS12].

Theorem 4.37. *The hBGW scheme is IND-Dyn2-Ad2-CCA2 secure if the OBDHE and the GKEA assumptions hold and* H *is a universal one-way hash function.*

We note that we prove CCA security based on the GKEA assumption, an assumption which is much weaker than the generic model itself (and instances of it are shown to be falsifiable [BP04]), and in fact, proving the equivalence of CPA and CCA security is trivial if the generic group model is used directly, since on a decryption query with a first element g^t, we may assume that t is known.

Chapter 5

Decentralized Dynamic Broadcast Encryption

Contents

Broadcast encryption (BE), introduced by Fiat and Naor [FN94] in 1993, allows a sender to securely send private messages to a subset of users, the target set. In 2001, Naor, Naor, and Lotspiech (NNL [NNL01]) introduced the subset-cover framework, where for any target set, the sender can find a partition of the user set, encrypt a session key using the keys associated to each subset in the partition, and finally encrypt the content using the session key. The ciphertext length of the subset-difference (SD) version of NNL depends linearly on the number of users in the revoked set, which was considered to be efficient enough for use in the AACS DRM standard [AAC09]. We generalize the subset-cover framework of NNL to deal with both public-key encryption and dynamic changes of the registered user sets. We furthermore remove the need for trusted authorities by eliminating the group manager, who typically interacts with users to distribute keys at the setup phase or when users join the system. Our approach makes use of group key exchange with subgroup keys [Man09, ACMP10], a primitive that simultaneously distributes different keys to certain subsets of the user group and applies well to the subset-cover framework if one can assign keys for the subgroups involved in the subset cover.

We first instantiate our construction with the Diffie-Hellman key agreement for the key generation and the ElGamal encryption for the public-key encryption, which leads to quite an efficient scheme. The complete-subtree (CS) tree construction resembles the tree-based group key agreement in [KPT04], with the exception that we also create key pairs for internal nodes, and we go beyond their scheme in our construction of SD trees. We then show how our scheme can be extended to achieve the strongest security notion by using Cramer-Shoup encryption, which allows adaptive corruptions and chosen-ciphertext attacks, in the standard model, under the DDH assumption. In addition, we consider various criteria of efficiency: ciphertext size, private part and public part of the decryption keys, number of rounds for the key generation, etc. Thanks to the modularity of our approach, we can use any appropriate group key exchange with subgroup keys: our initial technique iteratively uses the two-party Diffie-Hellman key exchange in a binary tree, which requires a logarithmic number of rounds; we can replace it by logarithmically many parallel executions of the Burmester-Desmedt group key exchange protocol [BD05],

which reduces the number of rounds to two. Besides allowing members to join the system, we also sketch how groups could merge at low cost, and how to permanently revoke some users. Our scheme thus achieves a maximum of functionality and security under minimal assumptions, while still being reasonably efficient.

Related work Broadcast encryption without a central authority replaces the traditional setup with a group key exchange process that can be an interactive protocol. It was proposed under the name "contributory broadcast encryption" (CBE) in [WQZ+11], along with a semi-adaptively IND-CPA secure scheme that is not dynamic. A possible application of this could be communication in a social network, where some private information is meant to be read only be a subset of a user's acquaintances, and the network is either peer-to-peer or the service provider is not trusted. The first steps toward subgroup key exchange were done by Manulis [Man09], who extended a group key exchange (GKE) protocol to allow any two users to compute a common key after the initial phase in which the group key is computed. Following this work, Abdalla et al. [ACMP10] generalized this approach to allow the computation of session keys for arbitrary subsets. We use such a group key exchange protocol with subgroup keys to derive asymmetric encryption keys for subsets. Something similar has been done under the name of "asymmetric group key agreement" (ASGKA) [WMS+09]. In [WMS+09], ASGKA is defined in a way that guarantees only that the keys held by the participants are good for use with a specific encryption scheme. We want to generalize this requirement so that at the end of the protocol run, each user has some randomness, which can thereafter be used for any key generation, and namely to generate key pairs for any key encapsulation mechanism. Since this randomness is shared between various subgroups, we call the scheme we use for the setup "subgroup key exchange" (SKE). Kurnio, Safavi-Naini, and Wang [KSNW03] explicitly consider sponsorship of group candidates by existing members. In our scheme, because of the tree structure, each user can act as a sponsor, and only one sponsor is required for a candidate to join the user set.

Contributions and organization In section 5.1, we define decentralized dynamic broadcast encryption and subgroup key exchange, a building block we use in our construction that may be of independent interest. We extend the security notions of adaptive IND-CPA and IND-CCA from chapter 4 to our case. We describe a black-box construction of decentralized dynamic broadcast encryption using the subset-cover framework in section 5.2 and prove the security of the construction, assuming that the building blocks are secure. In section 5.3, we construct a subgroup key exchange protocol based on any secure two-party key exchange protocol. We give two concrete instantiations using our methodology in section 5.4, that provide keys for subgroups in the CS and SD structures. Combined with the Cramer-Shoup encryption scheme, this gives us a decentralized dynamic broadcast encryption schemes which additionally achieves the highest security level (fully adaptive IND-CCA security) in the standard model under the DDH assumption.

5.1 Definitions

5.1.1 Decentralized Broadcast Encryption

Broadcast encryption generally involves a group manager, that deals with the membership of the users, and an encryptor that specifies the target group (a subgroup of the registered members) for a ciphertext. In secret-key broadcast encryption, the encryptor is a specific entity. This limitation is lifted in public-key broadcast encryption, which leaves only the group manager as a trusted entity. The group manager is either involved once only, at the setup phase, in static schemes, or at any time a new member wants to join the system, in dynamic schemes [DPP07]. The latter dynamic situation is the most realistic, but makes the group manager crucial for both security and availability. Our goal is to construct a scheme which does not rely on a single trusted authority.

We thus extend the dynamic broadcast encryption setting [DPP07] so that the membership management can be decentralized. At the same time, we would like to keep parameters such as ciphertext or key length as small as possible.

1. The ciphertext size should be as small as possible: the ciphertext has to contain the target group structure, and so cannot be smaller than the representation of this structure, which can either be encoded on N bits, where N is the total number of users, and each bit tells whether a user is in the target group or not, or on $r \log N$ bits (resp. $s \log N$ bits), where r (resp. s) is the number of revoked users (resp. included users) among the N registered users. This is sometimes considered

$\mathrm{Exp}^{\mathsf{corr}}_{\mathcal{DDBE}}(\kappa, U, V, \mathcal{S}, u^*)$
$\mathsf{param} \leftarrow \mathsf{Setup}(1^\kappa)$;
$(\mathsf{EK}, Reg, \{\mathsf{sk}_u\}_{u\in U}) \leftarrow \mathsf{KeyGen}(\mathsf{param}, U)$;
`for all` $v \in V \setminus U$: $(\mathsf{sk}_v, \mathsf{pt}_u) \leftarrow \mathsf{Join}(v, \{u(\mathsf{dk}_u)\}_{u\in U}, Reg, \mathsf{EK})$;
$(H, K) \leftarrow \mathsf{Encaps}(\mathsf{EK}, Reg, \mathcal{S})$;
`if` $\mathsf{Decaps}(\mathsf{dk}_{u^*}, \mathcal{S}, H) = K$ `then return` 1;
`else return` 0;

Figure 5.1: $\mathcal{DDBE}$: Correctness

independently from the ciphertext, in the header, but anyway both the target set and the encrypted data have to be sent. Our goal is to make the global length as small as possible.

2. When a new user joins the system, it should have minimal impact on other users' secret information and the public information: no impact at all on the keys as in [DPP07] is of course optimal, but when one wants to achieve forward secrecy, this is not possible: some of the keys have to be modified. We will try to keep the impact as small as possible too.

Since we want to avoid any centralized group manager, we will also focus on public-key broadcast encryption, in which a public key is enough to target any subgroup at the encryption time. In addition, instead of encrypting a message, our schemes will generate an encapsulation (or key header) and session keys to be used with any symmetric encryption scheme [Sho00].

Definition 5.1 (Decentralized dynamic broadcast encapsulation)**.** A decentralized dynamic broadcast encapsulation scheme (DDBE) with key space $\mathcal{K} = \{\mathcal{K}_\kappa\}_{\kappa\in\mathbb{N}}$ for a set of user identifiers $\mathcal{UI} \subset \{0,1\}^*$ is a 5-tuple of PPT algorithms and protocols $\mathcal{DDBE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Join}, \mathsf{Encaps}, \mathsf{Decaps})$:

- $\mathsf{Setup}(1^\kappa)$, where κ is the security parameter, generates the global parameters param of the system.
- $\mathsf{KeyGen}(\mathsf{param}, U)$ is an interactive protocol between the users in the set $U \subset \mathcal{UI}$. After the protocol run, it returns the public encryption key EK and a list Reg of the registered users with additional public information. Each user $u \in U$ eventually gets a secret decryption key dk_u.
- $\mathsf{Join}(v, \{u(\mathsf{dk}_u)\}_{u\in U}, Reg, \mathsf{EK})$ is an interactive protocol run between a user v and the set of users U, described in Reg. Each user takes as input his secret key and/or some random coins, the list Reg, and the encryption key EK. After the protocol, Reg and EK are updated, and each user (including v) has a secret decryption key.
- $\mathsf{Encaps}(\mathsf{EK}, Reg, S)$ takes as input the encryption key EK, the user register Reg, and a target set S. It outputs a key header H and a session key $K \in \mathcal{K}_\kappa$.
- $\mathsf{Decaps}(\mathsf{dk}_u, S, H)$ takes as input the target set S and a user decryption key dk_u together with a key header H. If dk_u corresponds to a recipient user, it outputs the session key K, else it outputs the error symbol $\perp$.

The correctness requirement is that for all (polynomial size) sets of users U, V with $U \subseteq V \subseteq \mathcal{UI}$, any target set $S \subset V$ and for any $u^* \in \mathcal{S}$, $\Pr[\mathrm{Exp}^{\mathsf{corr}}_{\mathcal{DDBE}}(\kappa, U, V, \mathcal{S}, u^*) = 1]$ is overwhelming in κ, where the experiment is defined in figure 5.1. A decentralized scheme requires that no authority is involved in the KeyGen and Join protocols.

Security notions We extend the strongest security notion from chapter 4 to the decentralized setting. The adversary is still given unlimited access to the Join oracle (dynamic), the $\mathsf{Corrupt}$ oracle (adaptive) and Decaps oracle (chosen-ciphertext security). For the group key generation, the definition from chapter 4 models passive adversaries only, since they only receive the public keys. Since in our case this group key generation may be an interactive protocol, we make it more explicit with a $\mathsf{Execute}$-oracle that outputs the public transcript of the full run of this protocol. The security game for DDBE is presented in figure 5.2: the restriction for the adversary is not to ask for the decapsulation of the challenge ciphertext (which includes the target set S) nor corrupt any user in the target set.

The adversary can ask once the generation of the group structure with a single call to $\mathsf{OExecute}$ on a group U of its choice, from which it gets the transcript τ, the encryption key EK and the register Reg. It can thereafter make as many calls it wants to OJoin, to add a user to the structure Reg, which updates EK. The adversary also gets the transcript τ of this interactive protocol. At any time, the adversary can

$\mathrm{Exp}^{\mathsf{ind\text{-}acca}\text{-}b}_{\mathcal{DDBE},\mathcal{A}}(\kappa)$
$\mathcal{Q}_C \leftarrow \emptyset$; $\mathcal{Q}_D \leftarrow \emptyset$;
$\mathsf{param} \leftarrow \mathsf{Setup}(1^\kappa)$;
$(st, U) \leftarrow \mathcal{A}(\mathsf{SETUP}; \mathsf{param})$;
$(\mathsf{EK}, Reg, \tau) \leftarrow \mathsf{OExecute}(U)$;
$(st, S) \leftarrow \mathcal{A}^{\mathsf{OJoin}(\cdot),\mathsf{OCorrupt}(\cdot),\mathsf{ODecaps}(\cdot,\cdot,\cdot)}(st; \mathsf{EK}, Reg, \tau)$;
$(H, K) \leftarrow \mathsf{Encaps}(\mathsf{EK}, Reg, S)$;
$K_b \leftarrow K$; $K_{1-b} \xleftarrow{\$} \mathcal{K}_\kappa$;
$b' \leftarrow \mathcal{A}^{\mathsf{OJoin}(\cdot),\mathsf{OCorrupt}(\cdot),\mathsf{ODecaps}(\cdot,\cdot,\cdot)}(st; H, K_0, K_1)$;
`if` $\exists i \in S, (i, S, H) \in \mathcal{Q}_D$ `or` $i \in \mathcal{Q}_C$;
`then return` 0
`else return` b';

$\mathsf{OExecute}(U)$
$(\mathsf{EK}, Reg, \{\mathsf{dk}_u\}) \leftarrow \mathsf{KeyGen}(\mathsf{param}, U)$;
`return` EK, Reg, τ;

$\mathsf{OJoin}(v)$
$(\mathsf{EK}, Reg, \{\mathsf{dk}_u\}) \leftarrow \mathsf{Join}(v, U, Reg, \mathsf{EK})$;
`return` EK, Reg, τ;

$\mathsf{OCorrupt}(u)$
$\mathcal{Q}_C \leftarrow \mathcal{Q}_C \cup \{u\}$; `return` dk_u;

$\mathsf{ODecaps}(u, S, H)$
$\mathcal{Q}_D \leftarrow \mathcal{Q}_D \cup \{(u, S, H)\}$;
$K \leftarrow \mathsf{Decaps}(\mathsf{dk}_u, S, H)$;
`return` K;

Figure 5.2: $\mathcal{DDBE}$: Key indistinguishability (IND-ACCA)

also corrupt a user with a key pair, calling OCorrupt and getting back all the secret information of the user, and decapsulate a ciphertext H, calling ODecaps in the name of a user u.

The main security goal of an encryption scheme (or an encapsulation scheme) is the indistinguishability of a challenge ciphertext: at some point, the adversary thus gets a challenge (H, K_0, K_1), where H encapsulates either K_0 or K_1 for a target set S chosen by the adversary. It has to guess which key is actually encapsulated. Of course, there are the natural restrictions, which are controlled granted the lists $\mathcal{Q}_C$ and $\mathcal{Q}_D$:

- (S, H) has not been asked to the decapsulation oracle for a user u in S
- none of the users in S have been corrupted

Definition 5.2. A decentralized dynamic broadcast encapsulation scheme $\mathcal{DDBE}$ is $(t, N, q_C, q_D, \varepsilon)$-IND-ACCA secure (security against adaptive corruption and chosen-ciphertext attacks) if for all $\kappa \in \mathbb{N}$, in the security game presented in figure 2.11, the advantage $\mathrm{Adv}^{\mathsf{ind\text{-}acca}}_{\mathcal{DDBE}}(\kappa, t, N, q_C, q_D)$ of any t-time adversary $\mathcal{A}$ creating at most N users (OJoin oracle), corrupting at most q_C of them (OCorrupt oracle), and asking for at most q_D decapsulation queries (ODecaps oracle), is bounded by ε:

$$\mathrm{Adv}^{\mathsf{ind\text{-}acca}}_{\mathcal{DDBE}}(\kappa, t, N, q_C, q_D) = \max_{\mathcal{A}}\{\Pr[\mathrm{Exp}^{\mathsf{ind\text{-}acca\text{-}1}}_{\mathcal{DDBE},\mathcal{A}}(\kappa) = 1] - \Pr[\mathrm{Exp}^{\mathsf{ind\text{-}acca\text{-}0}}_{\mathcal{DDBE},\mathcal{A}}(\kappa) = 1]\}.$$

This definition includes IND-ACPA (for adaptive chosen-plaintext attacks) when $q_D = 0$.

Remark 5.3 (Forward-secrecy)**.** *This definition includes forward secrecy against new users, i. e. a new user cannot decrypt ciphertexts that were created before he joined. For a definition without forward secrecy, the adversary is prohibited from corrupting users that joined after the challenge phase.*

5.1.2 Subgroup Key Exchange

The novelty of our definition is the decentralized key generation procedure, that should also generate keys for certain subgroups in order to be able to broadcast to any target set. This is thus in the same vein as the notion of group key exchange with on-demand computation of subgroup keys (GKE+S) from [ACMP10], that allows some subgroups of users to run a protocol to establish keys between them. But we extend this definition by allowing for keys of some subgroups to be computed during the first protocol run that establishes the global key, without any additional interaction.

Since we want to remain independent of the encryption scheme to be used with the session key, we require that for each subgroup a proto-key is computed, whose entropy can be used as input to a PKE key-pair generation, or to generate a symmetric encryption key.

Definition 5.4 (Dynamic $\mathcal{S}$-subgroup key exchange protocol)**.** For a collection $\mathcal{S} : \mathbb{N} \to \mathcal{P}(\mathcal{P}(\mathbb{N}))$ of subsets of the user set, where for any N, $\mathcal{S}(N) \subseteq \mathcal{P}([N])$, a dynamic $\mathcal{S}$-subgroup key exchange protocol $\mathcal{SKE}$ is a 3-tuple of PPT algorithms and interactive protocols:

- $\mathsf{Setup}(1^\kappa)$, where κ is the security parameter, generates the global parameters param of the system;
- $\mathsf{KeyGen}(\mathsf{param}, U)$ is an interactive protocol run between all users in U. It outputs a register Reg that contains a description of U and the subsets for which keys were established according to $\mathcal{S}$,

$\mathrm{Exp}^{\mathsf{ind}\text{-}b}_{\mathcal{SKE},\mathcal{A}}(\kappa)$
$Reg \leftarrow \emptyset$; $\mathcal{Q}_T \leftarrow \emptyset$;
$\mathsf{param} \leftarrow \mathsf{Setup}(1^\kappa)$;
$(state, U) \leftarrow \mathcal{A}(\mathsf{param})$;
$(Reg, \tau) \leftarrow \mathsf{OExecute}(U)$;
$b' \leftarrow \mathcal{A}^{\mathsf{OJoin}(\cdot),\mathsf{OTest}(\cdot,\cdot)}(state; \mathsf{EK}, Reg, \tau)$;
return b';

$\mathsf{OExecute}(U)$
$t \leftarrow 0$;
$(Reg, \{\mathsf{usk}_u\}_{u\in U}) \leftarrow \mathsf{KeyGen}(\mathsf{param}, U)$;
return Reg, τ;

$\mathsf{OTest}(t, S)$
if $\exists (t', K) \wedge t \equiv_S t' \wedge (t', S, K) \in \mathcal{Q}_T$
then return K;
elseif $b = 0$ then $K \leftarrow \mathsf{pt}_S(t)$;
else $K \xleftarrow{\$} \mathcal{K}$;
$\mathcal{Q}_T \leftarrow \mathcal{Q}_T \cup \{(t, S, K)\}$;
return K;

$\mathsf{OJoin}(v)$
$t \leftarrow t + 1$;
$(Reg, \{\mathsf{usk}_u\}_{u\in U}) \leftarrow \mathsf{Join}(v, U, Reg)$;
return Reg, τ;

Figure 5.3: $\mathcal{SKE}$: Key indistinguishability (IND)

and for each user $u \in U$ a secret usk_u that contains the proto-keys pt_S for all the sub-groups S containing u.
– $\mathsf{Join}(v, U, Reg)$ is an interactive protocol run between user v and the group of users U. It outputs an updated register Reg and for user v and some of the users in U a new secret usk_u that contains the proto-keys pt_S of all the subgroups S they are part of.

We require that all the users $u \in U$ that run $\mathsf{KeyGen}(\mathsf{param}, U)$ receive the same register Reg and compute matching proto-keys for the subsets they have in common. The same is required of Join.

For the security definition, we extend the definition given in [ACMP10], which seems to be most applicable to our case. Since the protocol is dynamic, the user set can change over time. As in the previous section, we stick to passive adversaries. This is a way of modularizing protocol construction, as passively secure protocols can be made secure against active adversaries using constructions such as [KY07], with additional authentication mechanisms.

The adversary can ask once the generation of the group structure with a unique call to $\mathsf{OExecute}$, at time $t = 0$, on a group U of its choice from which it gets the transcript τ and the register Reg. It can thereafter make as many calls as it wants to OJoin, to add a user to the structure Reg. Each query increases the time index t. The adversary also gets the transcripts τ of these interactive protocols.

The main security goal of key exchange is the indistinguishability of the keys from truly random keys, and their independence. Hence, we use the stronger notion proposed in [AFP05], similar to the real-or-random [BDJR97] for encryption. The adversary has access to many $\mathsf{OTest}(t, S)$ queries, that are either answered with the real key for set S at time t or with truly random and independent key. Note that according to the protocol, some keys may remain unchanged even when the time period evolves. We even hope to have as many keys as possible that do not evolve, since we want that not too many users are impacted by a new member in the system. We thus say that two pairs (t_1, S) and (t_2, S) are equivalent (denoted by $t_1 \equiv_S t_2$) if S is unchanged between the time periods and therefore they should have the same key. For such equivalent pairs, the same random key is output. We do not provide direct access to a $\mathsf{OReveal}$ oracle, which returns the secret key of a user, because as explained in [AFP05], having access to many OTest queries annihilates the advantage provided by $\mathsf{OReveal}$ queries.

Definition 5.5. A subgroup key exchange protocol $\mathcal{SKE}$ is (t, N, q_T, ε)-IND secure if for all $\kappa \in \mathbb{N}$, in the security game presented in figure 5.3, the advantage $\mathrm{Adv}^{\mathsf{ind}}_{\mathcal{SKE}}(\kappa, t, N, q_T)$ of any t-time adversary $\mathcal{A}$ creating at most N users (the final size of the set U) and testing at most q_T keys is bounded by ε:

$$\mathrm{Adv}^{\mathsf{ind}}_{\mathcal{SKE}}(\kappa, t, N, q_T) = \max_{\mathcal{A}}\{\Pr[\mathrm{Exp}^{\mathsf{ind}\text{-}1}_{\mathcal{SKE},\mathcal{A}}(\kappa) = 1] - \Pr[\mathrm{Exp}^{\mathsf{ind}\text{-}0}_{\mathcal{SKE},\mathcal{A}}(\kappa) = 1]\}.$$

5.2 Generic Decentralized Broadcast Encryption

As already remarked, in the first definition of dynamic broadcast encryption schemes [DPP07], it is required that the existing users are not affected by a join: their decryption keys should not be modified. Only the encryption key could be modified. This constraint is actually achieved by their scheme, but this is possible because the scheme is not forward secure: a new user can decrypt all ciphertexts that were sent before he joined (since he cannot be in any revoked set).

To achieve forward secrecy, we have to relax their definition and allow updates of the user decryption keys. Namely, updates of the decryption keys are necessary for forward secrecy in the subset-cover framework [NNL01], because some keys are shared by several users. With an appropriate subset-cover structure, it can reach asymptotically optimal overall ciphertext size. On the other hand, the naive scheme, where each user has a single key specific to him, can be made dynamic without decryption key updates, but has ciphertexts whose length is linear in the number of users. As soon as keys are shared between users, forward secrecy makes it necessary to update these shared keys. Hence our relaxation of the model. However, we require these updates of existing keys to be made via public channels.

5.2.1 Generic Public-Key Subset Cover

A subset-cover structure $\mathcal{SC} = \{S_i\}_{i\in I}$ is a set of subsets S_i of a user set U such that for any subset $S \subset U$ there is a subset $\mathcal{L} \subset I$ such that S can be *partitioned* as $S = \bigcup_{i\in\mathcal{L}} S_i$. In particular, this implies that for all users $u \in U$, $\{u\} \in \mathcal{SC}$. In [NNL01], a secret key is assigned to each set S_i, so a message can be encrypted to any subset $S \subset U$ by finding the cover $\mathcal{L}$ of S. Then a session key is encrypted under all the keys associated to the selected subsets. All the other users are then implicitly revoked, since they cannot decrypt the session key. Because of the partition property, a user in S is in one subset S_i only. Efficiency will thereafter depend on the subset-cover structure.

We extend this framework in three directions:

1. First, we transfer this approach to the public-key world. Each S_i is assigned a key pair of some PKE scheme by some key assignment procedure. This means that the assignment of keys to the subsets depends on the PKE scheme used as well as the assignment procedure. For example, for a subset-cover structure $\mathcal{SC}$ and a PKE $\mathcal{PKE}$, we can use the key assignment that assigns each subset with a key pair drawn independently at random by the trusted centre.
2. Second, we replace the trusted centre by an interactive protocol, a subgroup key exchange.
3. Third, we allow for the addition of users, hence using a *dynamic subgroup key exchange* to generate the keys for a dynamic subset-cover structure.

We first deal with a dynamic subset-cover structure, assuming a subgroup key exchange as a black box. Thereafter, we will consider concrete structures and efficient subgroup key exchanges.

5.2.2 Dynamic Subset-Cover

We define a dynamic subset-cover as a sequence of subset-covers $\{\mathcal{SC}_i\}$ for $i \geq 0$ users, where each $\mathcal{SC}_i$ contains subsets S_j. These subsets never change, so instead of adding a user to a subset, we remove the old one and add a new one. This also means that the same subset S_j can occur in different time periods (the time period changes each time a new user joins). We start with $\mathcal{SC}_0 = \emptyset$ and an empty user set $U_0 = \emptyset$, and then have $U_{n+1} = U_n \cup \{u_{n+1}\}$. From the definition, it is clear that $|U_n| = n$, and w.l.o.g. $U_n = [n]$.

For subset-cover based dynamic broadcast encryption, we will have to generate the keys for all the subsets that are involved in $\mathcal{SC}_n$. The following property will optimize efficiency, in the sense that a minimal number of existing users will be impacted by a new member.

Definition 5.6 (Splitting property)**.** We say that a dynamic subset cover $\mathcal{SC}$ has the *splitting property*, if the subset cover at time $n+1$ is composed of subsets that either were part of the subset cover at time n, or contain the new user. $\mathcal{SC}_{n+1} = \mathcal{SC}'_{n+1} \cup \mathcal{SC}''_{n+1}$, where $\mathcal{SC}'_{n+1} \subset \mathcal{SC}_n$ and $S_i \in \mathcal{SC}''_{n+1} \Rightarrow u_{n+1} \in S_i$.

With this property, if a subset changes, it is either removed, or it contains u_{n+1}. Then only sets with the new user need new key generation, which is a minimal requirement anyway.

5.2.3 SC-based Decentralized Dynamic Broadcast Encryption

We first assume we have a dynamic subgroup key exchange $\mathcal{SKE}$ that is compatible with our dynamic subset-cover structure, which means that for any n, the subgroup key exchange provides keys for all the subsets S in $\mathcal{SC}_n$. We will later instantiate such a dynamic subgroup key exchange for some dynamic subset-cover structures.

Let us recall that the SC-based broadcast encryption [NNL01] consists in encrypting the same message under the keys of all the subsets that cover the target set. Since one of our goals is to achieve the highest security level, adaptive chosen-ciphertext security, any modification of the description of the target set

or one of the ciphertexts in the list should make the global ciphertext invalid, otherwise the scheme is somewhat malleable, and thus insecure against chosen-ciphertext attacks. We will add a MAC to bind the target header and the ciphertexts together. A similar approach has been used by [BK05, DK05]. Instead of a master secret key, our scheme needs only a public register *Reg* to keep track of the users currently enrolled in the system and their public keys.

We first present in details our construction, and then state the security of the construction. It is important to remember that the subgroup key exchange scheme is only assumed to be passively secure, meaning that the protocol requires authenticated channels. This can be achieved in several ways that we will not discuss here. Because the subset cover is a fixed part of the protocol and defines the subsets for each number of users, and we assume that the number of users in the system is always known, the number of a new user and the subsets he belongs to can be computed deterministically by all users. Meta-issues like trust between users and how they should agree on which users to allow into the group are beyond the scope of this work.

Definition 5.7 (dBE)**.** Let $\mathcal{PKE}$ be a PKE, $\mathcal{MAC}$ a MAC, $\mathcal{F} : \mathcal{K} \to \mathcal{R}$ a pseudo-random generator, $\mathcal{SC}$ a dynamic subset-cover with the splitting property, and $\mathcal{SKE}$ a dynamic subgroup key exchange compatible with $\mathcal{SC}$ with keys in $\mathcal{K}$. Our broadcast encryption scheme is defined as follows.

- $\mathsf{Setup}(1^\kappa)$:
  1. Run $\mathcal{PKE}.\mathsf{Setup}(1^\kappa)$ to get $\mathsf{param}_{\mathcal{PKE}}$;
  2. Run $\mathcal{SKE}.\mathsf{Setup}(1^\kappa)$ to get $\mathsf{param}_{\mathcal{SKE}}$;
  3. Publish $\mathsf{param} = (\mathsf{param}_{\mathcal{PKE}}, \mathsf{param}_{\mathcal{SKE}})$.
- $\mathsf{KeyGen}(\mathsf{param}, U_n)$, for some integer n:
  1. Run $\mathcal{SKE}.\mathsf{KeyGen}(\mathsf{param}_{\mathcal{SKE}}, U_n)$ to get *Reg*; Each user $u \in U_n$ gets as output of the protocol the proto-keys pt_S for all subsets S he belongs to according to $\mathcal{SC}$. The decryption key dk_u consists of all these pt_S.
  2. Each user computes $(\mathsf{dk}_S, \mathsf{ek}_S) \leftarrow \mathcal{PKE}.\mathsf{KeyGen}(\mathsf{param}_{\mathcal{PKE}}; \mathcal{F}(\mathsf{pt}_S))$, where we use the PRG to generate from the proto-key the random coins of the key generation algorithm;
  3. All the encryption keys ek_S are published as EK;
  4. The decryption keys dk_S can be either stored in dk_u for users $u \in S$, or deleted since they can be recomputed;
- $\mathsf{Join}(v, \{u(\mathsf{dk}_u)\}_{u \in U_n}, Reg, \mathsf{EK})$:
  1. Run $\mathcal{SKE}.\mathsf{Join}(v, \{u(\mathsf{dk}_u)\}_{u \in U_n}, Reg)$ to get the new *Reg*; User v and each user $u \in U_n$ gets as output of the protocol the proto-keys pt_S for all subsets S that contain v;
  2. Each user u does as above to compute the updates dk_S, ek_S and dk_u;
  3. All the encryption keys ek_S are added to EK, old encryption keys are deleted.
- $\mathsf{Encaps}(\mathsf{EK}, Reg, S)$:
  1. From the target set S, generate the partition $\mathcal{L}$ with $S = \cup_{\mathcal{L}} S_i$;
  2. Generate a random session key $\mathcal{K}_e$ and a random MAC key $\mathcal{K}_m$;
  3. For each subset $i \in \mathcal{L}$, generate $c_i = \mathcal{PKE}.\mathsf{Encrypt}(\mathsf{ek}_{S_i}, \mathcal{K}_e || \mathcal{K}_m)$;
  4. Compute $\sigma = \mathcal{MAC}.\mathsf{GenMac}(\mathcal{K}_m, S || (c_i)_{i \in \mathcal{L}})$;
  5. Output $\mathcal{K}_e$ and $H = ((c_i)_{i \in \mathcal{L}}, \sigma)$.
- $\mathsf{Decaps}(\mathsf{dk}_u, S, H)$:
  1. Parse H as $((c_i)_{i \in \mathcal{L}}, \sigma)$);
  2. If $u \in S$, then there is a unique i such that $u \in S_i$, and then dk_u allows to derive dk_{S_i};
  3. Extract $\mathcal{K}_e || \mathcal{K}_m = \mathcal{PKE}.\mathsf{Decrypt}(\mathsf{dk}_{S_i}, c_i)$;
  4. Check if $\mathcal{MAC}.\mathsf{VerifMac}(\mathcal{K}_m, S || (c_i)_{i \in \mathcal{L}}, \sigma)$;
  5. In case of validity, output $\mathcal{K}_e$, otherwise output $\perp$.

The scheme is a correct dynamic broadcast encryption scheme, because of the correctness of the basic primitives $\mathcal{PKE}$, $\mathcal{MAC}$ and $\mathcal{F}$, but also $\mathcal{SKE}$.

Theorem 5.8. *Let us consider the scheme $\mathcal{BE}^{\mathcal{PKE},\mathcal{MAC},\mathcal{F},\mathcal{SKE}}$ from definition 5.7. We define L_N to be the total number of distinct subsets over all time periods and ℓ_N to be the maximal number of subsets necessary to cover any authorized target set S in $\mathcal{SC}_i$ for any i. If $\mathcal{PKE}$ is an IND-CCA secure PKE, $\mathcal{MAC}$ is a SUF-CMA secure MAC, $\mathcal{SKE}$ is a IND secure SKE, and $\mathcal{F}$ is a pseudo-random generator, then this scheme is a forward secure IND-ACCA secure BE scheme:*

$$\begin{aligned}\mathrm{Adv}^{\mathsf{ind\text{-}acca}}_{\mathcal{DDBE}}(\kappa,t,N,q_C,q_D) \leq\ & 2\mathrm{Adv}^{\mathsf{ind}}_{\mathcal{SKE}}(\kappa,t,L_N,L_N) + 3\ell_N L_N \mathrm{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{PKE}}(\kappa,t,q_D)\\ &+ 2L_N\mathrm{Adv}^{\mathsf{prg}}_{\mathcal{F}}(\kappa,t) + 2\mathsf{Succ}^{\mathsf{suf\text{-}cma}}_{\mathcal{MAC}}(\kappa,t,1,q_D).\end{aligned}$$

The variables L_N and ℓ_N depend on the type of subset cover used in the scheme. For CS, L_N is less than $N\log N$ (since at most $\log N$ sets change in each of the at most N steps), and ℓ_N is $r\log\frac{N}{r}$, which is bounded by $N/2$ (the worst-case ciphertext length). For SD, we have $L_N \leq N\log^2 N$ and $\ell_N = 2r-1$.

Proof. We assume that $\mathcal{A}$ is an adversary against the IND-ACCA security game. We define a sequence of games, $\mathbf{G}_0, \ldots, \mathbf{G}_9$, where $\mathbf{G}_0$ is the IND-ACCA experiment with $b=0$ and $\mathbf{G}_9$ is the IND-ACCA experiment with $b=1$. Let ℓ be the number of components in a challenge ciphertext (the size of the partition $\mathcal{L}^*$ of the challenge target set S^*). By definition, ℓ is not greater than ℓ_N.

Game $\mathbf{G}_0$: This is the IND-ACCA-Experiment with $b=0$. We just recall the generation of the challenge ciphertext (the Encaps oracle), and the simulation of the ODecaps oracle:

$\mathsf{Setup}(1^\kappa)$:

1. Run $\mathcal{PKE}.\mathsf{Setup}(1^\kappa)$ to get $\mathsf{param}_{\mathcal{PKE}}$;
2. Run $\mathcal{SKE}.\mathsf{Setup}(1^\kappa)$ to get $\mathsf{param}_{\mathcal{SKE}}$;
3. Publish $\mathsf{param} = (\mathsf{param}_{\mathcal{PKE}}, \mathsf{param}_{\mathcal{SKE}})$.

$\mathsf{KeyGen}(\mathsf{param}, U_n)$:

1. All the proto-keys pt_S, for all the subsets S in $\mathcal{SC}_n$, are generated using the $\mathcal{SKE}.\mathsf{KeyGen}$ protocol;
2. Each user $u \in U_n$ gets the proto-keys for all subsets S he belongs to. The decryption key dk_u consists of all these pt_S;
3. He computes $(\mathsf{dk}_S, \mathsf{ek}_S) \leftarrow \mathcal{PKE}.\mathsf{KeyGen}(\mathsf{param}_{\mathcal{PKE}}; \mathcal{F}(\mathsf{pt}_S))$, where we use the PRG to generate the random coins of the key generation algorithm;
4. The adversary receives the transcript of the execution of the $\mathcal{SKE}.\mathsf{KeyGen}$ protocol.

$\mathsf{Join}(v, \{u(\mathsf{dk}_u)\}_{u\in U_n}, Reg, \mathsf{EK})$: similar to KeyGen

$\mathsf{Encaps}(\mathsf{EK}, Reg, \mathcal{S}^*)$:

1. From the target set S^*, generate a partition $S^* = \cup_{\mathcal{L}^*} S_i$, we assume of size ℓ;
2. Generate two session keys $\mathcal{K}_e^0$ and $\mathcal{K}_e^1$, as well as a MAC key $\mathcal{K}_m^0$;
3. For each subset $i \in \mathcal{L}^*$, generate $c_i^* = \mathcal{PKE}.\mathsf{Encrypt}(\mathsf{ek}_{S_i}, \mathcal{K}_e^0 || \mathcal{K}_m^0)$;
4. Then, compute $\sigma^* = \mathcal{MAC}.\mathsf{GenMac}(\mathcal{K}_m^0, S^* || (c_i^*)_{i\in\mathcal{L}^*})$;
5. Outputs $\mathcal{K}_e^0$, $\mathcal{K}_e^1$ and $H^* = ((c_i^*)_{i\in\mathcal{L}^*}, \sigma^*)$.

$\mathsf{ODecaps}(u, S, H)$:

1. If u is in S, then there is a unique i such that $u \in S_i$, and then dk_u allows to derive dk_{S_i};
2. Extract $\mathcal{K}_e || \mathcal{K}_m = \mathcal{PKE}.\mathsf{Decrypt}(\mathsf{dk}_{S_i}, c_i)$;
3. If $i \in \mathcal{L}^*$ and $c_i = c_i^*$, check if $\mathcal{MAC}.\mathsf{VerifMac}(\mathcal{K}_m, S || (c_i)_{i\in\mathcal{L}}, \sigma)$;
4. Else, check if $\mathcal{MAC}.\mathsf{VerifMac}(\mathcal{K}_m, S || (c_i)_{i\in\mathcal{L}}, \sigma)$;
5. In case of validity, output $\mathcal{K}_e$, otherwise output $\perp$.

Game $\mathbf{G}_1$: We first replace all the proto-keys by random keys: We thus apply the key indistinguishability of the $\mathcal{SKE}$ scheme:

> $\mathsf{KeyGen}(\mathsf{param}, U_n)$:
>
> 1. All the proto-keys pt_S are drawn independently at random for all subsets S;

The difference between $\mathbf{G}_1$ and $\mathbf{G}_0$ is bounded by

$$\Pr_1[\mathcal{A} \to 1] - \Pr_0[\mathcal{A} \to 1] \leq \mathrm{Adv}^{\mathsf{ind}}_{\mathcal{SKE}}(\kappa, t, L_N, L_N).$$

Game $\mathbf{G}_2$: We now replace all PKE keys by random keys: we thus apply the pseudo-randomness of the PRG $\mathcal{F}$:

> $\mathsf{KeyGen}(\mathsf{param}, U_n)$:
>
> 3. Each user gets $(\mathsf{dk}_S, \mathsf{ek}_S) \leftarrow \mathcal{PKE}.\mathsf{KeyGen}(\mathsf{param}_{\mathcal{PKE}}; r_S)$, where r_S are random coins, for all subsets S he belongs to;

Using a classical hybrid proof, the difference between $\mathbf{G}_2$ and $\mathbf{G}_1$ is bounded by

$$\Pr_2[\mathcal{A} \to 1] - \Pr_1[\mathcal{A} \to 1] \leq L_N \times \mathrm{Adv}^{\mathsf{prg}}_{\mathcal{F}}(\kappa, t).$$

Game $\mathbf{G}_3$: We introduce an additional MAC key that will be used later in the sub-ciphertexts:

> $\mathsf{Encaps}(\mathsf{EK}, Reg, S^*)$:
>
> 2. Generate two session keys $\mathcal{K}_e^0$ and $\mathcal{K}_e^1$, as well as two MAC keys $\mathcal{K}_m^0$ and $\mathcal{K}_m^1$;

$\mathbf{G}_3$ and $\mathbf{G}_2$ are perfectly indistinguishable:

$$\Pr_3[\mathcal{A} \to 1] = \Pr_2[\mathcal{A} \to 1].$$

Game $\mathbf{G}_4$: We now use the additional MAC key $\mathcal{K}_m^1$ in the challenge sub-ciphertexts, but still use $\mathcal{K}_m^0$ for the MAC computation:

> $\mathsf{Encaps}(\mathsf{EK}, Reg, S^*)$:
>
> 3. For each subset $i \in \mathcal{L}^*$, generate $c_i^* = \mathcal{PKE}.\mathsf{Encrypt}(\mathsf{ek}_i, \mathcal{K}_e^0 || \mathcal{K}_m^1)$;

Lemma 5.9. *The difference between $\mathbf{G}_4$ and $\mathbf{G}_3$ is bounded by*

$$\Pr_4[\mathcal{A} \to 1] - \Pr_3[\mathcal{A} \to 1] \leq \ell \times L_N \times \mathrm{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{PKE}}(\kappa, t, q_D).$$

Game $\mathbf{G}_5$: In this game, we reject decryption queries that should decrypt a sub-ciphertext from the challenge ciphertext.

> $\mathsf{ODecaps}(u, S, H = ((c_i)_{i \in \mathcal{L}}, \sigma))$:
>
> 3. If $i \in \mathcal{L}^*$ and $c_i = c_i^*$, output $\perp$;

Lemma 5.10. *The difference between $\mathbf{G}_5$ and $\mathbf{G}_4$ is bounded by*

$$\Pr_5[\mathcal{A} \to 1] - \Pr_4[\mathcal{A} \to 1] \leq \mathsf{Succ}^{\mathsf{suf\text{-}cma}}_{\mathcal{MAC}}(\kappa, t, 1, q_D).$$

Game $\mathbf{G}_6$: We define the game $\mathbf{G}_6$ as the game $\mathbf{G}_5$, but we encapsulate $\mathcal{K}_e^1$ instead of $\mathcal{K}_e^0$:

> $\mathsf{Encaps}(\mathsf{EK}, Reg, S^*)$:
>
> 3. For each subset $i \in \mathcal{L}^*$, generate $c_i^* = \mathcal{PKE}.\mathsf{Encrypt}(\mathsf{ek}_{S_i}, \mathcal{K}_e^1 || \mathcal{K}_m^1)$;

Lemma 5.11. *The difference between $\mathbf{G}_6$ and $\mathbf{G}_5$ is bounded by*

$$\Pr_6[\mathcal{A} \to 1] - \Pr_5[\mathcal{A} \to 1] \leq \ell \times L_N \times \mathrm{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{PKE}}(\kappa, t, q_D).$$

Game $\mathbf{G}_7$: Previous game is similar to $\mathbf{G}_5$, but with $\mathcal{K}_e^1$ in the challenge ciphertext. We now go back, as in game $\mathbf{G}_4$: we check MAC values of sub-ciphertexts of the challenge ciphertext under $\mathcal{K}_m^0$:

$\mathsf{ODecaps}(u, S, H)$:
3. If $i \in \mathcal{L}^*$ and $c_i = c_i^*$, check if $\mathcal{MAC}.\mathsf{VerifMac}(\mathcal{K}_m, S||(c_i)_{i\in\mathcal{L}}, \sigma)$.

Since we have the same gap as from $\mathbf{G}_4$ to $\mathbf{G}_5$:

$$\Pr_7[\mathcal{A} \to 1] - \Pr_6[\mathcal{A} \to 1] \leq \mathsf{Succ}^{\mathsf{suf\text{-}cma}}_{\mathcal{MAC}}(\kappa, t, 1, q_D).$$

Game $\mathbf{G}_8$: We eventually change back the use of the MAC key $\mathcal{K}_m^0$ in the challenge sub-ciphertexts:

$\mathsf{Encaps}(\mathsf{EK}, Reg, S^*)$:
3. For each subset $i \in \mathcal{L}^*$, generate $c_i^* = \mathcal{PKE}.\mathsf{Encrypt}(\mathsf{ek}_{S_i}, \mathcal{K}_e^1||\mathcal{K}_m^0)$;

Since we have the same gap as from $\mathbf{G}_3$ to $\mathbf{G}_4$:

$$\Pr_8[\mathcal{A} \to 1] - \Pr_7[\mathcal{A} \to 1] \leq \ell \times L_N \times \mathrm{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{PKE}}(\kappa, t, q_D).$$

We do not use anymore the key $\mathcal{K}_m^1$: this is exactly the IND-ACCA security game with $b = 1$, except for the generation of the encryption keys.

Game $\mathbf{G}_9$: We now change back the generation of the encryption keys, using the $\mathcal{SKE}$ protocol and the PRG:

$$\Pr_9[\mathcal{A} \to 1] - \Pr_8[\mathcal{A} \to 1] \leq \mathrm{Adv}^{\mathsf{ind}}_{\mathcal{SKE}}(\kappa, t, L_N, L_N) + L_N \times \mathrm{Adv}^{\mathsf{prg}}_{\mathcal{F}}(\kappa, t).$$

If we sum up all the gaps, we obtain:

$$\begin{aligned}
\Pr_1[\mathcal{A} \to 1] - \Pr_0[\mathcal{A} \to 1] &\leq \mathrm{Adv}^{\mathsf{ind}}_{\mathcal{SKE}}(\kappa, t, L_N, L_N)\\
\Pr_2[\mathcal{A} \to 1] - \Pr_1[\mathcal{A} \to 1] &\leq L_N \times \mathrm{Adv}^{\mathsf{prg}}_{\mathcal{F}}(\kappa, t)\\
\Pr_4[\mathcal{A} \to 1] - \Pr_2[\mathcal{A} \to 1] &\leq \ell \times L_N \times \mathrm{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{PKE}}(\kappa, t, q_D)\\
\Pr_5[\mathcal{A} \to 1] - \Pr_4[\mathcal{A} \to 1] &\leq \mathsf{Succ}^{\mathsf{suf\text{-}cma}}_{\mathcal{MAC}}(\kappa, t, 1, q_D)\\
\Pr_6[\mathcal{A} \to 1] - \Pr_5[\mathcal{A} \to 1] &\leq \ell \times L_N \times \mathrm{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{E}}(\kappa, t, q_D)\\
\Pr_7[\mathcal{A} \to 1] - \Pr_6[\mathcal{A} \to 1] &\leq \mathsf{Succ}^{\mathsf{suf\text{-}cma}}_{\mathcal{MAC}}(\kappa, t, 1, q_D)\\
\Pr_8[\mathcal{A} \to 1] - \Pr_7[\mathcal{A} \to 1] &\leq \ell \times L_N \times \mathrm{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{PKE}}(\kappa, t, q_D)\\
\Pr_9[\mathcal{A} \to 1] - \Pr_8[\mathcal{A} \to 1] &\leq \mathrm{Adv}^{\mathsf{ind}}_{\mathcal{SKE}}(\kappa, t, L_N, L_N) + L_N \times \mathrm{Adv}^{\mathsf{prg}}_{\mathcal{F}}(\kappa, t)
\end{aligned}$$

And this concludes the proof, since $\ell \leq \ell_N$. □

Proof of Lemma 5.9.

Let ℓ be the size of the partition $\mathcal{L}^*$. In order to so show that the adversary cannot detect whether we use the same MAC key that is part of the ciphertext or not, we proceed by another sequence of hybrid games: We define the game G_j (for $j = 0, \ldots, \ell$), in which the j-first sub-ciphertexts c_i^* are defined as in $\mathbf{G}_4$, that is $c_i^* = \mathcal{PKE}.\mathsf{Encrypt}(\mathsf{ek}_i, \mathcal{K}_e^0||\mathcal{K}_m^1)$, and the next ones are defined as in $\mathbf{G}_3$, that is $c_i^* = \mathcal{PKE}.\mathsf{Encrypt}(\mathsf{ek}_{S_i}, \mathcal{K}_e^0||\mathcal{K}_m^0)$. It is clear that $G_0 = \mathbf{G}_3$, whereas $G_\ell = \mathbf{G}_4$.

For any $J \in [0, \ell]$, let us play the following game against the IND-CCA challenger of the $\mathcal{PKE}$ encryption scheme:

– Setup/KeyGen:
  1. We receive the challenge public key ek;
  2. We randomly choose one subset $I \in [1, L_N]$ (we bet it will correspond to the J-th ciphertext in the target partition $\mathcal{L}^*$. This guess is correct with probability $1/L_N$, otherwise we abort the game and make $\mathcal{B}$ output 0);
  3. We generate all the pairs $(\mathsf{dk}_{S_i}, \mathsf{ek}_{S_i})$ at random, except for $i = I$, where $\mathsf{ek}_{S_I} = \mathsf{ek}$;
– $\mathsf{Encaps}(\mathsf{EK}, Reg, S^*)$:
  1. From the target set S^*, generate a partition $S^* = \cup_{\mathcal{L}^*} S_i$, we assume of size ℓ;

2. If our guess at setup time was correct, the J-th element in $\mathcal{L}^*$ is I;
3. Generate two session keys $\mathcal{K}_e^0$ and $\mathcal{K}_e^1$, as well as two MAC keys $\mathcal{K}_m^0$ and $\mathcal{K}_m^1$;
4. For the $J-1$-first elements $i \in \mathcal{L}^*$, generate $c_i^* = \mathcal{PKE}.\mathsf{Encrypt}(\mathsf{ek}_{S_i}, \mathcal{K}_e^0||\mathcal{K}_m^1)$;
5. For the J-th element $i \in \mathcal{L}^*$, assumed to be I, ask to the IND-CCA-challenger on the two plaintexts $\mathcal{K}_e^0||\mathcal{K}_m^0$ and $\mathcal{K}_e^0||\mathcal{K}_m^1$, and set c_I^* to be the answer, according to the internal bit b of the IND-CCA challenger;
6. For the next elements $i \in \mathcal{L}^*$, generate $c_i^* = \mathcal{PKE}.\mathsf{Encrypt}(\mathsf{ek}_{S_i}, \mathcal{K}_e^0||\mathcal{K}_m^0)$;
7. Then, compute $\sigma^* = \mathcal{MAC}.\mathsf{GenMac}(\mathcal{K}_m^0, S^*||(c_i^*)_{i\in\mathcal{L}^*})$;
8. Output $\mathcal{K}_e^0$, $\mathcal{K}_e^1$ and $H^* = ((c_i^*)_{i\in\mathcal{L}^*}, \sigma^*)$.

– OCorrupt Queries: Since we condition on the good choice for I, we can answer all the OCorrupt queries by outputting the corresponding decryption keys (they cannot be for I, otherwise the challenge target set would contain corrupted players);
– ODecaps Queries:
1. If this is for a player that lies in a set $S_i \notin S_I$, we can easily decrypt c_i;
2. If this is for a player that lies in S_I,
   – either $c_I \neq c_I^*$, and then we can ask the decryption query to the decryption oracle
   – or $c_I = c_I^*$, then check the MAC value with $\mathcal{K}_m^0$. If it is valid, output $\mathcal{K}_e^0$, otherwise output $\perp$.

Our adversary $\mathcal{B}$ against IND-CCA simply forwards the output b' of $\mathcal{A}$ (or outputs zero in case of abort):

$$\begin{aligned}\Pr[\mathcal{B}\to 1|b=0] - \Pr[\mathcal{B}\to 1|b=1] &= \Pr[\mathcal{B}\to 1\wedge I|b=0] - \Pr[\mathcal{B}\to 1\wedge I|b=1]\\ &\quad + \Pr[\mathcal{B}\to 1\wedge \neg I|b=0] - \Pr[\mathcal{B}\to 1\wedge \neg I|b=1]\\ &= \frac{1}{L_N}\times(\Pr[\mathcal{B}\to 1|b=0\wedge I] - \Pr[\mathcal{B}\to 1|b=1\wedge I])\\ &= \frac{1}{L_N}\times(\Pr[\mathcal{A}\to 1|b=0\wedge I] - \Pr[\mathcal{A}\to 1|b=1\wedge I])\,.\end{aligned}$$

In the RHS, the output is independent of the correct guess of I, whereas the LHS is bounded by the best advantage against IND-CCA within time t:

$$\frac{1}{L_N}\times|\Pr[\mathcal{A}\to 1|b=0] - \Pr[\mathcal{A}\to 1|b=1]| \leq \mathrm{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{PKE}}(\kappa,t,q_D).$$

Furthermore, the above game with $b=0$ is exactly G_{J-1}, whereas with $b=1$ this is G_J:

$$\left|\Pr_{G_{J-1}}[\mathcal{A}\to 1] - \Pr_{G_J}[\mathcal{A}\to 1]\right| \leq L_N\times\mathrm{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{PKE}}(\kappa,t,q_D).$$

This concludes the proof.

Proof of Lemma 5.10.

In order to show that the adversary cannot detect whether we reject a valid MAC under the unknown key $\mathcal{K}_m^0$, we use the following game against the MAC: more precisely, we play the SUF-CMA security game against the MAC, using the challenge MAC key sk as the unknown $\mathcal{K}_m^0$ key. All the other keys are known to the simulator. The MAC generation oracle OGenMac is called for the challenge MAC value by the Encaps oracle, and the MAC verification oracle OVerifMac is called in case of a challenge sub-ciphertext in a decapsulation ODecaps oracle query. A valid MAC value asked to OVerifMac is a forgery, otherwise it should be a reject. Hence, the probability that a valid MAC value is refused is bounded by $\mathsf{Succ}^{\mathsf{suf\text{-}cma}}_{\mathcal{MAC}}(\kappa,t,1,q_D)$.

Proof of Lemma 5.11.

Let ℓ be the size of the partition $\mathcal{L}^*$. In order to so show that the adversary cannot detect whether we encrypt $\mathcal{K}_e^0$ or $\mathcal{K}_e^1$, we proceed as for the proof of lemma 5.9, by a sequence of hybrid games: We define the game G_j (for $j=0,\ldots,\ell$), in which the j-first sub-ciphertexts c_i^* are defined as in $\mathbf{G}_6$, that is $c_i^* = \mathcal{PKE}.\mathsf{Encrypt}(\mathsf{ek}_i, \mathcal{K}_e^1||\mathcal{K}_m^1)$, and the next ones are defined as in $\mathbf{G}_5$, that is $c_i^* = \mathcal{PKE}.\mathsf{Encrypt}(\mathsf{ek}_i, \mathcal{K}_e^0||\mathcal{K}_m^1)$. It is clear that $G_0 = \mathbf{G}_5$, whereas $G_\ell = \mathbf{G}_6$. Exactly the same analysis as in the proof of lemma 5.9 leads to the result. The trick comes from the simulation of ODecaps Queries, in which we output $\perp$ in case a sub-ciphertext of the challenge ciphertext is involved. We do not have to care whether this is $\mathcal{K}_e^0$ or $\mathcal{K}_e^1$.

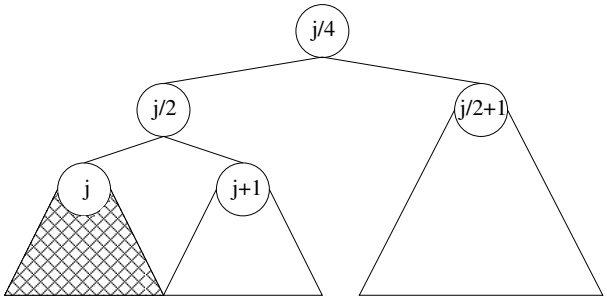

Figure 5.4: SD key assignment for $S_{j/4,j}$

5.3 Tree-based Subgroup Key Exchange

In this section, we define two subgroup key exchange protocols compatible with the efficient tree-based methods defined in [NNL01]. The tree-based methods are special cases in the subset-cover framework, where the users are organized as leaves in a binary tree, and the subsets S_i can be described in terms of subtrees of this tree.

Complete subtree We first review the complete subtree (CS) structure for N users $\{u_0, \ldots, u_{N-1}\}$. For simplicity, we assume $N = 2^d$, but the description can be generalized to any N. All the users are leaves of the tree, and can be seen as singletons $S_{2^d+i} = \{u_i\}$, for $i = 0, \ldots, 2^d - 1$. Then, for $i = 2^d - 1$ to 1, $S_i = S_{2i} \cup S_{2i+1}$ which contains all the leaves below the node with index i.

Subset difference The subset difference (SD) method uses subsets $S_{i,j} = S_i \setminus S_j$, where S_i, S_j are defined as in the CS method, and S_j is a subtree of S_i. All sets S_i from the CS tree are also contained in the SD method, because $S_i = S_{\text{parent}(i),\text{sibling}(i)}$; S_0 is included as a special set.

5.3.1 Static Tree Construction

Let us show how such subset-cover structures naturally give rise to subgroup key exchange protocols. The main tools for our construction of the subgroup key exchange are two primitives: a 2-party key exchange protocol $\mathcal{KE}$ that outputs keys in $\mathcal{K}_{\mathcal{KE}}$ and a pseudo-random generator $\mathcal{G} : \mathcal{K}_{\mathcal{KE}} \to \mathcal{K} \times \mathcal{R}_{\mathcal{KE}}$.

Two users start from random coins in $\mathcal{R}_{\mathcal{KE}}$, and run a key exchange protocol $\mathcal{KE}.\mathsf{CommonKey}$ in order to derive a secret value ck for the subset represented by the node in the tree that is their parent. This common key ck is used as the seed for the PRG $\mathcal{G}$ to derive the two secret keys, the proto-key $\mathsf{pt} \in \mathcal{K}$ and the random coins $r \in R_{\mathcal{KE}}$ for the next key exchange at the level above. Internal nodes thus involve "virtual" users. In summary, the tree is constructed by executing $\mathcal{KE}.\mathsf{CommonKey}$, then computing $\mathcal{G}$, at each level from the bottom up. We derive generic instantiations of the complete subtree (CS) and subset difference (SD) methods on binary trees described in [NNL01].

CS tree We define the neighbour of user u with identifier i to be the user u' with identifier $i + 1$ if $i \equiv 0 \mod 2$, $i - 1$ else and its parent to be the user w with identifier $\lfloor i/2 \rfloor$. At round r, each (virtual) user u created in round $r - 1$ has a uniquely defined neighbour u' and a parent w. If he does not, the protocol run is completed: we are either at the root of the tree, or the tree is not complete. The users u and u' have random coins r_u and $r_{u'}$, which they use to run the $\mathcal{KE}$ protocol, resulting in a common key ck_w. From this common key, they derive the proto-key of node w and the randomness for the virtual user w to participate in the next round of key exchanges. The user with the smaller identifier then plays the role of the virtual user w in the next round. As a consequence, for N users, there are $\log N$ rounds. Round r involves $N/2^{r-1}$ (virtual) users.

- $\mathsf{KeyGen}(U_n)$: In round r, for $r = 1, \ldots, \log n$, the users u, u' with parent w at level $(\log n - r)$ proceed as follows:
  1. $\mathsf{ck}_w \leftarrow \mathcal{KE}.\mathsf{CommonKey}(\mathsf{sk}_u, u')$;
  2. $(\mathsf{pt}_w, r_w) \leftarrow \mathcal{G}(\mathsf{ck}_w)$;
  3. $(\mathsf{pk}_w, \mathsf{sk}_w) \leftarrow \mathsf{KeyGen}(\mathsf{param}; r_w)$;
  4. if $u < u'$, set $u \stackrel{\mathsf{def}}{=} w$;

SD tree We can modify the construction of the above CS tree to obtain keys for any subset $S_{i,j} = S_i \setminus S_j$, when $S_j \subset S_i$: To exclude all leaves below node j (w.l.o.g. $j \equiv 0 \mod 4$), we skip the key exchange between j and $j+1$ and directly compute a common key $\mathsf{ck}_{j/2+1,j+1}$ between $j/2+1$ (j's uncle) and $j+1$ (j's sibling). Basically, we identify the public key of $j+1$ with that of $S_{j/2,j}$. After applying $\mathcal{G}$, we have the two key pairs for $S_{j/4,j}$, which we treat as being at node $j/4$. We then continue with $\mathcal{KE}.\mathsf{CommonKey}$ and $\mathcal{G}$ as normal up to node i, to get $S_{i,j}$. This allows us to construct an SD tree in much the same way as a CS tree, except that we "omit" one node in the computation of the key (see figure 5.4). Each node i at depth ℓ ($2^\ell \leq i < 2^{\ell+1}$) contains $2^{d-\ell+1} - 4$ blocks of keys that can be computed iteratively, excluding all the possible subtrees, from depth $\ell+2$ (4 of them) to d ($2^{d-\ell}$ of them).

We define the neighbour and parent of user i as for the CS scheme, and the neighbour of (i,j) to be the neighbour of i. At round r, each user (i,j) created in round $r-1$ has a uniquely defined neighbour i' (if he does not, the protocol run is completed). They both have key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$ and $(\mathsf{pk}_{i'}, \mathsf{sk}_{i'})$:

- First, i computes the keys of $i/2$ as for the CS tree, running $\mathsf{ck}_{i/2} \leftarrow \mathcal{KE}.\mathsf{CommonKey}(\mathsf{sk}_i, i')$ and deriving $(\mathsf{pt}_{i/2}, r_{i/2}) \leftarrow \mathcal{G}(\mathsf{ck}_{i/2}); (\mathsf{pk}_{i/2}, \mathsf{sk}_{i/2}) \leftarrow \mathcal{KE}.\mathsf{KeyGen}(\mathsf{param}; r_{i/2})$.
- Next, he sets $(\mathsf{sk}_{(i/2,i')}, \mathsf{pk}_{(i/2,i')}) \stackrel{\mathsf{def}}{=} (\mathsf{pk}_i, \mathsf{sk}_i)$, which omits i' in the computation.
- For all his identities (i,j), i runs $\mathsf{ck}_{(i/2,j)} \leftarrow \mathcal{KE}.\mathsf{CommonKey}(\mathsf{sk}_{(i,j)}, i'))$ and sets $(\mathsf{pt}_{(i/2,j)}, r_{(i/2,j)}) \leftarrow \mathcal{G}(\mathsf{ck}_{(i/2,j)}); (\mathsf{pk}_{(i/2,j)}, \mathsf{sk}_{(i/2,j)}) \leftarrow \mathsf{KeyGen}(\mathsf{param}; r_{(i/2,j)})$ to derive the information for the parent node $(i/2, j)$.

If $i < i'$, it plays the role of the virtual user $i/2$ in the next round.

5.3.2 Dynamic Tree Construction

Dynamic CS We define a join procedure for the CS tree described above. We go from $\mathcal{SC}_n$ to $\mathcal{SC}_{n+1}$ by taking the leaf u' with the lowest distance to the root, and if there are several with that property, the one with the lowest index. We then replace it with an inner node w, to which we append both the leaf u' and the new user v. We note that the user identifiers will not be in the same order as the node numbers in the tree. Then we replace the subsets S_j where j is an ancestor of the new user with the new subsets. This ensures that our dynamic CS scheme is forward secure and has the splitting property of definition 5.6. The CS key assignment is done as follows.

First the new user v derives a common key c_w with its sibling u'. From this common key, he derives the proto-key of node w and the randomness for the virtual user w to participate in the next round of key exchanges. The user with the smaller identifier then plays the role of w in the next round. This procedure is repeated until the keys of all ancestors of v are recomputed.

- $\mathsf{Join}(v, U_n)$ In the first round, set $u \stackrel{\mathsf{def}}{=} v$. In round r, for $r = 1, \ldots, \log(n+1)$, the user u with neighbour u' and parent w at level $(\log(n+1) - r)$ proceeds as follows:
  1. $\mathsf{ck}_w \leftarrow \mathcal{KE}.\mathsf{CommonKey}(\mathsf{sk}_u, u')$;
  2. $(\mathsf{pt}_w, r_w) \leftarrow \mathcal{G}(\mathsf{ck}_w)$;
  3. $(\mathsf{pk}_w, \mathsf{sk}_w) \leftarrow \mathsf{KeyGen}(\mathsf{param}; r_w)$;
  4. set $u \stackrel{\mathsf{def}}{=} w$, $u' \stackrel{\mathsf{def}}{=} \mathit{neighbour}(w)$, $w \stackrel{\mathsf{def}}{=} \mathit{parent}(w)$;

Dynamic SD To join a user, we go from $\mathcal{SC}_n$ to $\mathcal{SC}_{n+1}$ by appending the new user as described for CS. Then we replace those subsets $S_{i,j}$ that contain the new user with the new subsets.

We show that our dynamic SD scheme has the splitting property of definition 5.6. All $S_{i,j}$ for which i is not an ancestor of the new node are unchanged. All $S_{i,j}$ for which i, but not j is an ancestor of the new node contain the new user. All $S_{i,j}$ for which i is an ancestor of the new node and j is a true ancestor of the new node are unchanged as well. All $S_{i,j}$ for which i is an ancestor and j is the new node correspond to full subtrees $S_{\mathrm{parent}(i),\mathrm{sibling}(i)}$ in the old subset cover. The key assignment for SD is similar to the CS key assignment, but we cannot identify nodes and subsets, and must "jump" the omitted subtree in the computation (figure 5.4).

We state exactly the security of the dynamic CS construction. Because of the similarities in the construction, a similar result can be obtained for SD.

Theorem 5.12. *Let $\mathcal{KE}$ be an IND secure KE scheme with session keys in $\mathcal{K}_{\mathcal{KE}}$, and $\mathcal{G} : \mathcal{K}_{\mathcal{KE}} \to \mathcal{K} \times \mathcal{R}_{\mathcal{KE}}$ be a PRG. Then our dynamic CS construction of a $\mathcal{SKE}$ is IND secure and*

$$\mathrm{Adv}^{\mathsf{ind}}_{\mathcal{SKE}}(\kappa, t, N, q_T) \leq (N \log N) \left(\mathrm{Adv}^{\mathsf{ind}}_{\mathcal{KE}}(\kappa, t, N, N-1, q_T) + \mathrm{Adv}^{\mathsf{prg}}_{\mathcal{G}}(\kappa, t) \right).$$

Proof. Let $\mathcal{A}$ be an adversary against the IND security of our CS construction $\mathcal{SKE}$ that invokes at most N users, among them the user set U_M that runs the KeyGen protocol (where $M = |U_M|$ denotes its size) and T users that join once at a time, in T time steps. Since in each of the T time periods at most $\log N$ nodes are updated, at most $N \log N$ keys will be generated overall (for $M = 1$). Game $\mathrm{Exp}^{\mathsf{ind\text{-}0}}_{\mathcal{SKE},\mathcal{A}}(\kappa)$ is the experiment where all keys are generated as usual. This will be our initial game. Game $\mathrm{Exp}^{\mathsf{ind\text{-}1}}_{\mathcal{SKE},\mathcal{A}}(\kappa)$ is the experiment where all keys are chosen uniformly at random. This will be our final game. To go from the first game to the final one, we define intermediate games, in which we first replace the session keys that are produced by the two-player key exchange protocols by random keys, and then we replace the proto-keys by random keys.

Game $\mathbf{G}_0$: This is the initial game, that appears in the experiment where $b = 0$.

> $\mathsf{KeyGen}(U_M)$: In round r, for $r = \log M, \ldots, 1$, the simulator executes the following steps for each node w at level $(\log M - r)$ of the tree with children u, u':
>
> 1. $\mathsf{ck}_w \leftarrow \mathcal{KE}.\mathsf{CommonKey}(\mathsf{sk}_u, u')$;
> 2. $(\mathsf{pt}_w, r_w) \leftarrow \mathcal{G}(\mathsf{ck}_w)$;
> 3. $(\mathsf{pk}_w, \mathsf{sk}_w) \leftarrow \mathsf{KeyGen}(\mathsf{param}; r_w)$;
> 4. If $u < u'$, set $u \stackrel{\mathsf{def}}{=} w$.

> $\mathsf{Join}(v, U_n)$ In the first round, set $u = v$. In round r, for $r = \log(n+1), \ldots, 1$, the simulator executes the following steps for user u with neighbour u' and parent w at level $(\log(n+1) - r)$:
>
> 1. $\mathsf{ck}_w \leftarrow \mathcal{KE}.\mathsf{CommonKey}(\mathsf{sk}_u, u')$;
> 2. $(\mathsf{pt}_w, r_w) \leftarrow \mathcal{G}(\mathsf{ck}_w)$;
> 3. $(\mathsf{pk}_w, \mathsf{sk}_w) \leftarrow \mathsf{KeyGen}(\mathsf{param}; r_w)$;
> 4. set $u \stackrel{\mathsf{def}}{=} w$, $u' \stackrel{\mathsf{def}}{=} \mathit{neighbour}(w)$, $w \stackrel{\mathsf{def}}{=} \mathit{parent}(w)$;

Game $\mathbf{G}_1$: We replace all KE session keys on level 1 of the tree with random keys.

> $\mathsf{KeyGen}(U_M)$: In round 1, the simulator executes the following steps for each node w at level 1 of the tree with children u, u':
>
> 1. $\mathsf{ck}_w \leftarrow \mathcal{KE}.\mathsf{CommonKey}(\mathsf{sk}_u, u')$; $\mathsf{ck}_w \stackrel{\$}{\leftarrow} \mathcal{K}_{\mathcal{KE}}$;

With a classical hybrid proof, where we successively replace all the real keys by random keys in the $M/2$ two-party key exchanges, we get that the difference between $\mathbf{G}_1$ and $\mathbf{G}_0$ is bounded by

$$\Pr_1[\mathcal{A} \to 1] - \Pr_0[\mathcal{A} \to 1] \leq \frac{M}{2} \mathrm{Adv}^{\mathsf{ind}}_{\mathcal{KE}}(\kappa, t, M, M-1, q_T).$$

Game $\mathbf{G}_2$: We replace all proto-keys on level 1 of the tree with random keys.

> $\mathsf{KeyGen}(U_M)$: In round 1, the simulator executes the following steps for each node w at level 1 of the tree with children u, u':
>
> 2. $(\mathsf{pt}_w, r_w) \stackrel{\$}{\leftarrow} \mathcal{K} \times \mathcal{R}_{\mathcal{KE}}$;

With a classical hybrid proof, where we successively replace all the real values by random values in the $M/2$ key derivations, we get that the difference between $\mathbf{G}_2$ and $\mathbf{G}_1$ is bounded by

$$\Pr_2[\mathcal{A} \to 1] - \Pr_1[\mathcal{A} \to 1] \leq \frac{M}{2} \mathrm{Adv}^{\mathsf{prg}}_{\mathcal{G}}(\kappa, t).$$

Game $\mathbf{G}_3$: We replace all proto-keys in the initial tree with random keys.

> $\mathsf{KeyGen}(U_M)$: In round r, the simulator executes the following steps for each node w at level r of the tree with children u, u':
>
> 1. $\mathsf{ck}_w \leftarrow \mathcal{KE}.\mathsf{CommonKey}(\mathsf{sk}_u, u')$; $\mathsf{ck}_w \stackrel{\$}{\leftarrow} \mathcal{K}_{\mathcal{KE}}$;
> 2. $(\mathsf{pt}_j, r_j) \stackrel{\$}{\leftarrow} \mathcal{K} \times \mathcal{R}_{\mathcal{KE}}$;

By applying iteratively the 2 previous hops at level 2 on $M/2^2$ pairs, and at level 3 on $M/2^3$ pairs, etc, we get that the difference between $\mathbf{G}_3$ and $\mathbf{G}_2$ is bounded by

$$\Pr_3[\mathcal{A} \to 1] - \Pr_2[\mathcal{A} \to 1] \leq (\frac{M}{2} - 1) \left(\mathrm{Adv}^{\mathsf{ind}}_{\mathcal{KE}}(\kappa, t, M, M-1, q_T) + \mathrm{Adv}^{\mathsf{prg}}_{\mathcal{G}}(\kappa, t) \right).$$

Game $\mathbf{G}_4$: We replace all proto-keys created during joins with random keys. The result is a protocol where all proto-keys are drawn independently at random, which describes the experiment with $b = 1$.

$\mathsf{Join}(v, U_n)$

1. $\mathsf{ck}_w \leftarrow \mathcal{KE}.\mathsf{CommonKey}(\mathsf{sk}_u, u')$; $\mathsf{ck}_w \xleftarrow{\$} \mathcal{K}_{\mathcal{KE}}$;
2. $(\mathsf{pt}_j, r_j) \xleftarrow{\$} \mathcal{K} \times \mathcal{R}_{\mathcal{KE}}$;

Lemma 5.13. *The difference between $\boldsymbol{G}_4$ and $\boldsymbol{G}_3$ is bounded by*

$$\Pr_4[\mathcal{A} \to 1] - \Pr_3[\mathcal{A} \to 1] \leq (T \log N) \left(\mathrm{Adv}^{\mathsf{ind}}_{\mathcal{KE}}(\kappa, t, N, N-1, q_T) + \mathrm{Adv}^{\mathsf{prg}}_{\mathcal{G}}(\kappa, t) \right).$$

In summary, we have

$$\Pr_3[\mathcal{A} \to 1] - \Pr_0[\mathcal{A} \to 1] \leq (M-1) \left(\mathrm{Adv}^{\mathsf{ind}}_{\mathcal{KE}}(\kappa, t, M, M-1, q_T) + \mathrm{Adv}^{\mathsf{prg}}_{\mathcal{G}}(\kappa, t) \right)$$
$$\Pr_4[\mathcal{A} \to 1] - \Pr_3[\mathcal{A} \to 1] \leq (T \log N) \left(\mathrm{Adv}^{\mathsf{ind}}_{\mathcal{KE}}(\kappa, t, N, N-1, q_T) + \mathrm{Adv}^{\mathsf{prg}}_{\mathcal{G}}(\kappa, t) \right).$$

Because $M + T = N$, we obtain

$$\mathrm{Adv}^{\mathsf{ind}}_{\mathcal{SKE}}(\kappa, t, N, q_T) \leq (N \log N) \left(\mathrm{Adv}^{\mathsf{ind}}_{\mathcal{KE}}(\kappa, t, N, N-1, q_T) + \mathrm{Adv}^{\mathsf{prg}}_{\mathcal{G}}(\kappa, t) \right).$$

Note that this is independent of q_T, because we change all the keys. □

Proof of Lemma 5.13.

Let G_0 be the game $\mathbf{G}_3$, G_T be the game $\mathbf{G}_4$. We define $T-1$ intermediate hybrid games G_j ($j = 1 \ldots T-1$), in which we replace all session keys and proto-keys computed during the first j joins with random keys. We proceed as in the previous proofs and obtain

$$\Pr_j[\mathcal{A} \to 1] - \Pr_{j-1}[\mathcal{A} \to 1] \leq \log N \cdot \left(\mathrm{Adv}^{\mathsf{ind}}_{\mathcal{KE}}(\kappa, t, N, N-1, q_T) + \mathrm{Adv}^{\mathsf{prg}}_{\mathcal{G}}(\kappa, t) \right).$$

5.4 Concrete Instantiations

We now give two instantiations of our scheme. The first one is probably the simplest possible case, and achieves IND-ACPA security under the DDH assumption. We use the Diffie-Hellman protocol [DH76] as our KE (where the users publish g^x and g^y from their random coins x and y, and get g^{xy} as common key) and ElGamal [Elg85] as the PKE where $\mathsf{ek} = g^{\mathsf{dk}}$, for a random scalar dk). A similar idea can be found in [KPT04], where the authors use a group key exchange protocol on a DH-tree. Because the random coin spaces of both protocols are identical, when we run both in the same group $\mathbb{G}$ of order q (scalars in $\mathbb{Z}_q$), if we only want to prove IND-ACPA security, we can identify dk with the random coins for the key exchange, and thus ek is part of the transcript of the key exchange protocol, leaving us with a single key pair for both schemes. There are several alternatives for the PRG, the simplest one being a hash function modeled by a random oracle, to extract $\mathsf{dk} \in \mathbb{Z}_q$ from the proto-key $\mathsf{pt} \in G$. But we can avoid it, and even any computational assumption, by using a deterministic randomness extractor, as described in [CFGP06, th. 7], that is a bijection and thus a perfect generator (see definition 2.25):

Definition 5.14. If $p = 2q + 1$, and $\mathbb{G}$ is defined as the sub-group of the squares in $\mathbb{Z}_p^*$, then $\mathrm{ord}(\mathbb{G}) = q$ and f is a bijection from $\mathbb{G}$ onto $\mathbb{Z}_q$: $f(x) = x$ (`if` $x \leq q$) or $p - x$ (`if` $x > q$).

The second instantiation is more involved. To achieve IND-ACCA security, we use Cramer-Shoup encryption [CS98] as our PKE. Because the keys in Cramer-Shoup are larger, our KE is a 3-to-8 parallel Diffie-Hellman, where we use public and private keys consisting of three elements each to generate a

shared key consisting of eight elements, which allows us to generate additional pseudo-randomness in each step. Our PRG is an embedding function $\mathbb{G}^8 \to \mathbb{Z}_q^3 \times \mathbb{Z}_q^5$ that applies the above function f to all components. The first part in $\mathbb{Z}_q^3$ will be used again as random coins for the key exchange, whereas the second part in $\mathbb{Z}_q^5$ leads to the Cramer-Shoup decryption key. To counter malleability of our scheme, we also need a SUF-CMA secure MAC scheme. As the first scheme, this one relies only on the DDH assumption.

When using the Cramer-Shoup PKE, the decryption key of node i is the tuple $\mathsf{dk}_i = (v_i, w_i, x_i, y_i, z_i)$, the corresponding encryption key ek_i is $(X_i, Y_i, H_i) = (g^{x_i}h^{v_i}, g^{y_i}h^{w_i}, g^{z_i})$. We need to generate more pseudo-randomness than before, so we define a new key exchange that is essentially a parallel Diffie-Hellman.

We define a modified Diffie-Hellman key exchange scheme.

Definition 5.15 (3-8-DHKE)**.** Let $\mathbf{G}$ be a generator of cyclic groups.

- $\mathsf{Setup}(1^\kappa)$ runs $\mathbf{G}(1^\kappa)$ to obtain $(q, \mathbb{G})$, draws $g \xleftarrow{\$} \mathsf{Gens}(\mathbb{G})$ and outputs $\mathsf{param} = (g, q)$.
- $\mathsf{KeyGen}(g, q)$ draws random elements $a, b, c \xleftarrow{\$} \mathbb{Z}_q^*$ and outputs $(\mathsf{pk}, \mathsf{sk}) = ((g^a, g^b, g^c), (a, b, c))$.
- $\mathsf{CommonKey}((a_j, b_j, c_j), (A_i, B_i, C_i))$ outputs $\mathsf{ck} = (A_i^{a_j}, A_i^{b_j}, A_i^{c_j}, B_i^{a_j}, B_i^{b_j}, B_i^{c_j}, C_i^{a_j}, C_i^{b_j})$.

Its security is based on a modified DDH problem. We omit the proof, which is analogous to the proof of security of the standard Diffie-Hellman key exchange (DHKE) from DDH.

Definition 5.16 (3-8-DDH)**.** Let $\mathbf{G}$ be a generator of cyclic groups. 3-8-$\mathsf{DDH}_\mathbf{G}$ is the problem of, for $\kappa \in \mathbb{N}$, $(q, \mathbb{G}) \in [\mathbf{G}(1^\kappa)], g \xleftarrow{\$} \mathsf{Gens}(\mathbb{G}), a, b, c, a', b', c' \xleftarrow{\$} \mathbb{Z}_q$, distinguishing between the 15-tuple $(g, g^a, g^b, g^c, g^{a'}, g^{b'}, g^{c'}, g^{aa'}, g^{ab'}, g^{ac'}, g^{ba'}, g^{bb'}, g^{bc'}, g^{ca'}, g^{cb'})$ and a 15-tuple of random elements in $\mathbb{G}$.

The hardness of this problem is related directly to the hardness of the standard DDH problem in $\mathbf{G}$.

Theorem 5.17. $\mathrm{Adv}^{\text{3-8-}ddh}_{\mathbf{G}}(\kappa, t) \leq 8 \cdot \mathrm{Adv}^{ddh}_{\mathbf{G}}(\kappa, t + 11\tau_{exp})$*, where* τ_{exp} *is the time for an exponentiation.*

Proof. We define tuple T_0 to be the tuple as defined above, T_i as the same tuple with all "combined" elements up to the i-th one replaced by a random element. T_8 is therefore a tuple of 15 random elements. Given a distinguisher $\mathcal{A}$ between T_i and T_{i+1}, we construct a solver $\mathcal{B}$ for DDH as follows. Let $(X, Y, Z) = (g^x, g^y, g^z)$ be a DDH challenge tuple. Let $g^{de'}$ be the $i+1$-st combined element. $\mathcal{B}$ chooses a tuple T_i and replaces g^d with X, $g^{e'}$ with Y, and $g^{de'}$ with Y. All other combined elements can be constructed because at least one exponent is known, which takes 11 exponentiations ($11\tau_{exp}$) time. If $z = xy$, $T' = T_i$, else $T' = T_{i+1}$ and the theorem follows. □

As a PRG we use the PRG of definition 5.14 on each component of the common key. This gives us all the components we need to construct an IND-ACCA secure BE scheme, whose security is based only on the DDH assumption. (The DDH assumption implies the existence of OWF, which is sufficient for MACs.)

5.4.1 Efficiency Properties

One of the main advantages of the NNL constructions [NNL01] is the efficient revocation with small ciphertext lengths ($\mathcal{O}(r \log N/r)$ for CS, $\mathcal{O}(2r-1)$ for SD) which is immediately inherited in our public-key scheme. The decryption key is the same length for CS, where each user has to store $\log N$ keys only, and longer ($\mathcal{O}(N \log N)$ for SD), where we cannot use the same key derivation.

In our scheme, for many instantiations of the 2-party key exchange, the private part of the decryption key can even be constant-size: each user keeps his secret random coins r_i, which is enough to iteratively generate all the private information from the public transcript of the key exchange protocols (stored in *Reg* or in the public key). Then, granted the key exchange scheme and $\log N$ public keys, each user can iteratively compute the decryption keys along the path to the root of the tree, and it is in this sense that the user random coins "contain" the keys used to decrypt, as required by the decapsulation algorithm.

Permanent revocation Because the length of the ciphertext for SC schemes depends on the number of revoked users, it is desirable to be able to completely remove users from a group. To permanently remove a user at leaf $2i$, we remove it and its sibling leaf $2i+1$ and simply move the user at $2i+1$ to be at node i which becomes a leaf. The keys of the user now at i remain the same as his own key before (at node $2i+1$) and we thus have to update the keys of all subsets in which the revoked user was a member.

Concerning the security, it is easy to see that the user $2i$, not having the key of the user $2i+1$, can not learn anything about the updated keys, and this ensures the forward secrecy.

The only problem we face is that we need to keep the tree balanced. Fortunately, our constructions allow a re-organization of the tree in a very efficient manner. Indeed, the tree could be maintained to be an AVL tree at low cost [AVL62]. Whenever a user leaves the system and makes the tree unbalanced, by using $\log N$ rotations, we can re-balance the tree. Note that a rotation needs $\log N$ update operations at worst, so the total cost for a re-balancing is just $\log^2 N$ update operations at worst.

Merging groups Instead of joining a single user, we can also efficiently merge two existing groups by executing the key exchange protocol for their root nodes. This will allow every user in the two groups to compute the keys of the new root node.

Constant-round key generation While this construction achieves constant-size secrets for the users and requires very little interaction during the Join-procedure, it requires a logarithmic number of rounds for the subgroup key exchange protocol to complete. The Burmester-Desmedt group key exchange protocol (def 2.51) is, like the above scheme, passively secure in the standard model under the DDH assumption. It requires only two rounds, and several instances could be run in parallel to compute keys for all subsets in two rounds. This would however require interaction between all the users each time a new users wants to join.

Chapter 6

Message-based Traitor Tracing

Contents

Classical traitor tracing techniques start from a pirate decoder that is able to decrypt protected messages and find the users who contributed keys to this decoder. This approach is fundamentally unable to deal with users which, instead of contributing keys to a pirate decoder, distribute the decrypted content. Schemes which use hybrid encryption can be targeted even more efficiently, since it is sufficient to redistribute the symmetric keys which decrypt the message block and are the same for all users. One way to protect against the redistribution of content is to produce two watermarked versions of each message block and let each user decrypt only one of the two. This way, each user decrypts a message with a specific sequence of watermarks, but this doubles the length of the ciphertext.

Our goal is to improve the efficiency of this technique, so that each user ends up with a message containing a specific sequence of watermarks, but without doubling all the blocks. As a starting point, we present in section 6.1.1 the simplest scheme, where we have for each message block m_i two equivalent blocks m_i^0 and m_i^1, so that any sequence $\{m_i^{w_i}\}$, whatever $w \in \{0,1\}^n$ is, corresponds to a valid message m. The two versions m_i^0 and m_i^1 can be provided by either adding watermarks to the original message block m_i, or directly, e. g. by recording a movie with different angles or distances of the shots [BS98]. The blocks, m_i^0 or m_i^1, are both sent over the public channel. However, the user secret keys, usk_i^0 or usk_i^1, have been distributed to the users according to codewords in a fingerprinting code. This means when the authority sees the decoded message m' or the symmetric keys, from each block m_i', it can tell whether it is m_i^0, m_i^1, or the block has been dropped, and then learns which decryption key has been used: usk_i^0, usk_i^1, or none. From this, it can derive one bit of a word: 0, 1, or 'erasure' respectively. Thanks to the collusion-resistance of the code with erasures, if not too many traitors colluded, at least one of them can be traced back. This method thus consists of encrypting each pair of blocks with two keys. Each user owns only one of the two according to the codeword he received from a fingerprinting code. This results in a ciphertext twice the length of the original message, plus the cost for two key encapsulations per block.

The only way to reduce the length of the encrypted payload is to protect only a few blocks, not all of them. We present our basic construction in section 6.1.2. It resists collusions, but only if the adversary has to output complete messages, i. e. he is not allowed to drop blocks.

If an adversary can detect which blocks are protected because he has access to several different user keys, he can drop some of them without impacting too much the quality of the original message (i. e. a few seconds from a movie). If the adversary can specifically drop protected blocks after decryption,

the output contains less information about the keys that were used, which prevents tracing. We thus propose in section 6.1.3 an extension of fingerprinting codes to solve this problem so that the adversary cannot specifically erase bits of the codeword: even if we protect 1% of the blocks and the adversary drops 20% of the blocks, he will only drop 20% of the protected blocks, and not all of them.

This scheme still suffers from long user keys, as we need two key pairs for each message block. In section 6.2.1, we use anonymous broadcast encryption as a primitive instead of PKE to achieve shorter key lengths. We first focus on the two-user case (one message block), which we later extend to cover N users. A message block either consists of a unique message m_i (not protected) or of two versions m_i^0 and m_i^1: in the former case, m_i should be encrypted for the two users, whereas in the latter case, m_i^0 has to be encrypted for user 0, and m_i^1 for user 1. To this aim, we use a 2-user anonymous broadcast encryption scheme (2ABE) that we construct in section 6.2.3. Anonymous broadcast encryption allows the selection of any subset of the user set that should be able to decrypt the ciphertext, while hiding who is able to decrypt (sec 2.6.2). Suppose we have a 2ABE scheme, and we consider ℓ blocks $(m_1, \ldots, m_\ell)$, among which only the k-th block is protected and thus provided as a pair (m_k^0, m_k^1). We encrypt all the unique blocks m_i for both users, whereas we encrypt m_k^0 for user 0, and m_k^1 for user 1. The ciphertexts are thereafter randomly permuted (but we assume that the message blocks contain indices to reorder them). User 0 and user 1 will both be able to decrypt ℓ ciphertexts among the $\ell+1$, and after reordering will be able to get the original message. Due to the anonymity, they do not know which block the other user cannot decrypt, therefore they have no idea which block is protected.

The encrypted payload is only $(1+1/\ell)$-times as long as the original message, plus the cost of 2ABE key encapsulations, but we need only one key for each user, which is the minimum, given that when viewing the decrypted message, the authority can extract one bit. To achieve full tracing, we extend this case to an arbitrary number of users by allocating the user secret keys for the 2ABE using our extension of a fingerprinting code (sec. 2.7.3).

6.1 A Generic Construction from PKE

In this section, we present a series of three simple constructions that illuminate the concepts behind the construction in the next section. The first construction exemplifies a well-known paradigm of constructing a message-based traitor tracing scheme from a PKE scheme and a fingerprinting code, and serves as a stepping stone for the second construction, which shows how to achieve optimal ciphertext rate. The first two constructions are only secure when the adversary is required to retransmit only complete messages. Our third construction modifies the previous one to account for adversaries that drop parts of the message.

6.1.1 A Simple Construction

To have a baseline against which to compare our later constructions, we first outline a simple way to construct a message-based traitor tracing scheme. The construction uses a PKE scheme Π and assigns user keys according to the codewords of the fingerprinting code $\mathcal{T}$. If the codewords have length n, we need $2n$ instances of the PKE scheme.

- $\mathsf{Setup}(1^\kappa, N, t, \varepsilon)$
  1. Generate a t-fingerprinting code, using $(\Gamma, \mathsf{tk}) \leftarrow \mathcal{T}.\mathsf{Gen}(N, \varepsilon)$, with a low error value ε. We let n denote the length of a codeword in Γ, and enumerate codewords with indices associated to each users: $\Gamma = \{w_{\mathsf{id}}\}_{\mathsf{id}=1,\ldots,N} \subset \{0,1\}^n$.
  2. Call $2n$ times $\Pi.\mathsf{Setup}(1^\kappa)$ to obtain $(\mathsf{dk}_i^b, \mathsf{ek}_i^b)_{b=0,1,i=1\ldots n}$.
  3. Set $\mathsf{EK} \leftarrow \{\mathsf{ek}_i^b\}_{b=0,1,i=1\ldots n}$, $\mathsf{USK}_{\mathsf{id}} \leftarrow (\mathsf{dk}_i^{w_{\mathsf{id}}[i]})_{i=1\ldots n}$ for all $\mathsf{id} \in [N]$, $\mathsf{TK} \leftarrow (\{\mathsf{dk}_i^0, \mathsf{dk}_i^1\}_{i=1,\ldots,n}, \mathsf{tk})$.
- $\mathsf{Encrypt}(\mathsf{EK}, m)$ First split m into n blocks $m_1, \ldots m_n$. For each block,
  1. Create two versions m_i^0, m_i^1 of each block m_i
  2. Encrypt the versions as $c_i^b = \Pi.\mathsf{Encrypt}(\mathsf{ek}_i^b, m_i^b)$
  3. Set $c = (c_1^0, c_1^1, \ldots, c_n^0, c_n^1)$.
- $\mathsf{Decrypt}(\mathsf{USK}_{\mathsf{id}}, c)$
  1. Parse $c = (c_1^0, c_1^1, \ldots, c_n^0, c_n^1)$
  2. For $i = 1, \ldots, n$, decrypt $m_i^{w_{\mathsf{id}}[i]} = \Pi.\mathsf{Decrypt}(\mathsf{dk}_i^{w_{\mathsf{id}}[i]}, c_i^{w_{\mathsf{id}}[i]})$. Output $m \stackrel{\mathsf{def}}{=} m_1^{w_{\mathsf{id}}[1]} || \ldots || m_n^{w_{\mathsf{id}}[n]}$.
- $\mathsf{Trace}(\mathsf{TK}, c, m)$ extracts the word w' from m and calls $\mathcal{T}.\mathsf{Trace}(\mathsf{tk}, w')$ to get the codeword w_{id} of a colluder.

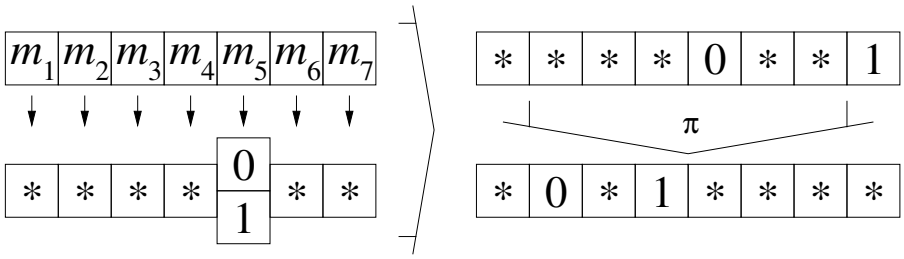

Figure 6.1: Hiding a mark at position 5 in a sequence of 7 blocks.

6.1.2 Improved Construction

We can reduce the ciphertext rate for long messages by watermarking only some blocks. We now describe a generic construction that accomplishes this by encrypting a message consisting of n sequences of ℓ blocks each in such a way that in sequence i, $\ell - 1$ blocks can be decrypted by both users; these blocks are not used for tracing. The other block is duplicated using two different marks and encrypted at two positions $v_0[i], v_1[i]$, each time for one key only: the message at position $v_0[i]$ cannot be decrypted by users with key 0, and the message at position $v_1[i]$ cannot be decrypted by users with key 1. By doing this, the ciphertext will have a length of $(1 + 1/\ell)$-times the length of the message, plus the overhead for encryption.

To reduce the overhead for encryption for long messages, we now model the PKE scheme as a KEM. Given any symmetric cipher $\mathcal{E} = (\mathsf{Enc}, \mathsf{Dec})$, a PKE Π, and a fingerprinting code $\mathcal{T}$, we construct a message-traceable encryption scheme $\hat{\Psi}(\ell, n)$ as follows:

- $\mathsf{Setup}(1^\kappa, N, t, \varepsilon, \ell)$:
  1. Generate a t-fingerprinting code, using $(\Gamma, \mathsf{tk}) \leftarrow \mathcal{T}.\mathsf{Gen}(N, \varepsilon)$, with a low error value ε. We thus denote n the length of a codeword in Γ, and enumerate codewords with indices associated to each users: $\Gamma = \{w_{\mathsf{id}}\}_{\mathsf{id}=1,\ldots,N} \subset \{0,1\}^n$.
  2. Call $\Pi.\mathsf{Setup}(1^\kappa)$ $n(\ell+1)$ times to obtain, for $i = 1, \ldots, n$ and $j = 1, \ldots, \ell+1$, $\mathsf{ek}_{i,j}, \mathsf{dk}_{i,j}$. Draw two random vectors $v_0, v_1 \in [\ell + 1]^n$ with the condition that $v_0[i] \neq v_1[i]$ for all $i = 1, \ldots n$. The position $v_b[i]$ describes the secret key that the users with $w_{\mathsf{id}}[i] = b$ do *not* have. Set

$$\begin{aligned} \mathsf{USK}_{\mathsf{id}} &\leftarrow \{\mathsf{dk}_{i,j}\}_{\substack{i=1,\ldots,n \\ j \neq v_{w_{\mathsf{id}}[i]}[i]}}, \\ \mathsf{EK} &\leftarrow (\{\mathsf{ek}_{i,j}\}_{\substack{i=1,\ldots,n \\ j=1,\ldots,\ell+1}}, v_0, v_1), \\ \mathsf{TK} &\leftarrow (\{\mathsf{dk}_{i,j}\}_{\substack{i=1,\ldots,n \\ j=1,\ldots,\ell+1}}, \mathsf{tk}). \end{aligned}$$

- $\mathsf{Encrypt}(\mathsf{EK}, m)$ first splits m in $n\ell$ blocks $\{m_{i,j}\}_{\substack{i=1,\ldots,n \\ j=1,\ldots,\ell}}$. For each sequence, at position $i \in \{1, \ldots, n\}$:
  1. Choose a random position $k \in \{1, \ldots, \ell\}$, to protect block $m_{i,k}$;
  2. Generate two equivalent versions $m^0_{i,k}, m^1_{i,k}$, of this block (see figure 6.1), resulting in a list of $\ell + 1$ blocks;
  3. Prepend the position to the block: $M_j = j \| m_{i,j}$, for $j = 1, \ldots, \ell$, $j \neq k$, $M_k = k \| m^0_{i,k}$, $M_{\ell+1} = k \| m^1_{i,k}$;
  4. Choose a random permutation $\pi \in S_{\ell+1}$ with the restriction that the position of the marked blocks is $v_1[i] = \pi(k)$ and $v_0[i] = \pi(\ell + 1)$; and permute the blocks: $M'_j = M_{\pi(j)}$.
  5. Generate the session keys: $(c_{i,j}, K_{i,j}) \leftarrow \Pi.\mathsf{Encaps}(\mathsf{ek}_{i,j})$,
  6. Encrypt the blocks under the symmetric keys: $C_{i,j} \leftarrow (c_{i,j}, c'_{i,j} = \mathsf{Enc}_{K_{i,j}}(M'_j))$ for $j = 1, \ldots, \ell + 1$.

  The final ciphertext consists of all the pairs $C_{i,j}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, \ell + 1$.
- $\mathsf{Decrypt}(\mathsf{USK}_{\mathsf{id}}, C)$ takes as input the key $\mathsf{USK}_{\mathsf{id}} = \{\mathsf{dk}_{i,j}\}_{\substack{i=1,\ldots,n \\ j \neq v_{w_{\mathsf{id}}[i]}[i]}}$ and a ciphertext $C = \{C_{i,j}\}$.

  For each sequence, at position $i \in \{1, \ldots, n\}$:
  1. Call $\Pi.\mathsf{Decaps}(\mathsf{usk}^{w_{\mathsf{id}}[i]}_{i,j}, c_{i,j})$, for $j \neq v_{w_{\mathsf{id}}[i]}[i]$, to obtain the session key $K_{i,j}$;
  2. Decrypt the message with $\mathsf{Dec}(K_{i,j}, c'_{i,j})$, which outputs $M'_{i,j}$;

3. It should be possible to parse the $M'_{i,j} = p_j \| m_{i,j}$, with $\{p_j\} = \{1, \ldots, \ell\}$, otherwise stop and output $\perp$;
4. Reorder the messages according to p_j.

It concatenates the ℓ blocks in each sequences, and the n resulting sequences to output the full message m.

– $\mathsf{Trace}(\mathsf{TK}, C, m)$ can detect the protected blocks using the decryption keys in TK. From the block that was actually decrypted in each sequence i, it can learn the value of the bit $w[i]$. Then, thanks to the traceability of the code $\mathcal{T}$, the $\mathcal{T}.\mathsf{Trace}(\mathsf{tk}, w)$ outputs a traitor.

6.1.3 Adapting the Code for Deletions

Because the public EK contains the vectors v_0, v_1 which describe the positions of the marked blocks in each sequence, the above scheme works well only if we require that the users output only complete messages. However, in practice removing small parts of a movie might still result in an acceptable quality and a reasonable scheme should take this into account and guarantee traceability even in the presence of deletions.

To cover this case, we must ensure that the adversary cannot drop specifically the blocks that contain watermarks, but that in order to delete a certain fraction of the codeword, he has to delete the same fraction of blocks. To accomplish this, the scheme needs several modifications. We cannot remove v_0, v_1 from EK, since the encryptor needs to know at which positions to put the marked blocks. The EK must therefore be kept secret. If users collude in constructing the message, i. e. the adversary has access to several user secret keys, they can choose to drop only the blocks that contain watermarks. In this case, the adversary can find the places where the codewords that correspond to the user secret keys differ, which reveals the marked message blocks, and drop only those. The tracing authority cannot know if a block was dropped at random or because the adversary knew it was marked, so tracing is impossible.

To solve this problem, we design a new code from a fingerprinting code by repeating and permuting the bits of the codeword. If the adversary cannot tell which bits of the new code are repetitions of the same bit, it is unlikely that he succeeds in erasing all repetitions of a bit from the fingerprinting code purely by accident. The tracing authority can then assume that for an erased bit, both keys are known to the attacker. The only problem is that colluding users can detect in which sequence they all decrypt the same blocks, which implies that they have the same code bit in this position. They can then drop blocks specifically from sequences where their code bits agree. To prevent this, we insert dummy bits that are the same for all users, which doubles the length of the code.

Let η be the fraction of blocks that the adversary is allowed to drop. We choose an integer ρ such that $\eta^\rho \leq \varepsilon/2n$. Let $w = w[1] \ldots w[n]$ be a codeword from the fingerprinting code. We generate a codeword of our new code by repeating each bit of w ρ times, padding it with $n\rho$ dummy bits that are identical for all users, then applying a permutation:
We first describe $\mathsf{Gen}(N, \varepsilon)$:

1. Generate the fingerprinting code: $(\Gamma, \mathsf{tk}) \leftarrow \mathcal{T}.\mathsf{Gen}(N, \varepsilon/2)$
2. Choose a random permutation $\pi : \{0,1\}^{2n\rho} \to \{0,1\}^{2n\rho}$
3. $\mathsf{tk}' \stackrel{\mathsf{def}}{=} (\mathsf{tk}, \pi^{-1})$
4. Choose a random string $s \stackrel{\$}{\leftarrow} \{0,1\}^{n\rho}$
5. $\Gamma' \stackrel{\mathsf{def}}{=} \{w'_{\mathsf{id}}\}_{\mathsf{id}=1,\ldots,N} \subset \{0,1\}^{2n\rho}$ where $w'_{\mathsf{id}} \stackrel{\mathsf{def}}{=} \pi(w_{\mathsf{id}}[1] \ldots w_{\mathsf{id}}[1] \ldots w_{\mathsf{id}}[n] \ldots w_{\mathsf{id}}[n] \| s)$.
6. Output (Γ', tk').

To trace, run the modified algorithm $\mathsf{Trace}(\mathsf{tk}', w')$:

1. Reconstruct a word w from $\pi^{-1}(w')$:
   (a) If all of the ρ replications of bit $w[i]$ that were not erased are equal to b, set $w[i] = b$.
   (b) If at least one of the replications of bit $w[i]$ has the value 1 and at least one of them has the value 0, choose $w[i]$ at random (in this case the adversary knows both keys).
   (c) If all replications of the bit $w[i]$ have been erased, choose $w[i]$ at random.
2. Return the output of $\mathcal{T}.\mathsf{Trace}(\mathsf{tk}, w)$.

Since a codeword is 2ρ times as long as previously, the user secret keys will also be 2ρ times as long. The tracing key tk contains a secret permutation π^{-1}, so tracing is no longer public as opposed to tracing for the fingerprinting code. However, in our construction we need to include the user secret keys in the tracing key, so in our construction tracing was not public even before the change in the code used.

Security Since these constructions are not the main result of this chapter, we do not give a security proof for them. We believe that given the security proof of our final construction, a security proof for the above constructions can be derived.

Remark 6.1. *The traceability of the scheme rests on the fact that a user does not know which of the keys are common to all users and which are specific to those with the same bit in the codeword. While a user that shares the information which positions he cannot decrypt with other users is considered to be misbehaving and thus corrupted in our security model, the real-life cost of sharing some of these positions is quite low. The scheme is thus susceptible to a Pirates 2.0-attack as described in [BP09].*

6.2 A Construction with Shorter Keys

The main disadvantage of the PKE-based construction is the length of the user keys, which must contain a PKE key for each block. Since a codeword has n bits, we can hope to reduce the number of different user keys to n as well. To achieve this, we use a primitive that allows encryption to either of two users or to both of them: 2-user broadcast encryption. [1]

Our message-traceable encryption scheme makes use of codes, where the bits of the codewords are embedded in a message by doubling some parts of it, the so-called *protected blocks*. Because we do not want the adversary to learn which parts of the message contain bits of the codeword, we need a broadcast encryption scheme where a user cannot tell whether a block is destined only for his key or for both keys, a 2-user anonymous broadcast encryption (2ABE).

This requires the symmetric cipher used with this construction to be weakly robust [ABN10], since one of the decapsulated keys will be either $\perp$ or an unusable key. The construction uses one instance of the 2ABE scheme Π per bit of the codeword, to encrypt all $\ell + 1$ messages in one sequence, with the target sets determined by the positions where the watermarks are embedded. In this construction, the length of the EK and USK is n times that of Π, and to encrypt a sequence of ℓ blocks, doubling one block, we need $\ell + 1$ Π key-encapsulations plus $\ell + 1$ symmetrically encrypted message blocks.

6.2.1 Construction of a Message-Traceable Encryption Scheme

Our first construction combines a fingerprinting code $\mathcal{T}$ with a 2ABE scheme Π. If the codewords have length n, we need n instances of the 2ABE scheme. Given any weakly robust [ABN10] symmetric cipher $\mathcal{E} = (\mathsf{Enc}, \mathsf{Dec})$, a 2-user anonymous broadcast encryption Π, and a traceable code $\mathcal{T}$, we construct a message-traceable encryption scheme $\Psi(\ell, n)$ as follows:

- $\mathsf{Setup}(1^\kappa, N, t, \varepsilon, \ell)$:
  1. Generates a t-fingerprinting code, using $(\Gamma, \mathsf{tk}) \leftarrow \mathcal{T}.\mathsf{Gen}(N, \varepsilon)$. We thus denote n the length of a codeword in Γ, and enumerate codewords with indices associated to each users: $\Gamma = \{w_{\mathsf{id}}\}_{\mathsf{id}=1,\ldots,N} \subset \{0,1\}^n$.
  2. Call $\Pi.\mathsf{Setup}(1^\kappa, 2)$ n times to obtain, for $i = 1, \ldots, n$, $\mathsf{ek}_i, \mathsf{usk}_i^0, \mathsf{usk}_i^1$. We set

$$\begin{aligned} \mathsf{USK}_{\mathsf{id}} &\leftarrow (\mathsf{usk}_1^{w_{\mathsf{id}}[1]}, \ldots, \mathsf{usk}_n^{w_{\mathsf{id}}[n]}), \\ \mathsf{EK} &\leftarrow (\mathsf{ek}_1, \ldots, \mathsf{ek}_n), \\ \mathsf{TK} &\leftarrow (\{\mathsf{usk}_i^0, \mathsf{usk}_i^1\}_{i=1,\ldots,n}, \mathsf{tk}). \end{aligned}$$

- $\mathsf{Encrypt}(\mathsf{EK}, m)$: Split m in $n\ell$ blocks $\{m_{i,j}\}_{\substack{i=1,\ldots,n \\ j=1,\ldots,\ell}}$. For each sequence, at position $i \in \{1, \ldots, n\}$:
  1. Choose a random position $k \in \{1, \ldots, \ell\}$, to protect block $m_{i,k}$;
  2. Generate two equivalent versions $m_{i,k}^0, m_{i,k}^1$, of this block (see figure 6.1), resulting in a list of $\ell + 1$ blocks;
  3. Prepends the position to the block: $M_j = j \| m_{i,j}$, for $j = 1, \ldots, \ell$, $j \neq k$, $M_k = k \| m_{i,k}^0$, $M_{\ell+1} = k \| m_{i,k}^1$;
  4. Choose a random permutation $\pi \in S_{\ell+1}$ and permute the blocks: $M'_i = M_{\pi(i)}$. We note $v = \pi(k)$ and $w = \pi(\ell + 1)$, the positions of the two equivalent blocks;

1. We view the scheme as an anonymous broadcast encryption scheme because broadcast encryption is a well-known concept that is intuitively understood. In [KY02c], the same primitive was called a "2-key, 1-copyrighted public-key encryption scheme".

5. Generate session keys for all the blocks, except M'_v and M'_w, with the 2ABE scheme Π, with the full target set $\{0,1\}$, whereas M'_v is targeted to $\{0\}$ only, and M'_w is targeted to $\{1\}$ only. More precisely, first generate the session keys: $(c_{i,j}, K_{i,j}) \leftarrow \Pi.\mathsf{Encaps}(\mathsf{ek}_i, \{0,1\})$, for $j \neq v, w$, $(c_{i,v}, K_{i,v}) \leftarrow \Pi.\mathsf{Encaps}(\mathsf{ek}_i, \{0\})$, and $(c_{i,w}, K_{i,w}) \leftarrow \Pi.\mathsf{Encaps}(\mathsf{ek}_i, \{1\})$.
6. Then encrypt the blocks under the symmetric keys: $C_{i,j} \leftarrow (c_{i,j}, c'_{i,j} = \mathsf{Enc}_{K_{i,j}}(M'_j))$ for $j = 1, \ldots, \ell+1$.

The final ciphertext consists of all the pairs $C_{i,j}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, \ell+1$.

- $\mathsf{Decrypt}(\mathsf{USK}_{\mathsf{id}}, C)$ takes as input the key $\mathsf{USK}_{\mathsf{id}} = (\mathsf{usk}_1^{w_{\mathsf{id}}[1]}, \ldots, \mathsf{usk}_n^{w_{\mathsf{id}}[n]})$ and a ciphertext $C = \{C_{i,j}\}$. For each sequence, at position $i \in \{1, \ldots, n\}$:
  1. Call $\Pi.\mathsf{Decaps}(\mathsf{usk}_i^{w_{\mathsf{id}}[i]}, c_{i,j})$, for $j = 1, \ldots, \ell+1$, to obtain the session key $K_{i,j}$;
  2. Decrypt the message with $\mathsf{Dec}(K_{i,j}, c'_{i,j})$, which outputs either $M'_{i,j}$ or $\perp$ (because of the robustness);
  3. It should be possible to parse the $M'_{i,j} = p_j \| m_{i,j}$, with $\{p_j\} = \{1, \ldots, \ell\}$, otherwise stop and output $\perp$;
  4. Reorder the messages according to p_j, and concatenates the other parts.

  Concatenate the ℓ blocks in each sequence, then the n resulting sequences to obtain the full message m.
- $\mathsf{Trace}(\mathsf{TK}, C, m)$ can detect the protected blocks using the decryption keys in TK. From the block that was actually decrypted in each sequence i, it can learn the value of the bit $w[i]$. Then, thanks to the traceability of the code $\mathcal{T}$, the $\mathcal{T}.\mathsf{Trace}(\mathsf{tk}, w)$ outputs a traitor.

6.2.2 Security of the Construction

We show that our construction fulfills IND-CPA security and explain under which conditions it is traceable.

Theorem 6.2. *If the 2ABE scheme Π is IND-CPA and the symmetric encryption scheme $\mathcal{E}$ is IND-CPA, then our construction $\Psi(\ell, n)$ is IND-CPA, and for $\tau' \approx \tau + \tau_{sim}$, where τ_{sim} is the running time of the simulator, we have*

$$\mathrm{Adv}^{\mathsf{ind\text{-}cpa}}_{\Psi(\ell,n)}(\kappa, \tau, N, t, \varepsilon) \leq n \cdot (\ell+1) \times \left(2 \cdot \mathrm{Adv}^{\mathsf{ind\text{-}cpa}}_{\Pi}(\kappa, \tau', 2) + \mathrm{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{E}}(\kappa, \tau')\right).$$

Proof. Let $\mathcal{A}$ be an adversary against the IND-CPA security of our construction Ψ. We provide a bound on its advantage using a series of games. The simulator first asks for n public keys ek_i, for $i = 1, \ldots, n$, to $\Pi.\mathsf{Setup}(1^\kappa, 2)$, and thus sends the public key EK to the adversary $\mathcal{A}$. The latter sends back two messages $m^0 = (m^0_{1,1}, \ldots, m^0_{1,\ell}, \ldots, m^0_{n,1}, \ldots, m^0_{n,\ell})$ and $m^1 = (m^1_{1,1}, \ldots, m^1_{1,\ell}, \ldots, m^1_{n,1}, \ldots, m^1_{n,\ell})$:

- In game G_0, the simulator encrypts m^0, with k_i the indices of the protected blocks for sequences $i = 1, \ldots, n$, and π_i the permutations;
- In game G_1, the simulator still encrypts m^0, but with random keys for all the key encapsulations of the 2ABE scheme: with an hybrid sequence of games, we can show that the distance between game G_1 and game G_0 is bounded by $n(\ell+1) \cdot \mathrm{Adv}^{\mathsf{ind\text{-}cpa}}_{\Pi}(\kappa, \tau', 2)$;
- In game G_2, the simulator encrypts m^1, still with random keys for all the key encapsulations of the 2ABE scheme: with an hybrid sequence of games, we can show that the distance between game G_2 and game G_1 is bounded by $n(\ell+1) \cdot \mathrm{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{E}}(\kappa, \tau')$;
- In game G_3, the simulator encrypts m^1, with k_i the indices of the protected blocks for sequences $i = 1, \ldots, n$, and π_i the permutations: with an hybrid sequence of games, we can show that the distance between game G_3 and game G_2 is bounded by $n(\ell+1) \cdot \mathrm{Adv}^{\mathsf{ind\text{-}cpa}}_{\Pi}(\kappa, \tau', 2)$;

This concludes the proof. □

Before we turn to tracing, we state an intermediate result. The following lemma says that no adversary can tell which blocks are not encrypted to all users, if he only has one of the two keys, w.l.o.g. usk_i^0.

Lemma 6.3. *If the 2ABE scheme Π is both IND-CPA and ANO-CPA, and the symmetric encryption scheme $\mathcal{E}$ is IND-CPA, then an adversary who only has the usk_i^0 for a sequence i cannot distinguish between the case where the block at position v is encrypted to the target set $\{0\}$ and the block at position w is encrypted to the target set $\{1\}$ and the case where the block at position v is encrypted to the target*

set $\{0,1\}$ *and the block at position* w *contains a random message. If we denote by* $\mathrm{Adv}^{\mathsf{find}}_{\Psi}(\kappa,\tau,N,t)$ *the maximal advantage of any adversary within time* τ*, on any index* i*, then for* $\tau' \approx \tau + \tau_{sim}$*, we have*

$$\mathrm{Adv}^{\mathsf{find}}_{\Psi}(\kappa,\tau,N,t,\varepsilon) \leq \mathrm{Adv}^{\mathsf{ano\text{-}cpa}}_{\Pi}(\kappa,\tau',2) + \mathrm{Adv}^{\mathsf{ind\text{-}cpa}}_{\Pi}(\kappa,\tau',2) + \mathrm{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{E}}(\kappa,\tau').$$

Proof. Let $\mathcal{A}$ be an adversary who can distinguish the two cases for our construction Ψ. We provide a bound on its advantage using a series of games. The simulator first asks for n public keys ek_i, for $i = 1,\ldots,n$, to $\Pi.\mathsf{Setup}(1^\kappa,2)$, and thus sends the public key EK to the adversary $\mathcal{A}$, together with $\mathsf{USK}_{\mathsf{id}}$, that corresponds to a codeword w_{id}. The latter sends back a message $M = (M_{1,1},\ldots,M_{1,\ell},\ldots,M_{n,1},\ldots,M_{n,\ell})$. For all the sequences, except the i-th sequence, for any i, the simulator does as in the real encryption of message M, but we denote by m the sequence $M_{i,1}\ldots M_{i,\ell}$.

- In game G_0, the simulator encrypts m, with k the index of the protected block, and (v,w) the positions of the specific ciphertexts to key 0 and key 1 respectively.
- In game G_1, the simulator encrypts m, with k the index of the protected block, and (v,w) the two above positions, but v-th block is encrypted for the two keys, and w-th block is encrypted for the key 1 only. Since the simulator can choose the index v to be the same as the one asked to the anonymity-game, in an indistinguishable way, the distance between game G_1 and game G_0 is bounded by $\mathrm{Adv}^{\mathsf{ano\text{-}cpa}}_{\Pi}(\kappa,\tau',2)$;
- In game G_2, the simulator encrypts m, with k the index of the protected block, and (v,w) the two above position, but v-th block is encrypted for the two keys, and a random session key is used for the w-th block, while a key encapsulation is sent for key 1 only. Under the semantic security of the 2ABE scheme, since the adversary does not know key 1, the two games are indistinguishable: the distance between game G_2 and game G_1 is bounded by $\mathrm{Adv}^{\mathsf{ind\text{-}cpa}}_{\Pi}(\kappa,\tau',2)$;
- In game G_3, the simulator encrypts m, with k the index of the protected block, and (v,w) the two above position, but the v-th block is encrypted for the two keys, and at position w, a truly random block (with no position appended) is encrypted under a random session key, while a key encapsulation is sent for key 1 only. Under the semantic security of $\mathcal{E}$, the two games are indistinguishable: the distance between game G_3 and game G_2 is bounded by $\mathrm{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathcal{E}}(\kappa,\tau')$.

□

It follows immediately from the lemma that as long as the message is complete, i. e. no blocks were dropped, any collusion of up to t users can be traced.

We now consider the case where the adversary is allowed to drop blocks. If there are no collusions, the case is unproblematic if robust codes are used (that resist erasures), because if the adversary drops a fraction δ of the blocks, he cannot erase significantly more than a fraction δ of the bits in the codeword.

Theorem 6.4. *Even if an adversary with a single user secret key can drop a fraction* η *of the message, if the 2ABE scheme* Π *is both IND-CPA and ANO-CPA, the symmetric encryption scheme* $\mathcal{E}$ *is IND-CPA, and the code* $\mathcal{T}$ *is t-fingerprinting for* $\mathcal{FS}^\star$ *for a fraction* $\delta > \eta$ *of erasures, then our construction* Ψ *is traceable. More precisely, one needs*

$$(\delta-\eta)^2 \geq \frac{1}{2} \times \mathrm{Adv}^{\mathsf{find}}_{\Psi}(\kappa,\tau,N,t,\varepsilon).$$

Proof. We first fix some notation. Let $\eta_{i,j} \stackrel{\mathsf{def}}{=} \Pr[drop(i,j)]$ be the probability that block j in sequence i is dropped. We denote by $\eta \stackrel{\mathsf{def}}{=} E[\eta_{i,j}]$ the global average probability of a block to be dropped. This means that overall, $n\ell\eta = \sum_{i,j}\eta_{i,j}$ blocks will be dropped. We also consider the average in a sequence i: $\eta_i \stackrel{\mathsf{def}}{=} E[\eta_{i,j}]$. Then $\eta = E[\eta_i]$. Now we consider only the protected blocks. Analogously, let $\theta_i \stackrel{\mathsf{def}}{=} \Pr[drop(i,k_i)]$ be the probability for the protected block in sequence i to be dropped. The average probability of a protected block to be dropped is $\theta \stackrel{\mathsf{def}}{=} E[\theta_i]$. This means that the global number of erasures (dropped protected blocks) is $n\theta = \sum_i \theta_i$. We also consider the gap between protected blocks and any block, by first defining $\gamma_i \stackrel{\mathsf{def}}{=} \theta_i - \eta_i$. We additionally define $\gamma \stackrel{\mathsf{def}}{=} E[\gamma_i] = \theta - \eta$, which we want to show to be small.

Let us define the subset of the sequences in which the gap γ_i is greater than $\gamma - \alpha$ for some α: $I \stackrel{\mathsf{def}}{=} \{i | \gamma_i \geq \gamma - \alpha\}$. We choose a random sequence index $i \in \{1,\ldots,n\}$, and by the splitting lemma [PS00, lemma 1], we know that $\Pr[i \in I] = \Pr[\gamma_i \geq \gamma - \alpha] \geq \alpha$, so with probability greater than α, the gap $\gamma_i = \theta_i - \eta_i$ between the probabilities for the protected block and any block in sequence i to be dropped is greater than $\gamma - \alpha$.

We now focus on this sequence i and the sub-message $m = m_1 \dots m_\ell$, where m_k is the protected block. We consider the probability of the adversary to drop block k. If the adversary drops the block k, the simulator outputs 1, otherwise the simulator outputs 0. When interacting with the real scheme, the advantage of the simulator (defined as $\Pr[1 \leftarrow \mathcal{S}] - \Pr[0 \leftarrow \mathcal{S}]$) in this game is $\theta_i - (1 - \theta_i) = 2\theta_i - 1$. If we change the scheme so that now m_k is encrypted to both parties, and instead of doubling it, a random block is added, then the advantage is $2\eta_i - 1$.

By lemma 6.3, the difference between these games is bounded by $\mathrm{Adv}_\Psi^{\mathsf{find}}(\kappa, \tau, N, t)$. We conditioned on $i \in I$, or, equivalently, $\theta_i - \eta_i = \gamma_i \geq \gamma - \alpha$, which happens with probability α.

$$\begin{aligned} 2\theta_i - 1 - (2\eta_i - 1) &\leq \frac{1}{\alpha} \times \mathrm{Adv}_\Psi^{\mathsf{find}}(\kappa, \tau, N, t) \\ 2(\gamma - \alpha) &\leq \frac{1}{\alpha} \times \mathrm{Adv}_\Psi^{\mathsf{find}}(\kappa, \tau, N, t) \\ \gamma &\leq \alpha + \frac{1}{2\alpha} \times \mathrm{Adv}_\Psi^{\mathsf{find}}(\kappa, \tau, N, t) \\ \theta &\leq (\alpha + \eta) + \frac{1}{2\alpha} \times \mathrm{Adv}_\Psi^{\mathsf{find}}(\kappa, \tau, N, t). \end{aligned}$$

Then, except when it drops blocks, the adversary outputs messages for the protected blocks, and under the assumptions we did on the marking of contents, we can detect a bit. Furthermore, the detected bit follows the rule for the feasible set $\mathcal{FS}$, and erasures (dropped blocks) extend it to $\mathcal{FS}^\star$. Since the code $\mathcal{T}$ is (N, t, ε, n)-traceable for $\mathcal{FS}^\star$ for a fraction δ of erasures, then our construction Ψ is traceable with tracing-error probability less than ε if the average fraction θ of dropped protected blocks is less than δ: that is, if there exists α such that

$$\begin{aligned} (\alpha + \eta) + \frac{1}{2\alpha} \times \mathrm{Adv}_\Psi^{\mathsf{find}}(\kappa, \tau, N, t) &\leq \delta \\ \alpha + (\eta - \delta) + \frac{1}{2\alpha} \times \mathrm{Adv}_\Psi^{\mathsf{find}}(\kappa, \tau, N, t) &\leq 0 \\ 2\alpha^2 - 2(\delta - \eta)\alpha + \mathrm{Adv}_\Psi^{\mathsf{find}}(\kappa, \tau, N, t) &\leq 0. \end{aligned}$$

To find the minimum, we take the derivative:

$$\begin{aligned} 4\alpha - 2(\delta - \eta) &= 0 \\ \alpha &= \frac{\delta - \eta}{2}. \end{aligned}$$

This is possible as soon as

$$\begin{aligned} 2\alpha^2 - 2(\delta - \eta)\alpha + \mathrm{Adv}_\Psi^{\mathsf{find}}(\kappa, \tau, N, t) &\leq 0 \\ \frac{(\delta - \eta)^2}{2} - (\delta - \eta)^2 + \mathrm{Adv}_\Psi^{\mathsf{find}}(\kappa, \tau, N, t) &\leq 0 \\ \mathrm{Adv}_\Psi^{\mathsf{find}}(\kappa, \tau, N, t) &\leq \frac{(\delta - \eta)^2}{2}. \end{aligned}$$

This means that we can trace as long as we choose

$$\delta \geq \eta + \sqrt{2\mathrm{Adv}_\Psi^{\mathsf{find}}(\kappa, \tau, N, t)}.$$

□

We now show that our improvement from section 6.1.3 allows tracing collusions of users even if they can drop blocks.

Theorem 6.5. *Even if an adversary with a single user secret key can drop a fraction η of the message, if the 2ABE scheme Π is both IND-CPA and ANO-CPA, the symmetric encryption scheme $\mathcal{E}$ is IND-CPA, and the code $\mathcal{T}$ is t-fingerprinting with error $\varepsilon/2$, then our construction Ψ with ρ repetitions of each bit of the codeword and $n\rho$ dummy bits is t-fingerprinting with error at most ε as long as*

$$(2\eta)^\rho \leq \frac{\varepsilon}{2n}.$$

Proof. We need to show that the tracing algorithm works. We accomplish this by showing that from any word w' we can reconstruct a word w that we can use with $\mathcal{T}.\mathsf{Trace}$ to identify a traitor. We inspect the reconstruction of bit $w[i]$ from its ρ repetitions $w[i]_1, \dots w[i]_\rho$ that can take values in $\{0, 1, \star\}$ and define two events where the tracing fails. Let $C = \{w_1, \dots w_t\}$ denote the set of codewords that correspond to the corrupted users. We define $\mathcal{FS}(C[i])$ to be the set $\{w_1[i], \dots w_t[i]\}$ of the i-th bits, which can be either of $\{0\}, \{1\}, \{0, 1\}$.

- Event E_a is the case if there is an index $j \in [\rho]$ with $w[i]_j \neq \star$ and $w[i]_j \notin \mathcal{FS}(C[i])$.
- Event E_b is the case if all $w[i]_j$ are equal to $\star$ and $\mathcal{FS}(C[i]) \neq \{0, 1\}$

Event E_a only happens if the adversary inverted the encryption or broke the watermarking assumption. Event E_b occurs with probability at most $\varepsilon/2n$. Due to lemma 6.3 and the random choice of π, the best strategy of the adversary is to drop any block in a sequence where he does not have both keys with probability η. Since half the code bits are dummy bits, he can at most double the fraction of dropped blocks from η to 2η by concentrating the deletions in this way. Therefore, the probability that all ρ copies of $w[i]$ are dropped is $(2\eta)^\rho \leq \varepsilon/2n$.

It follows that the error probability for each of the n bits of the codeword w is at most $\varepsilon/2n$, so the error probability of the tracing algorithm is at most $n(2\eta)^\rho + \varepsilon/2 \leq \varepsilon$. □

6.2.3 A 2-user Anonymous Broadcast Encryption Scheme

We now present a concrete instance of a 2ABE scheme to use as a building block in our message-based traitor tracing scheme. We view the 2-key 1-copyrighted public-key encryption scheme of Kiayias and Yung [KY02c], as a 2-user 1-collusion secure anonymous broadcast encryption scheme (2ABE). For ease of exposition, we model the scheme as a KEM.

Let $\mathbb{G}$ be a group of prime order q, with a generator g. The public parameters consist of $(\mathbb{G}, q, g)$. Since we consider the 2-user case, we drop the parameter N:

- $\mathsf{Setup}(1^\kappa)$ picks $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_q^\times$. For the two user-keys one chooses $d'_0, d'_1 \in \mathbb{Z}_q$, and sets $\mathsf{usk}_u \stackrel{\mathsf{def}}{=} (d_u = \alpha - d'_u \cdot \beta, d'_u)$, for $u = 0, 1$. The encryption key is $\mathsf{ek} \stackrel{\mathsf{def}}{=} \{(f = g^\alpha, h = g^\beta), \mathsf{upk}_0 = h^{d'_0}, \mathsf{upk}_1 = h^{d'_1}\}$.
- $\mathsf{Encaps}(\mathsf{ek}, S; r)$ where $r \in \mathbb{Z}_q^\times$
  - if $S = \{0, 1\}$ then $c = (g^r, h^r), K = f^r$
  - else if $S = \{u\}$ then $r' \xleftarrow{\$} \mathbb{Z}_q^\times$, and $c = (g^r, h^{r'}), K = (f/\mathsf{upk}_u)^r \times \mathsf{upk}_u^{r'}$
- $\mathsf{Decaps}(\mathsf{usk}_u, c)$ computes $K = c_0^{d_u} c_1^{d'_u}$. This is equal to $g^{r d_u} \times h^{r' d'_u} = (f/\mathsf{upk}_u)^r \times \mathsf{upk}_u^{r'}$, which ensures correctness in the second case, where $S = \{u\}$. In the first case, since $r' = r$, both users get the same key f^r.

This is a broadcast encryption, because when $S = \{u\}$, the user $1-u$ decapsulates differently. Anonymity comes from the fact that a ciphertext is either a Diffie-Hellman pair, when $S = \{0, 1\}$, and a random pair in the other case.

6.2.4 Security of the 2ABE

Theorem 6.6. *If solving the DDH problem in the underlying group is hard, then the 2ABE scheme presented in section 6.2.3 is ANO-ACPA secure and for $\tau' \approx \tau + 6\tau_{exp}$ we have*

$$\mathrm{Adv}^{\mathsf{ano\text{-}acpa}}_{2ABE}(\kappa, \tau) \leq 4 \cdot \mathrm{Adv}^{ddh}_{\mathbf{G}}(\kappa, \tau').$$

Proof. The simulator is given a tuple $(g, A = g^a, B = g^b, C = g^c)$. First the simulator will guess the target set. There are only three possibilities (up to the order of the target sets). The simulator chooses

1. $\{0\}, \{1\}$ with probability 1/2.
   In this case, we implicitly set $d'_u = a$ for a user u, and $r = b$ for the challenge random coins. The simulator proceeds as follows. It first flips a bit u to determine where it will embed the challenge. It chooses random $\alpha, \beta \xleftarrow{\$} \mathbb{Z}_q^\times$, and defines $f = g^\alpha, h = g^\beta$. It generates $\mathsf{usk}_{1-u} = (d_{1-u}, d'_{1-u})$ as usual, and thus $\mathsf{upk}_{1-u} = h^{d'_{1-u}}$, but $\mathsf{upk}_u = A$. It thus defines $\mathsf{ek} = ((f, h), \mathsf{upk}_0, \mathsf{upk}_1)$. The simulator then draws a random r' and sets the challenge ciphertext to $(B, h^{r'})$. It also returns $K = B^\alpha A^{r'}/C$.
   If the tuple (g, A, B, C) was a DH tuple, we have a ciphertext $c = (g^b, h^{r'})$ and a key $K = g^{b\alpha + ar' - ab} = (f/\mathsf{upk}_u)^b \times \mathsf{upk}_u^{r'}$, that is for $S = \{u\}$. If the tuple was a random tuple, then the key is random.

Construction in	$\lvert\mathsf{EK}\rvert$	$\lvert\mathsf{USK}\rvert$	|KeyHeader|	|CTXT|	Ciphertext Rate
Section 6.1.1	$2\ell n\mathsf{G}$	$\ell n\mathsf{G}$	$\ell n\mathsf{G}$	$2\ell n\mathsf{B}$	$2+\mathsf{G}/\mathsf{B}$
Section 6.1.2	$2(\ell+1)n\mathsf{G}$	$\ell n\mathsf{G}$	$(\ell+1)n\mathsf{G}$	$(\ell+1)n\mathsf{B}$	$1+1/\ell+(1+1/\ell)\mathsf{G}/\mathsf{B}$
Section 6.1.3	$4(\ell+1)n\rho\kappa$	$2\ell n\rho\kappa$	$2(\ell+1)n\rho\kappa$	$2(\ell+1)n\rho\mathsf{B}$	$1+1/\ell+(1+1/\ell)\kappa/\mathsf{B}$
Section 6.2	$8n\rho\mathsf{G}$	$4n\rho\mathsf{G}$	$4(\ell+1)n\rho\mathsf{G}$	$2(\ell+1)n\rho\mathsf{B}$	$1+1/\ell+(2+2/\ell)\mathsf{G}/\mathsf{B}$

Table 6.1: Comparison: G is the bit-length of a group element; B is the bit-length of a message block. |PTXT| is always $\ell n\mathsf{B}$, except for the schemes from section 6.1.3 and 6.2, where it is $2\ell n\rho\mathsf{B}$.

2. $\{0\},\{0,1\}$ with probability 1/4.
   In this case, we implicitly set $\beta = a$, and $r = b$ for the challenge random coins. In this case, the simulator knows that only user 0 can be corrupted, so it chooses the secret key for user 0 in advance and uses it to compute a group element that implicitly has a matching α.

   The simulator proceeds as follows. It sets $h = A$, chooses random $d_0, d_0' \xleftarrow{\$} \mathbb{Z}_q$ and computes $f \stackrel{\mathsf{def}}{=} g^{d_0} A^{d_0'}$. It sets $\mathsf{usk}_0 = (d_0, d_0')$, $\mathsf{upk}_0 = A^{d_0'}$, and $\mathsf{upk}_1 = X$ for a random group element X. It thus defines $\mathsf{ek} = ((f, A), \mathsf{upk}_0, \mathsf{upk}_1)$. The challenge ciphertext is set to $c = (B, C)$, for the key $K = B^{d_0} C^{d_0'}$.

   If the tuple (g, A, B, C) was a DH tuple, we have a ciphertext $c = (g^b, g^{ab} = h^b)$ and a key $K = g^{bd_0 + abd_0'} = g^{b(d_0 + ad_0')} = f^b$, that is for $S = \{0,1\}$. If the tuple was a random tuple, then $c = (g^b, g^{ab'} = h^{b'})$ where $b' = c/a \neq b$ and a key $K = g^{bd_0 + ab'd_0'} = g^{b(d_0+ad_0')} \times g^{ad_0'(b'-b)} = (f/\mathsf{upk}_0)^b \times \mathsf{upk}_0^{b'}$, that is for $S = \{0\}$.
3. $\{1\},\{0,1\}$ with probability 1/4.
   This case is done analogously to case 2, exchanging user 0 and 1.

□

Theorem 6.7. *If solving the DDH problem in the underlying group is hard, then the 2ABE scheme presented in section 6.2.3 is a 2-user IND-CPA secure BE scheme and for $\tau' \approx \tau + 5\tau_{exp}$ we have*

$$\mathrm{Adv}^{\mathsf{ind\text{-}cpa}}_{2ABE}(\kappa, \tau) \leq \mathrm{Adv}^{\mathit{ddh}}_{\mathbf{G}}(\kappa, \tau').$$

Proof. The simulator is given a tuple $(g, A = g^a, B = g^b, C = g^c)$.

If there is no corruption, a ciphertext is $c = (g^r, h^r)$ for a key f^r. We can implicitly set $h = g^x$ for a known x, $r = a$, $f = B$, $c = (A, A^x)$ and $K = C$, which is a real key if C is the actual Diffie-Hellman value, or a random key otherwise.

In case of corruption, we proceed as in the cases 2 and 3 in the proof of theorem 6.6, where u is the user for which we know the secret key for corruption. □

6.3 Efficiency Considerations

Table 6.1 shows a comparison of several ways to do message-traceable encryption. We compare the length of the encryption key ($\lvert\mathsf{EK}\rvert$) and the user secret key ($\lvert\mathsf{USK}\rvert$), and the length of the key header and the symmetric encryption of the plaintext (CTXT), normalizing for a plaintext (PTXT) length (before marking blocks) of ℓn blocks. The ciphertext rate is defined as (|KeyHeader| + |CTXT|) / |PTXT|. In this comparison, we ignore the cost of $\log \ell$ bits for prepending a block's number and the cost of making the symmetric encryption scheme robust. Assuming we use κ bits per block to achieve robustness and these bits already contain the block number, the ciphertext rate of the first scheme increases by $\frac{2\ell n\kappa}{\ell n\mathsf{B}} = 2\frac{\kappa}{\mathsf{B}}$, and of the other schemes by $\frac{(\ell+1)n\kappa}{\ell n\mathsf{B}} = \frac{1+\ell}{\ell}\frac{\kappa}{\mathsf{B}}$.

The simplest way is to use any public-key encryption scheme to encrypt each message block twice, described in section 6.1.1: two pairs of keys are generated for each message block. Using ElGamal, we have one group element for the key header (the key encapsulation) per message-version block. Using this method it is impossible to reduce the ciphertext rate below 2 without leaving some part of the message unprotected and exposed to untraceable rebroadcasting, since each message block is encrypted twice with the symmetric keys.

Using our improved construction from section 6.1.2, we immediately cut the number of blocks that must be sent almost in half with only a small constant increase in the key header as compared to plain ElGamal. The modifications from section 6.1.3 multiply key and ciphertext length by 2ρ, but does not change the ciphertext rate (except for the fact that the scheme is secret-key with key length κ).

Our main construction from section 6.2 achieves the same asymptotic efficiency as the PKE construction, but with keys that are shorter by a factor of ℓ.

Since these results are all asymptotic, the question arises how our scheme performs in practice. Assume we want to design a scheme for $N = 2^{30}$ users, with a maximum false positive rate of $\varepsilon = 2^{-30}$. We set the collusion threshold to $t = 16$, which means we assume that retrieving the keys from 16 different decoders is prohibitively expensive (16 is the largest collusion threshold considered in [JL07, fig. 5]). Then the length of the code is $n = d_m t^2 \log(N/\varepsilon)$ for some constant d_m. Blayer and Tassa claim that in most real-world applications, one can find a code with $d_m < 8$ [BT08], giving us a code length of $n = 122\ 880$ bits in exchange for a higher false-negative error rate. We assume that we only want to protect against traitors that rebroadcast at least 97% of the blocks, and set $\eta = 2^{-5}$. Since we need $(2\eta)^\rho \leq \varepsilon/2N$, $\rho = 16$, and we double the code length by padding with dummy bits, then our modified code has a length of $2\rho n = 3\ 932\ 160$ bits. We choose the efficiency parameter $\ell = 22$, so we need to split the film into 86 507 520 blocks.

To determine the size of group elements, we look at the security reduction. We lost a factor of $2(2\rho n)(\ell+1)$ in the security reduction, which translates to a security loss of $\log(2(2\rho n)(\ell+1))$ or 28 bits, so to obtain a security level of 80 bits, we need to use 216-bit group elements in a suitable elliptic curve.

Assuming a block size of 1 kB, our film needs to be at least 86.5 GB. [2] Because of the doubled blocks and the additional 40 bits (5 B) per block to indicate its position and make the cipher weakly robust, the symmetrically encrypted part would be about 90.9 GB, and the length of the key header would be about 4.9 GB, for a total ciphertext size of 95.8 GB that would comfortably fit on an existing 100 GB BDXL Blu-ray disc. The overhead of 10.7% is close to the acceptable range — [JL07] state 10% as their limit — and as we can expect the size of media formats to grow, the concrete efficiency of our scheme will only increase.

2. To relate our ciphertext length to existing storage media sizes, we use SI prefixes.

Conclusion

The goal of this thesis was to bring the theory and praxis of broadcast encryption closer together. Solid security definitions and a good understanding of their relationships are the indispensable basis of the formal study of cryptology, but they are also important for applications. Users of cryptographic schemes like to know what security can be guaranteed to them, and it makes no sense to compare only the efficiency of schemes without looking at what security we get for the price we pay in efficiency. We were also interested in efficient black-box constructions, both because they improve our understanding of the tasks at hand, and because they allow modularization of schemes that brings us automatic improvements when more efficient building blocks are discovered.

This thesis was therefore divided into three parts. In the first one, we were able to give a clean framework that relates the different notions of security we defined, and discovered some surprising relationships where some definitions from the literature behave differently than might have been expected. We were also able to point out several instances where schemes from the literature achieve a stronger security notion than was originally claimed.

We proceeded with a black-box construction of a powerful broadcast encryption scheme from simple primitives. The scheme has no need for trusted parties, because the setup phase is run as a protocol between peers and the encryption key is public. The scheme achieves the strongest security notion we defined and can be instantiated in the standard model under the DDH assumption, which is today one of the standard assumptions.

We finally constructed a tracing scheme from a two-user anonymous broadcast encryption scheme, a symmetric encryption scheme and a robust fingerprinting code. The scheme uses watermarks in the message to trace traitors, which makes it possible to trace not only the producers of pirate decoders, but also traitors which distribute session keys or the content, even if the content is degraded. The scheme's overhead is asymptotically zero, and for reasonable concrete parameters the overhead can be considered practical.

Abbreviations

2ABE	2-user anonymous broadcast encryption scheme, page 122
ABE	attribute-based encryption, page 48
ANOBE	anonymous broadcast encryption scheme, page 54
BDHE	decisional bilinear Diffie-Hellman exponent assumption, page 37
BE	broadcast encryption, page 51
CS	complete subtree, page 62
DBE	dynamic broadcast encapsulation scheme, page 52
DDBE	decentralized dynamic broadcast encapsulation scheme, page 105
DEM	data encapsulation mechanism, page 47
DHKE	Diffie-Hellman key exchange, page 118
GBDHE	generalized decisional bilinear Diffie-Hellman exponent assumption, page 37
GGM	generic group model, page 34
GKA	group key agreement, page 50
GKEA	generalized knowledge of exponent assumption, page 100
HIBE	hierarchical identity-based encryption, page 48
IBE	identity-based encryption, page 47
KEM	key encapsulation mechanism, page 47
MT	message-traceable encryption scheme, page 57
OBDHE	oracle bilinear Diffie-Hellman assumption, page 100
OWF	one-way function, page 42
PRG	pseudo-random generator, page 43
ROM	random oracle model, page 34
SC	subset-cover, page 61
SD	subset difference, page 62
SKE	subgroup key exchange, page 104
TOWP	trapdoor one-way permutation, page 45
TT	traitor-tracing scheme, page 55
UOWHF	universal one-way hash function, page 42

Bibliography

[AAC09] AACS Consortium. Advanced Access Content System (AACS) - introduction and common cryptographic elements book. `http://www.aacsla.com/specifications/`, September 2009. Revision 0.951. 15, 26, 103

[ABC+11] Michel Abdalla, James Birkett, Dario Catalano, Alexander W. Dent, John Malone-Lee, Gregory Neven, Jacob C. N. Schuldt, and Nigel P. Smart. Wildcarded identity-based encryption. *J. Crypto*, 24(1):42–82, January 2011. 76

[ABN10] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 480–497. Springer, 2010. 125

[ABR01] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle diffie-hellman assumptions and an analysis of DHIES. In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, page 143–158. Springer, 2001. 101

[ACD+06] Michel Abdalla, Dario Catalano, Alexander W. Dent, John Malone-Lee, Gregory Neven, and Nigel P. Smart. Identity-based encryption gone wild. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006*, volume 4052 of *LNCS*, pages 300–311. Springer, 2006. Full version at `http://eprint.iacr.org/2006/304`. 25, 76

[ACMP10] Michel Abdalla, Céline Chevalier, Mark Manulis, and David Pointcheval. Flexible group key exchange with on-demand computation of subgroup keys. In *Africacrypt 2010*, volume 6055 of *LNCS*, pages 351–368. Springer, 2010. 15, 16, 27, 103, 104, 106, 107

[AF07] Masayuki Abe and Serge Fehr. Perfect nizk with adaptive soundness. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, page 118–136. Springer, 2007. 101

[AFI06] Nuttapong Attrapadung, Jun Furukawa, and Hideki Imai. Forward-secure and searchable broadcast encryption with short ciphertexts and private keys. In *Asiacrypt 2006*, volume 4284 of *LNCS*, pages 161–177. Springer, 2006. 69

[AFP05] Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. Password-based authenticated key exchange in the three-party setting. In *PKC 2005*, volume 3386 of *LNCS*, pages 65–84. Springer, 2005. 49, 107

[AHL+12] Nuttapong Attrapadung, Javier Herranz, Fabien Laguillaumie, Benoît Libert, Elie de Panafieu, and Carla Ràfols. Attribute-based encryption schemes with constant-size ciphertexts. *Theoretical Computer Science*, 422:15–38, March 2012. 76

[AI07] Nuttapong Attrapadung and Hideki Imai. Practical broadcast encryption from graph-theoretic techniques and subset-incremental-chain structure. *IEICE Transactions*, 90-A(1):187–203, 2007. A preliminary version appeared at Asiacrypt 2005. 60, 66

[AI09] Nuttapong Attrapadung and Hideki Imai. Conjunctive broadcast and attribute-based encryption. In H. Shacham and B. Waters, editors, *Pairing 2009*, volume 5671 of *LNCS*, pages 248–265. Springer, 2009. 75

[AK05] Tomoyuki Asano and Kazuya Kamio. A tree based one-key broadcast encryption scheme with low computational overhead. In C. Boyd and J.M. Gonzalez Nieto, editors, *ACISP 2005*, volume 3574 of *LNCS*, pages 89–100. Springer, 2005. 60, 64

[AL12] Nuttapong Attrapadung and Benoît Libert. Functional encryption for public-attribute inner products: Achieving constant-size ciphertexts with adaptive security or support for negation. *J. Mathematical Cryptology*, 5(2):115–158, 2012. A preliminary version appeared at PKC 2010. 70, 76

[AM09] Divesh Aggarwal and Ueli Maurer. Breaking rsa generically is equivalent to factoring. In *Eurocrypt 2009*, volume 5479 of *LNCS*, pages 36–53. Springer, 2009. 35

[Asa02] Tomoyuki Asano. A revocation scheme with minimal storage at receivers. In Y. Zheng, editor, *Asiacrypt 2002*, volume 2501 of *LNCS*, pages 433–450. Springer, 2002. 60, 64

[Asa03] Tomoyuki Asano. Reducing storage at receivers in SD and LSD broadcast encryption schemes. In K. Chae and M. Yung, editors, *Information Security Applications*, volume 2908 of *LNCS*, pages 401–412, 2003. 60, 64

[Asa04] Tomoyuki Asano. Secure and insecure modifications of the subset difference broadcast encryption scheme. In *Information Security and Privacy*, volume 3108 of *LNCS*, pages 12–23, 2004. 60, 64, 70

[AT83] Selim G. Akl and Peter D. Taylor. Cryptographic solution to a problem of access control in a hierarchy. *ACM Trans. Comp. Sys.*, 1(3):239–248, August 1983. 61, 65, 72

[AVL62] Georgii M. Adelson-Velskii and Evgenii M. Landis. An algorithm for the organization of information. *Proc. USSR Academy of Sciences*, 146:263–266, 1962. 119

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *Eurocrypt 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004. Full version at `http://eprint.iacr.org/2004/171`. 42

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In R. Cramer, editor, *Eurocrypt 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, 2005. Full version at `http://eprint.iacr.org/2005/015`. 37, 38, 66, 100

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *Eurocrypt 2000*, volume 1807 of *LNCS*, pages 259–274. Springer, 2000. Full version at `http://cseweb.ucsd.edu/~mihir/papers/musu.html`. 46, 47

[BBW06] Adam Barth, Dan Boneh, and Brent Waters. Privacy in encrypted content distribution using private broadcast encryption. In *Financial Cryptography and Data Security*, volume 4107 of *LNCS*, pages 52–64. Springer, 2006. 75

[BCK98] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *STOC 1998*, pages 419–428. ACM Press, 1998. Full version at `http://eprint.iacr.org/1998/009`. 12, 24, 49

[BD95] Mike Burmester and Yvo Desmedt. A secure and efficient conference key distribution system. In *Eurocrypt '94*, volume 950 of *LNCS*, pages 275–286. Springer, 1995. 51

[BD97] Mike Burmester and Yvo G. Desmedt. Efficient and secure conference-key distribution. In *Security Protocols*, volume 1189 of *LNCS*, pages 119–129. Springer, 1997. 51

[BD05] Mike Burmester and Yvo Desmedt. A secure and scalable group key exchange system. *Inf. Proc. Letters*, 94(3):137–143, May 2005. 16, 27, 51, 103

[BDJR97] Mihir Bellare, Anand Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *FOCS '97*, pages 394–403. IEEE, 1997. 107

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *Crypto '98*, volume 1462 of *LNCS*, pages 26–45. Springer, 1998. Full version at `http://www.di.ens.fr/~pointche/pub.php`. 89, 93

[Ber91] Shimshon Berkovits. How to broadcast a secret. In *Eurocrypt '91*, volume 547 of *LNCS*, pages 535–541. Springer, 1991. 66

[BF99] Dan Boneh and Matthew Franklin. An efficient public key traitor tracing scheme. In Michael Wiener, editor, *Crypto' 99*, volume 1666 of *LNCS*, pages 338–353, 1999. 17, 28, 80

[BF01] Dan Boneh and Matthew Franklin. Identity based encryption from the weil pairing. In J. Kilian, editor, *Crypto 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, 2001. Full version at `http://eprint.iacr.org/2001/090`. 48

[BG03] Olivier Billet and Henri Gilbert. A traceable block cipher. In *Asiacrypt 2003*, volume 2894 of *LNCS*, pages 331–346. Springer, 2003. 14, 26

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In V. Shoup, editor, *Crypto 2005*, volume 3621 of *LNCS*, pages 258–275. Springer, 2005. 53, 67, 69, 75, 86, 87, 88, 94, 98, 99

[BK05] Dan Boneh and Jonathan Katz. Improved efficiency for cca-secure cryptosystems built using identity-based encryption. In A. J. Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 87–103. Springer, 2005. 109

[BKR00] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of the cipher block chaining message authentication code. *J, Computer and System Sciences*, 61(3):362–399, December 2000. A preliminary version appeared in Crypto '94. 34, 39

[BMW05] Xavier Boyen, Qixiang Mei, and Brent Waters. Direct chosen ciphertext security from identity-based techniques. In Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels, editors, *ACM CCS '05*, page 320–329. ACM Press, 2005. 99

[BN08] Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertext. In *ACM CCS '08*, pages 455–470. ACM Press, 2008. 17, 18, 28, 29, 80, 82

[Bon13] Dan Boneh. Broadcast encryption: Recent progress and open problem, 2013. slides from a lecture at Bar Ilan Winter School, Feb. 2013, available at `http://crypto.biu.ac.il/winterschool2013/Boneh-broadcast.pdf`. 69

[BP04] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In Matthew Franklin, editor, *Crypto 2004*, volume 3152 of *LNCS*, page 273–289. Springer, 2004. 101

[BP08] Olivier Billet and Duong Hieu Phan. Efficient traitor tracing from collusion secure codes. In Reihaneh Safavi-Naini, editor, *ICITS*, volume 5155 of *LNCS*, pages 171–182. Springer, 2008. 17, 18, 28, 29, 79, 82

[BP09] Olivier Billet and Duong Hieu Phan. Traitors collaborating in public: Pirates 2.0. In A. Joux, editor, *Eurocrypt 2009*, volume 5479 of *LNCS*, pages 189–205. Springer, 2009. 79, 125

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS '93*, pages 62–73. ACM Press, 1993. 34

[BR94] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In *Eurocrypt '94*, volume 950 of *LNCS*, pages 92–111. Springer, 1994. 49

[BR04] Mihir Bellare and Phillip Rogaway. Code-based game-playing proofs and the security of triple encryption. Cryptology ePrint Archive, Report 2004/331, 2004. `http://eprint.iacr.org/`. 39

[BS98] Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data. *IEEE Trans. Inf. Th.*, 44(5):1897–1905, 1998. A preliminary version appeared in Crypto '95. 56, 121

[BS02] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2002. Full version at `http://eprint.iacr.org/2002/080`. 68

[BSW06] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In S. Vaudenay, editor, *Eurocrypt 2006*, volume 4004 of *LNCS*, pages 573–592. Springer, 2006. Full version at `http://eprint.iacr.org/2006/045`. 81, 82

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, 2011. Full version at `http://eprint.iacr.org/2010/543`. 76

[BT08] Oded Blayer and Tamir Tassa. Improved versions of Tardos' fingerprinting scheme. *Designs, Codes and Cryptography*, 48(1):79–103, July 2008. 131

[BW06] Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. In *ACM CCS '06*, pages 211–220. ACM Press, 2006. Full version at `http://eprint.iacr.org/2006/298`. 67, 74, 81, 82, 86, 88

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In S. P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 535 – 554. Springer, 2007. Full version available at Cryptology ePrint Archive `http://eprint.iacr.org/2006/287`. 13, 24

[CFGP06] Olivier Chevassut, Pierre-Alain Fouque, Pierrick Gaudry, and David Pointcheval. The twist-augmented technique for key exchange. In M. Yung, editor, *PKC 2006*, volume 3958 of *LNCS*, pages 410–426. Springer, 2006. Full version at `http://eprint.iacr.org/2005/061`. 117

[CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In Y. G. Desmedt, editor, *Crypto '94*, volume 839 of *LNCS*, pages 257–270. Springer, 1994. 13, 25, 76

[CFNP00] Benny Chor, Amos Fiat, Moni Naor, and Benny Pinkas. Tracing traitors. *IEEE Trans. Inf. Th.*, 46(3):893–910, May 2000. 76

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *JACM*, 51(4):557–594, July 2004. A preliminary version appeared in STOC '98, p. 209–218. 34

[CGI+99] Ran Canetti, Juan Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas. Multicast security: A taxonomy and some efficient constructions. In *INFOCOM '99*, volume 2, pages 708–716, 1999. 61

[Cha11] Ana Charpentier. *Identification de copies de documents multimédia grâce aux codes de Tardos*. PhD thesis, Université de Rennes 1, 2011. 57

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *Eurocrypt 2004*, volume 3027 of *LNCS*, pages 207–222. Springer, 2004. 49

[CLT13] Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *Crypto 2013*, LNCS. Springer, 2013. Full version at `http://eprint.iacr.org/2013/183`. 68

[CMN99] Ran Canetti, Tal Malkin, and Kobbi Nissim. Efficient communication-storage tradeoffs for multicast encryption. In *Eurocrypt '99*, volume 1592 of *LNCS*, pages 459–474. Springer, 1999. 61

[CPP05] Hervé Chabanne, Duong Hieu Phan, and David Pointcheval. Public traceability in traitor tracing schemes. In R. Cramer, editor, *Eurocrypt 2005*, volume 3494 of *LNCS*, pages 542–558. Springer, 2005. 13, 25, 80

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Crypto '98*, volume 1462 of *LNCS*, pages 13–25. Springer, 1998. 46, 47, 117

[CS03] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Computing*, 33(1):167–226, 2003. `http://shoup.net/papers`. 12, 23, 47

[CT90] Gerald C. Chick and Stafford E. Tavares. Flexible access control with master keys. In Gilles Brassard, editor, *Crypto '89*, volume 435 of *LNCS*, pages 316–322. Springer, 1990. 61, 64

[CVM10] Michelle Cotton, Leo Vegoda, and David Meyer. IANA guidelines for IPv4 multicast address assignments. RFC 5771 (Best Current Practice), Mar 2010. 11, 23

[Dam92] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *Crypto '91*, volume 576 of *LNCS*, page 445–456. Springer, 1992. 101

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000. A preliminary version appeared at STOC' 91. 49

[DdP11] Paolo D'Arco and Angel L. Perez del Pozo. Fighting pirates 2.0. In J. Lopez and G. Tsudik, editors, *ACNS 2011*, volume 6715 of *LNCS*, pages 359–376. Springer, 2011. 79

[Del08] Cécile Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In K. Kurosawa, editor, *Asiacrypt 2007*, volume 4833 of *LNCS*, pages 200–215. Springer, 2008. 70, 88, 98

[Den02] Alexander W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In *Asiacrypt 2002*, volume 2501 of *LNCS*, pages 100–109. Springer, 2002. 34

[Den06] Alexander W. Dent. The hardness of the dhk problem in the generic group model. Cryptology ePrint Archive, Report 2006/156, 2006. `http://eprint.iacr.org/`. 101

[DF03a] Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption for stateless receivers. In J. Feigenbaum, editor, *DRM 2003*, volume 2696 of *LNCS*, pages 61–80. Springer, 2003. 66, 69, 75, 78

[DF03b] Yevgeniy Dodis and Nelly Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In Y. G. Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 100–115. Springer, 2003. Full version at `http://eprint.iacr.org/2003/095`. 67, 85, 87, 88, 98

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Trans. Inf. Th.*, 22(6):644–654, November 1976. 50, 117

[DK05] Yevgeniy Dodis and Jonathan Katz. Chosen-ciphertext security of multiple encryption. In J. Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 188–209. Springer, 2005. 69, 109

[DLB07] Yvo Desmedt, Tanja Lange, and Mike Burmester. Scalable authenticated tree based group key exchange for ad-hoc groups. In *Financial Cryptography and Data Security*, volume 4886 of *LNCS*, pages 104–118. Springer, 2007. 51

[DLN96] Cynthia Dwork, Jeff Lotspiech, and Moni Naor. Digital signets — self-enforcing protection of digital content. In *STOC 1996*, pages 489–498. ACM, 1996. 79

[DLP+12] Jean Paul Degabriele, Anja Lehmann, Kenneth G. Paterson, Nigel P. Smart, and Mario Strefler. On the joint security of encryption and signature in EMV. In *CT-RSA 2012*, volume 7178 of *LNCS*, pages 116–135. Springer, 2012. 18

[DP08] Yvo Desmedt and Duong Hieu Phan. A CCA secure hybrid damgård's elgamal encryption. In *ProvSec*, volume 5324 of *LNCS*, page 68–82. Springer, 2008. 101

[DPP07] Cécile Delerablée, Pascal Paillier, and David Pointcheval. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In T. Takagi et al., editor, *Pairing 2007*, volume 4575 of *LNCS*, pages 39–59. Springer, 2007. 13, 24, 53, 69, 86, 94, 95, 98, 104, 105, 107

[ECR12] ECRYPT II yearly report on algorithms and keysizes (2011-2012) revision 1.0. Technical Report ICT-2007-216676, ECRYPT II, September 2012. available at `http://www.ecrypt.eu.org/documents/D.SPA.20.pdf`. 40, 73

[Elg85] Taher Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Th.*, 31(4):469–472, July 1985. 46, 117

[FN94] Amos Fiat and Moni Naor. Broadcast encryption. In D. R. Stinson, editor, *Crypto '93*, volume 773 of *LNCS*, pages 480–491. Springer, 1994. 11, 23, 60, 62, 103

[FNP07] Nelly Fazio, Antonio Nicolosi, and Duong Hieu Phan. Traitor tracing with optimal transmission rate. In *ISC*, volume 4779 of *LNCS*, pages 71–88. Springer, 2007. 17, 28, 79, 82

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In H. Imai and Y. Zheng, editors, *PKC '99*, volume 1560 of *LNCS*, pages 53–68. Springer, 1999. 49

[FOPS04] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. *J. Crypto*, 17(2):81–104, March 2004. 49

[FP12] Nelly Fazio and Irippuge Milinda Perera. Outsider-anonymous broadcast encryption with sublinear ciphertexts. In *PKC 2012*, volume 7293 of *LNCS*, pages 225–242. Springer, 2012. 75

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Crypto '86*, volume 263 of *LNCS*, pages 186–194. Springer, 1987. 34

[FST10] David Freeman, Michael Scott, and Edlyn Teske. A taxonomy of pairing-friendly elliptic curves. *J. Crypto*, 23(2):224–280, April 2010. Full version at `http://eprint.iacr.org/2006/372`. 41

[FT99] Amos Fiat and Tamir Tassa. Dynamic traitor tracing. In Michael Wiener, editor, *Crypto' 99*, volume 1666 of *LNCS*. Springer, 1999. 17, 28, 83

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *Eurocrypt 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, 2013. Full version at `http://eprint.iacr.org/2012/610`. 68

[Gol01] Oded Goldreich. *Foundations of Cryptography Volume 1 - Basic Tools*. Cambridge University Press, 2001. 43

[Gol04] Oded Goldreich. *Foundations of Cryptography - Volume 2 Basic Applications*. Cambridge University Press, 2004. 42, 43, 44

[GPS06] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165, 2006. `http://eprint.iacr.org/`. 41

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS '06*, pages 89–98. ACM Press, 2006. Full version at `http://eprint.iacr.org/2006/309`. 13, 24, 48, 75

[GR04] Craig Gentry and Zulfikar Ramzan. RSA accumulator based broadcast encryption. In K. Zhang and Y. Zheng, editors, *Information Security Conference — ISC 2004*, volume 3225 of *LNCS*, pages 73–86. Springer, 2004. 60, 65, 70

[GST04] Michael T. Goodrich, Jonathan Z. Sun, and Roberto Tamassia. Efficient tree-based revocation in groups of low-state devices. In M. Franklin, editor, *Crypto 2004*, volume 3152 of *LNCS*, pages 511–527. Springer, 2004. 60, 65

[GSY99] Eli Gafni, Jessica Staddon, and Yiqun Lisa Yin. Efficient methods for integrating traceability and broadcast encryption. In *CRYPTO' 99*, volume 1666 of *LNCS*, pages 372–387. Springer, 1999. 79

[GW09] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In A. Joux, editor, *Eurocrypt 2009*, volume 5479 of *LNCS*, pages 171–188. Springer, 2009. Full version at `http://eprint.iacr.org/2008/268`. 15, 26, 53, 68, 70, 74, 85, 86, 88, 97, 98

[HK07] Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In *Crypto 2007*, volume 4622 of *LNCS*, pages 553–571. Springer, 2007. Full version at `http://eprint.iacr.org/2007/288`. 47

[HMLS10] Dennis Hofheinz, John Malone-Lee, and Martijn Stam. Obfuscation for cryptographic purposes. *J. Crypto*, 23(1):121–168, January 2010. 14, 26

[HS02] Dani Halevy and Adi Shamir. The LSD broadcast encryption scheme. In M. Yung, editor, *Crypto 2002*, volume 2442 of *LNCS*, pages 145–161. Springer, 2002. 60, 63, 65, 73

[HT98] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In Hugo Krawczyk, editor, *Crypto '98*, volume 1462 of *LNCS*, page 408–423. Springer, 1998. 101

[IP08] Vincenzo Iovino and Giuseppe Persiano. Hidden-vector encryption with groups of prime order. In S. D. Galbraith and K. G. Paterson, editors, *Pairing 2008*, volume 5209 of *LNCS*, pages 75–88. Springer, 2008. 13, 24

[JL07] Hongxia Jin and Jeffery Lotspiech. Renewable traitor tracing: A trace-revoke-trace system for anonymous attack. In *ESORICS*, volume 4734 of *LNCS*, pages 563–577. Springer, 2007. 14, 17, 25, 28, 83, 131

[JLN04] Hongxia Jin, Jeffery Lotspiech, and Stefan Nusser. Traitor tracing for prerecorded and recordable media. In *ACM DRM '04*, pages 83 – 90. ACM Press, 2004. 83

[JS13] Tibor Jager and Jörg Schwenk. On the analysis of cryptographic assumptions in the generic ring model. *J. Crypto*, 26(2):225–245, April 2013. A preliminary verision appeared at Asiacrypt 2009. 35

[KAF+10] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen Lenstra, Emmanuel Thomé, Joppe Bos, Pierrick Gaudry, Alexander Kruppa, Peter Montgomery, Dag Arne Osvik, Herman te Riele, Andrey Timofeev, and Paul Zimmermann. Factorization of a 768-bit rsa modulus. Cryptology ePrint Archive, Report 2010/006, 2010. `http://eprint.iacr.org/`. 36

[KD98] Kaoru Kurosawa and Yvo Desmedt. Optimum traitor tracing and asymmetric schemes. In *Eurocrypt '98*, volume 1403 of *LNCS*, pages 145–157. Springer, 1998. 80

[KK11] Łukasz Krzywiecki and Mirosław Kutyłowski. Coalition resistant anonymous broadcast encryption scheme based on PUF. In *Trust and Trustworthy Computing*, volume 6740 of *LNCS*, pages 48–62. Springer, 2011. 75

[KKK06] Łukasz Krzywiecki, Przemysław Kubiak, and Mirosław Kutyłowski. A revocation scheme preserving privacy. In *Information Security and Cryptology*, volume 4318 of *LNCS*, pages 130–143. Springer, 2006. 75

[KKN08] Łukasz Krzywiecki, Mirosław Kutyłowski, and Maciej Nikodem. General anonymous key broadcasting via lagrangian interpolation. *IET Information Security*, 2(2):79–84, September 2008. 75

[KL08] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography.* Chapman & Hall/CRC, 2008. 47

[KP07] Aggelos Kiayias and Serdar Pehlivanoglu. Pirate evolution: How to make the most of your traitor keys. In A. Menezes, editor, *Crypto 2007*, volume 4622 of *LNCS*, pages 448–465. Springer, 2007. 14, 25

[KP09] Aggelos Kiayias and Serdar Pehlivanoglu. Tracing and revoking pirate rebroadcasts. In *ACNS 2009*, volume 5536 of *LNCS*, pages 253–271. Springer, 2009. 17, 28

[KP10] Aggelos Kiayias and Serdar Pehlivanoglu. *Encryption for Digital Content*, volume 52 of *Advances in Information Security*. Springer, 2010. 56

[KPT04] Yongdae Kim, Adrian Perrig, and Gene Tsudik. Tree-based group key agreement. *ACM Trans. Inf. Syst. Sec.*, 7(1):60–96, May 2004. 15, 27, 103, 117

[KSNW03] Hartono Kurnio, Rei Safavi-Naini, and Huaxiong Wang. A group key distribution scheme with decentralised user join. In *SCN 2003*, volume 2576 of *LNCS*, pages 146–163. Springer, 2003. 16, 27, 104

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In N. Smart, editor, *Eurocrypt 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, 2008. 14, 25

[KY02a] Aggelos Kiayias and Moti Yung. Breaking and repairing asymmetric public-key traitor tracing. In *DRM 2002*, 2002. 80

[KY02b] Aggelos Kiayias and Moti Yung. On crafty pirates and foxy tracers. In T. Sander, editor, *Security and Privacy in Digital Rights Management — ACM CCS-8 Workshop DRM 2001*, volume 2320 of *LNCS*, pages 22–39. Springer, 2002. 79

[KY02c] Aggelos Kiayias and Moti Yung. Traitor tracing with constant transmission rate. In *Eurocrypt 2002*, volume 2332 of *LNCS*, pages 450–465. Springer, 2002. 17, 28, 79, 125, 129

[KY07] Jonathan Katz and Moti Yung. Scalable protocols for authenticated group key exchange. *J. Crypto*, 20(1):85–113, 2007. 16, 27, 49, 51, 107

[Lin06] Yehuda Lindell. A simpler construction of cca2-secure public-key encryption under general assumptions. *J. Crypto*, 19(3):359–377, 2006. A preliminary version appeared at Eurocrypt 2003. 49

[LPQ12] Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Anonymous broadcast encryption: Adaptive security and efficient constructions in the standard model. In *PKC 2012*, volume 7293 of *LNCS*, pages 206–224. Springer, 2012. 54, 75

[LS98] Michael Luby and Jessica Staddon. Combinatorial bounds for broadcast encryption. In K. Nyberg, editor, *Eurocrypt '98*, volume 1403 of *LNCS*, pages 512–526. Springer, 1998. 65

[LS08] David Lubicz and Thomas Sirvent. Attribute-based broadcast encryption scheme made efficient. In S. Vaudenay, editor, *Africacrypt 2008*, volume 5023 of *LNCS*, pages 325–342. Springer, 2008. 75

[LSW10] Allison Lewko, Amit Sahai, and Brent Waters. Revocation systems with very small private keys. In *IEEE Symposium on Security and Privacy 2010*, 2010. Full Version at `http://eprint.iacr.org/2008/309`. 68, 75

[Man09] Mark Manulis. Group key exchange enabling on-demand derivation of peer-to-peer keys. In *ACNS 2009*, volume 5536 of *LNCS*, pages 1–19. Springer, 2009. Full version at `http://www.manulis.eu/pub.html`. 15, 16, 27, 103, 104

[May04] Alexander May. Computing the rsa secret key is deterministic polynomial time equivalent to factoring. In *Crypto 2004*, volume 3152 of *LNCS*, pages 213–219. Springer, 2004. 35

[Mog84] Jeffrey C. Mogul. Broadcasting internet datagrams. RFC 919 (Standard), Oct 1984. 11, 23

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *Crypto 2002*, volume 2442 of *LNCS*, pages 111–126. Springer, 2002. 34

[NK05] Ryo Nojima and Yuichi Kaji. Secure, efficient and practical key management scheme in the complete-subtree method. *IEICE Trans. Fundamentals*, E88-A(1):189–194, 2005. 60, 64

[NNL01] Dalit Naor, Moni Naor, and Jeff Lotspiech. Revocation and tracing schemes for stateless receivers. In J. Kilian, editor, *Crypto 2001*, volume 2139 of *LNCS*, pages 41–62. Springer, 2001. Full version at `http://eprint.iacr.org/2001/059`. 11, 13, 15, 23, 24, 26, 60, 61, 70, 76, 78, 82, 85, 103, 108, 114, 118

[NP01] Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. In Y. Frankel, editor, *FC 2000*, volume 1962 of *LNCS*, pages 1–20. Springer, 2001. 13, 25, 66, 67, 79, 82

[NSS99] David Naccache, Adi Shamir, and Julien P. Stern. How to copyright a function? In *PKC '99*, volume 1560 of *LNCS*, pages 188–196. Springer, 1999. 55, 79

[Nui10] Koji Nuida. A general conversion method of fingerprint codes to (more) robust fingerprint codes against bit erasure. In *ICITS*, volume 5973 of *LNCS*, pages 194–212. Springer, 2010. 57

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC 1990*. ACM Press, 1990. 49

[NYO08] Takashi Nishide, Kazuki Yoneyama, and Kazuo Ohta. Attribute-based encryption with partially hidden encryptor-specified access structures. In S.M. Bellovin, editor, *ACNS 2008*, volume 5037 of *LNCS*, pages 111–129. Springer, 2008. 14, 25

[OP01] Tatsuaki Okamoto and David Pointcheval. REACT: Rapid enhanced-security asymmetric cryptosystem transform. In D. Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 159–174. Springer, 2001. 49

[OT11] Tatsuaki Okamoto and Katsuyuki Takashima. Achieving short ciphertexts or short secret-keys for adaptively secure general inner-product encryption. In *CANS 2011*, volume 7092 of *LNCS*, pages 138–159. Springer, 2011. 76

[PP04] Duong Hieu Phan and David Pointcheval. On the security notions for public-key encryption schemes. In C. Blundo and S. Cimato, editors, *SCN 2004*, volume 3352 of *LNCS*, pages 33–46. Springer, 2004. 90

[PPS11] Duong Hieu Phan, David Pointcheval, and Mario Strefler. Security notions for broadcast encryption. In *ACNS 2011*, volume 6715 of *LNCS*, pages 377–394. Springer, 2011. full version available from the author's webpage. 18, 52, 98

[PPS12a] Duong Hieu Phan, David Pointcheval, and Mario Strefler. Decentralized dynamic broadcast encryption. In *SCN 2012*, volume 7485 of *LNCS*, pages 166–183. Springer, 2012. 19

[PPS12b] Duong Hieu Phan, David Pointcheval, and Mario Strefler. Message-based traitor tracing with optimal ciphertext rate. In *Latincrypt 2012*, 2012. full version available from the author's webpage. 19

[PPSS12] Duong-Hieu Phan, David Pointcheval, Siamak F. Shahandashti, and Mario Strefler. Adaptive CCA broadcast encryption with constant-size secret keys and ciphertexts. In *Information Security and Privacy*, volume 7372 of *LNCS*, pages 308–321. Springer, 2012. 98, 99, 100, 101

[PPSS13] Duong-Hieu Phan, David Pointcheval, Siamak F. Shahandashti, and Mario Strefler. Adaptive CCA broadcast encryption with constant-size secret keys and ciphertexts. *Int. J. Inf. Sec.*, 12, 2013. 18

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Crypto*, 13(3):361–396, 2000. 127

[PSNT06] Duong Hieu Phan, Reihaneh Safavi-Naini, and Dongvu Tonien. Generic construction of hybrid public key traitor tracing with full-public-traceability. In *ICALP 2006*, volume 4052 of *LNCS*, pages 264–275, 2006. 79

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS 1999*, pages 543–553. IEEE, 1999. 49

[SF07] Ryuichi Sakai and Jun Furukawa. Identity-based broadcast encryption. Cryptology ePrint Archive, Report 2007/217, 2007. `http://eprint.iacr.org/`. 70

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Crypto '84*, volume 196 of *LNCS*, pages 47–53. Springer, 1985. 47

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Eurocrypt '97*, volume 1233 of *LNCS*, pages 256–266. Springer, 1997. 34

[Sho00] Victor Shoup. Using hash functions as a hedge against chosen ciphertext attack. In B. Preneel, editor, *Eurocrypt 2000*, volume 1807 of *LNCS*, pages 275–288. Springer, 2000. 47, 105

[Sho02] Victor Shoup. OAEP reconsidered. *J. Crypto*, 15(4):223–249, September 2002. 49

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/`. 39

[Sir07] Thomas Sirvent. Traitor tracing scheme with constant ciphertext rate against powerful pirates. In *Proc. of Workshop on Coding and Cryptography (WCC'07)*, pages 379–388, 2007. Full version at `http://eprint.iacr.org/2006/383`. 17, 28, 57

[SNW03a] Reihaneh Safavi-Naini and Yejing Wang. Sequential traitor tracing. *IEEE Trans. Inf. Th.*, 49(5):1319–1326, May 2003. A preliminary version appeared at Crypto 2000. 17, 28, 83

[SNW03b] Reihaneh Safavi-Naini and Yejing Wang. Traitor tracing for shortened and corrupted fingerprints. In *DRM 2003*, volume 2696 of *LNCS*, pages 81–100. Springer, 2003. 57

[Sta96] Markus Stadler. Publicly verifiable secret sharing. In *Eurocrypt '96*, volume 1070 of *LNCS*, pages 190–199. Springer, 1996. 37

[SW05] Amit Sahai and Brent Waters. Fuzzy identity based encryption. In R. Cramer, editor, *Eurocrypt 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, 2005. Full version at `http://eprint.iacr.org/2004/086`. 13, 24, 48

[Tar08] Gábor Tardos. Optimal probabilistic fingerprint codes. *J. ACM*, 55(2), May 2008. A preliminary version appeared in STOC '03. 57

[Tas05] Tamir Tassa. Low bandwidth dynamic traitor tracing schemes. *J. Crypto*, 18(2):167–183, April 2005. 83

[TT01] Wen-Guey Tzeng and Zhi-Jia Tzeng. A public-key traitor tracing scheme with revocation using dynamic shares. In *PKC 2001*, volume 1992 of *LNCS*, pages 207–224. Springer, 2001. 67

[Wat11] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *PKC 2011*, volume 6571 of *LNCS*, pages 53–70. Springer, 2011. Full version at `http://eprint.iacr.org/2008/290`. 75

[WGL00] Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. Secure group communications using key graphs. *IEEE/ACM Trans. Networking*, 8(1):16–30, February 2000. A preliminary version appeared at ACM SIGCOMM '98. 13, 24, 61

[WHA99] Debby M. Wallner, Eric J. Harder, and Ryan C. Agee. Key management for multicast: Issues and architectures. RFC 2627 (Informational), June 1999. 13, 24, 61

[WMGP07] Brecht Wyseur, Wil Michiels, Paul Gorissen, and Bart Preneel. Cryptanalysis of white-box DES implementations with arbitrary external encodings. In *SAC 2007*, volume 4876 of *LNCS*, pages 264–277. Springer, 2007. 14, 26

[WMS+09] Qianhong Wu, Yi Mu, Willy Susilo, Bo Qin, and Josep Domingo-Ferrer. Asymmetric group key agreement. In Antoine Joux, editor, *Eurocrypt 2009*, volume 5479 of *LNCS*, pages 153–170. Springer, 2009. 16, 27, 104

[WNR04] Pan Wang, Peng Ning, and Douglas S. Reeves. Storage-efficient stateless group key revocation. In *ISC 2004*, volume 3225 of *LNCS*, pages 25–38, 2004. 60, 65

[WQZ+11] Qianhong Wu, Bo Qin, Lei Zhang, Josep Domingo-Ferrer, and Oriol Farras. Bridging broadcast encryption and group key agreement. In D.H. Lee and X. Wang, editors, *Asiacrypt 2011*, volume 7073 of *LNCS*, pages 143–160. Springer, 2011. 16, 27, 104

[YFDL04] Danfeng Yao, Nelly Fazio, Yevgeniy Dodis, and Anna Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *ACM CCS '04*. ACM Press, 2004. Full version at `http://www.cs.brown.edu/~anna/research.html`. 69, 86, 88

[YJCK04] Eun Sun Yoo, Nam-Su Jho, Jung Hee Cheon, and Myung-Hwan Kim. Efficient broadcast encryption using multiple interpolation methods. In *Information Security and Cryptology – ICISC 2004*, volume 3506 of *LNCS*, pages 87–103. Springer, 2004. 67

Résumé

Dans cette thèse, nous étudions les définitions et les constructions en boite noire avec des instantiations efficaces pour la diffusion chiffrée et le traçage de traîtres. Nous commençons par examiner les notions de sécurité pour la diffusion chiffrée présentes dans la littérature. Comme il n'y a pas de moyen facile de les comparer, nous proposons un cadre général et établissons des relations. Nous montrons alors comment les notions existantes s'inscrivent dans ce cadre.
Ensuite, nous présentons une construction en boite noire d'un système de diffusion chiffrée dynamique décentralisée. Ce système ne repose sur aucune autorité de confiance, et de nouveaux utilisateurs peuvent joindre à tout moment. Le système satisfait la notion de sécurité la plus forte sous des hypothèses de sécurité classiques de ses composantes. Il admet une instantiation efficace qui est sûre sous la seule hypothèse DDH dans le modèle standard.
Enfin, nous donnons une construction en boite noire d'un système de traçage de traîtres à base de messages, qui permet de tracer non seulement à partir des décodeurs pirates, mais aussi à partir des tatouages numériques contenus dans un message. Notre schéma est le premier à obtenir asymptotiquement le taux d'expansion optimal de 1. Nous montrons également que vus les débits de données actuels, le schéma est déjà pratique pour les choix de valeurs usuels.

Mots-clés : cryptographie, gestion des droits numériques, chiffrement à clé publique, notions de sécurité, diffusion chiffrée, traçage de traîtres.

Abstract

In this thesis, we look at definitions and black-box constructions with efficient instantiations for broadcast encryption and traitor tracing. We begin by looking at the security notions for broadcast encryption found in the literature. Since there is no easy way to compare these existing notions, we propose a framework of security notions for which we establish relationships. We then show where existing notions fit within this framework.
Second, we present a black-box construction of a decentralized dynamic broadcast encryption scheme. This scheme does not rely on any trusted authorities, and new users can join at any time. It achieves the strongest security notion based on the security of its components and has an efficient instantiation that is fully secure under the DDH assumption in the standard model.
Finally, we give a black-box construction of a message-based traitor tracing scheme, which allows tracing not only based on pirate decoders but also based on watermarks contained in a message. Our scheme is the first one to obtain the optimal ciphertext rate of 1 asymptotically. We then show that at today's data rates, the scheme is already practical for standard choices of values.

Keywords: cryptography, digital rights management, public-key encryption, security notions, broadcast encryption, traitor tracing.