Sécurité et Efficacité des Schémas de Diffusion de Données Chiffrés

Soutenue publiquement le 19 décembre 2013

Composition du jury :

<table>
<thead>
<tr>
<th>Nom</th>
<th>Titre</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Michel Abdalla</td>
<td>CR1 CNRS, HDR (École Normale Supérieure)</td>
<td></td>
</tr>
<tr>
<td>Sébastien Canard</td>
<td>Chercheur, HDR (Orange Labs)</td>
<td></td>
</tr>
<tr>
<td>Claude Carlet (Directeur de thèse)</td>
<td>Professeur (Université Paris 8)</td>
<td></td>
</tr>
<tr>
<td>Fabien Laguillaumie</td>
<td>Professeur (Université de Lyon 1)</td>
<td></td>
</tr>
<tr>
<td>Duong Hieu Phan (Co-directeur de thèse)</td>
<td>Maitre de conférences (Université Paris 8)</td>
<td></td>
</tr>
<tr>
<td>David Pointcheval</td>
<td>DR CNRS (École Normale Supérieure)</td>
<td></td>
</tr>
</tbody>
</table>
Contents

1 Broadcast Encryption and Tracing Traitors 17
  1.1 The State of the Art .............................................. 20
    1.1.1 State of the Art of Broadcast Encryption ................. 20
    1.1.2 State of the Art of Tracing Traitors ..................... 25
  1.2 Our Contributions ................................................ 28
    1.2.1 An Optimal Scheme in the Non-black-box Tracing Model and in the
          Single-key black box Tracing Model ......................... 28
    1.2.2 New Approaches to Resist Advanced Attacks .............. 28
    1.2.3 Multi-Channel Broadcast Encryption: A Generalization of Broad-
          cast Encryption ............................................... 29
  1.3 Definitions and Security Notions .............................. 30
    1.3.1 Broadcast Encryption ....................................... 30
    1.3.2 Tracing Traitors ............................................ 33

2 An Efficient Construction of Tracing Traitors 35
  2.1 Algebraic Schemes ............................................... 37
    2.1.1 Preliminaries ................................................. 37
    2.1.2 Algebraic Schemes .......................................... 39
    2.1.3 Traceability in Broadcast Encryption ..................... 41
  2.2 An Optimal Scheme in the Non-Black-Box Tracing Model and in the Single-
        key Black Box Tracing Model .................................. 47
    2.2.1 Motivation .................................................... 48
    2.2.2 Construction .................................................. 50
    2.2.3 Security ...................................................... 51
    2.2.4 Traitor Tracing .............................................. 52
3 Resistance to Advanced Attacks on Broadcast Encryption

3.1 Tree-based Schemes and Its Variants
   3.1.1 Logical Key Tree
   3.1.2 Subset-cover Framework
   3.1.3 Combining tree-based scheme with master key technique
   3.1.4 From Secret-key Setting to Public-key Setting

3.2 Pirate Evolution Attack - PEvoA
   3.2.1 Introduction
   3.2.2 Fighting against PEvoA- The Method of Jin and Lotspiech

3.3 Pirates 2.0 Attack
   3.3.1 Introduction
   3.3.2 Fighting against Pirates 2.0

3.4 The First Identity-based Trace and Revoke Schemes - IDTR
   3.4.1 Framework
   3.4.2 Solutions
   3.4.3 Our First Construction
   3.4.4 Resistance to Pirate Evolution Attack - PEvoA
   3.4.5 Resistance to Pirates 2.0
   3.4.6 Our Second Construction

3.5 Resisting Pirates 2.0 Attack in Bounded Leakage Model
   3.5.1 Key-Leakage Resilient Revoke Scheme
   3.5.2 A Construction of Key-Leakage Resilient Revoke Scheme - KIDTR
   3.5.3 KIDTR is Immune to Pirates 2.0 in Bounded Leakage Model

4 Multi-Channel Broadcast Encryption

4.1 Introduction

4.2 Definition and Security Notions of Multi-Channel Broadcast Encryption
   4.2.1 Syntax
   4.2.2 Security Model

4.3 Constructions
   4.3.1 BGW Scheme
   4.3.2 Intuition
4.3.3 Multi-Channel Broadcast Encryption I – MCBE$_1$ . . . . . . . . . . 109
4.3.4 Multi-Channel Broadcast Encryption II – MCBE$_2$ . . . . . . . . . . 113

5 Conclusion 119

Bibliography 121
Résumé en Français

Introduction

La cryptographie est utilisée depuis les temps anciens. On trouve un grand nombre de traces historiques de son utilisation. On peut citer par exemple le bâton de Plutarque (la Scytale) ou le chiffrement par transposition de César. L'objectif principal de la cryptographie est de permettre à l'expéditeur d'un message (ou d'une masse de données) de communiquer en toute sécurité sur un canal non sécurisé avec le récepteur du message. Le message original est appelé le clair. Il est transformé en quelque chose d'incompréhensible, appelé le chiffré. L'expéditeur envoie le chiffré au récepteur via un canal commun non sécurisé. On appelle chiffrement la procédure consistant à transformer le clair, à l'aide d'une clé de chiffrement et d'un algorithme, en un chiffré, et déchiffrement la procédure consistant à retrouver le clair à partir du chiffré, connaissant la clé de déchiffrement.

Pendant une longue première période du développement de la cryptographie, le modèle standard pour dissimuler ainsi l'information était celui du chiffrement symétrique (en abrégé “la SKE”) dans lequel l'expéditeur et le récepteur s'accordent sur une clé secrète partagée (appelée clé symétrique), et le message est ensuite chiffré avec cette clé commune. L'avantage principal de la SKE est que les procédures de chiffrement et déchiffrement sont rapides. C'est la raison pour laquelle la SKE est toujours un sujet important de recherche en cryptographie, et qu'il est beaucoup utilisé en pratique. Cependant, l'inconvénient de la SKE est que l'expéditeur et le récepteur doivent se mettre en contact pour convenir d'une clé secrète partagée avant l'envoi de messages. De plus, comme chaque paire (expéditeur - récepteur) doit partager une clé spécifique, le nombre des clés augmente rapidement avec le nombre des participants du système (un million de clés dans un système de mille personnes).

Depuis les années 70, avec l'invention du chiffrement à clé publique, les inconvénients de la SKE sont en partie résolus. Le chiffrement à clé publique (en abrégé PKE) permet à un expéditeur et un récepteur d'échanger des messages sans avoir besoin de partager une clé secrète. Son principe est de donner la possibilité à chaque personne de générer un couple de clés mathématiquement liées: une pour le chiffrement, appelée la clé publique, et une autre pour le déchiffrement, appelée la clé secrète. La propriété principale est que l'on peut publier la clé publique telle que n'importe qui peut l'utiliser pour le chiffrement mais personne ne peut en déduire la clé secrète pour déchiffrer les messages. Ce principe a été introduit pour la première fois par Diffie et Hellman en 1976 et le premier schéma chiffrement clé publique pratique (le célèbre chiffrement RSA) a été ensuite proposé par Rivest, Shamir et Adleman en 1977. Cependant, l’inconvénient de la PKE est que les
perocédures de chiffrement et de déchiffrement sont très lentes, en comparaison de la SKE. Heureusement, on peut combiner une PKE avec une SKE d’une manière très efficace. Cette méthode, appelée la méthode hybride, permet de chiffrer un message long de la façon suivante: on chiffre d’abord une clé symétrique aléatoire courte (la clé de session) en utilisant une PKE et on chiffre ensuite le message avec la clé de session en utilisant une SKE. Pour le déchiffrement, le récepteur calcule la clé de session et utilise ensuite cette clé pour récupérer le message d’origine.

La découverte du chiffrement à clé publique a entamé une deuxième période de développement de la cryptographie, souvent référée comme l’ère de la cryptographie moderne. En plus de la question originale de la PKE, de nouvelles fonctionnalités ont été tour à tour introduites: la signature numérique, les preuves interactives, etc. Dans le même temps, le traitement formel de la sécurité a été aussi reconsidéré. Plus précisément, la formalisation des notions de sécurité et la modélisation des attaques ont été étudiées afin de construire des schémas comportant une preuve rigoureuse de sécurité. Cela permet d’éviter le cycle traditionnel d’étapes dans la conception d’un schéma, à savoir trouver les attaques, puis modifier le schéma pour résister aux nouvelles attaques, car il garantit que le schéma est sûr sous certaines hypothèses algorithmiques bien reconnues. En 1982, les première notions de sécurité formelle d’un schéma a été proposé par Goldwasser et Micali. Ils ont introduit la notion de sécurité sémantique qui est une version calculatoire de la sécurité parfaite dans laquelle les adversaires possèdent une puissance de calcul en temps polynomial.

De nouveaux objectifs se sont récemment ajoutés aux objectifs d’origine de la cryptographie que sont les SKE, PKE, signature numérique, preuves interactives :

- La sécurité dans le cloud computing : le calcul sur des informations chiffrées dont la primitive la plus importante est le chiffrement homomorphique.

- La cryptographie résistant aux fuites des clés (key-leakage resilience cryptography en anglais) : le schéma doit rester sûr même si une partie des informations sur les clés secrètes est révélée (il s’agit de méthodes formelles pour les contre-mesures à certains types d’attaques pratiques telles que les attaques par canaux cachés)

- Des applications multi-utilisateurs généralisant les applications point-à-point, qui ont de nombreuses applications pratiques. À titre d’exemples : dans le partage des fichiers dans un système de fichiers chiffrés, on veut pouvoir partager un fichier entre plusieurs utilisateurs en toute sécurité ; dans le système d’e-mail chiffré, on veut envoyer en toute sécurité un e-mail à plusieurs destinataires ; dans les systèmes de télévision à péage, on veut diffuser un canal à un ensemble d’abonnés, etc.

Dans cette thèse, nous nous concentrerons sur les applications multi-utilisateurs, en particulier, nous étudions la diffusion de données chiffrées (en abrégé BE). Cette primitive est introduite pour la première fois par Berkovits [Ber91], dans laquelle les auteurs considèrent le problème de diffuser un message à un groupe d’utilisateurs légitimes. Plus tard, Fiat et Naor dans [FN94] donnent un paradigme sur la diffusion de données chiffrées. Dans le cadre de BE, un expéditeur est capable de chiffrer un message à une groupe d’utilisateurs légitimes et empêcher les utilisateurs non-légitimes de récupérer les informations diffusées.
En outre, même si les utilisateurs non-légitimes se coalisent, ils ne peuvent pas obtenir de renseignement sur le contenu envoyé par l’expéditeur.

Les schémas BE peuvent être divisés en deux types: les schémas inclusifs et les schémas exclusifs (ou les schémas de révocation). Dans le premier cas, l’expéditeur détermine directement l’ensemble des utilisateurs légitimes. Dans le second, l’expéditeur spécifie l’ensemble des utilisateurs non-légitimes ; l’ensemble des utilisateurs légitimes est défini comme le complémentaire de l’ensemble des utilisateurs non-légitimes (qui sont aussi appelés les utilisateurs révoqués).

Presque toutes les applications pratiques tombent dans l’un de ces deux cas: soit le nombre d’utilisateurs légitimes est petit, soit il est grand par rapport au nombre total d’utilisateurs. Par conséquent, les deux schémas inclusif et exclusif conviennent pour de nombreuses applications pratiques. Par exemple, les schémas inclusifs conviennent aux applications concernant l’e-mail, car on a souvent besoin d’envoyer un e-mail à un petit groupe de personnes et donc le nombre d’utilisateurs légitimes est petit. En revanche, les schémas exclusifs sont un bon choix pour les applications où le nombre d’utilisateurs légitimes est grand, comme dans les systèmes de télévision à pêage. Dans ces systèmes, le centre diffuse souvent à tous les abonnés dans le système, et le nombre d’utilisateurs révoqués (qui ne paient pas d’abonnement mensuel par exemple) est relativement petit.

Plus formellement, afin d’envoyer un message à un groupe d’utilisateurs légitimes, l’expéditeur diffuse un en-tête complet qui se compose généralement de trois parties: la première, appelée l’id-entête, est une chaîne de bits qui décrit l’ensemble des utilisateurs légitimes ; la seconde, appelée clé-entête est le chiffrement d’une clé de session correspondant à l’ensemble des utilisateurs légitimes, et la dernière, appelée corps du message, est un chiffrement du message sous la clé de session. Concernant le id-tête, soit $N$ le nombre des utilisateurs dans le système, $s$ le nombre d’utilisateurs légitimes, et $r$ le nombre d’utilisateurs révoqués, on a besoin normalement de $N$ bits pour décire l’ensemble des utilisateurs légitimes (pour un schéma inclusif) ou l’ensemble des utilisateurs révoqués (pour un schéma exclusif). Cependant, presque toutes les applications pratiques correspondent à des petites valeurs de $s$ ou de $r$, donc on a juste besoin de $s \log N$ bits pour décire l’ensemble des utilisateurs légitimes ou $r \log N$ bits pour décire l’ensemble des utilisateurs révoqués. En outre, dans le cas où la différence entre deux ensembles d’utilisateurs légitimes lors de deux transmissions successives est petite et où les récepteurs conservent les mêmes états, il suffit que le id-tête inclue seulement l’indication de cette différence. Pour le corps du message, le message est chiffré sous la clé de session en utilisant un chiffrement symétrique, ce qui conduit au fait que les techniques pour optimiser la taille du corps du message dans tous les BE schémas sont les mêmes, donc id-tête et le corps du message ne sont pas comptés souvent lorsque l’on compare l’efficacité entre deux schémas BE.

En ce qui concerne la construction d’un schéma BE, il y a deux manières triviales. Dans la première méthode, chaque utilisateur est muni d’une clé symétrique individuelle par le centre. Le centre chiffre ensuite la clé de session à l’aide des clés symétriques de l’ensemble des utilisateurs légitimes. Dans ce schéma, le id-entête vaut 0, la taille de la clé de déchiffrement et le temps de déchiffrement sont constants. Cependant, ce système n’est pas pratique parce que le chiffré est linéaire en la taille de l’ensemble des utilisateurs légitimes. Dans la deuxième méthode, chaque sous-ensemble d’utilisateurs est affecté d’une
clé, et chaque utilisateur stocke les clés des sous-ensembles auxquels il appartient. Ainsi, la taille de stockage de l’utilisateur est $O(2^N)$. Dans ce schéma, la clé-entête est vide car l’identifiant définit déjà la clé de session, donc la longueur de la clé-entête est de 0 et le temps de déchiffrement est constant. Cependant, comme le stockage de l’utilisateur est trop grand, $O(2^N)$, ce schéma est également impraticable. L’objectif principal dans la conception d’un schéma BE est d’optimiser les paramètres, à savoir la clé-entête, la clé de déchiffrement, la clé de chiffrement, et la complexité en temps de chiffrement et de déchiffrement. Parmi ces paramètres, le plus important est la longueur de la clé-entête car la bande passante est généralement coûteuse et parfois très limitée. En outre, l’expéditeur - le centre de diffusion dans le système de télévision à péage - a généralement une grande puissance de calcul par rapport aux récepteurs; la taille de la clé de déchiffrement et le temps de déchiffrement doivent également être considérés lors de la conception d’un schéma BE.

Dans la diffusion de données chiffrées, un utilisateur légitime possède une boîte de déchiffrement (ou au décodeur) contenant une clé de déchiffrement. Cette boîte de déchiffrement est utilisée pour déchiffrer les données chiffrées pour récupérer les données originales. Un problème se pose si les utilisateurs légitimes peuvent extraire les clés de déchiffrement dans les boîtes de déchiffrement et ensuite les vendre à quelqu’un pour produire une boîte de déchiffrement pirate. Le traçage des traîtres, introduites par Chor et al. [CFN94, CFNP00], vise à résoudre ce problème : lors de l’obtention d’une boîte de déchiffrement pirate, l’autorité doit être en mesure d’exécuter un algorithme de traçage pour identifier au moins un utilisateur (ci-après appelé traître) qui contribue à produire cette boîte de déchiffrement pirate. Cela décourage fortement les utilisateurs de redistribuer leurs clés de déchiffrement.

L’état de L’art de Diffusion de Données Chiffrées

Dans cette partie, nous décrivons la littérature générale sur la diffusion de données chiffrées. Le schéma BE a d’abord été introduit par Berkovits [Ber91] qui construisait un schéma BE à partir d’un schéma de partage de secret. En 1993, Fiat et Naor [FN94] ont introduit une formalisation d’un schéma BE et ont proposé deux schémas à clé secrète : le premier utilise une fonction à sens unique (one-way function, OWFs) et des arbres de hachage ; le deuxième repose sur l’extraction des racines modulo un nombre composé. A ce jour, de nombreux schémas BE ont été introduits ; ils peuvent être divisés en deux catégories : les schémas algébriques et les schémas combinatoires. Dans les schémas algébriques, grâce à l’exploitation des propriétés algébriques, la taille de la clé secrète de l’utilisateur peut être rendue très courte et chaque clé secrète n’a pas besoin d’être composée des sous-clés. Cela permet aux schémas algébriques de bien résister à certains types d’attaques avancées telles que les attaques de pirates évolutifs [KP07] et Pirates 2.0 [BP09]. Dans les schémas combinatoires, en revanche, la taille de la clé secrète de l’utilisateur est souvent longue et chaque clé contient de nombreuses sous-clés. C’est la raison pour laquelle beaucoup de schémas combinatoires sont sensibles aux attaques avancées. Cependant, un avantage des schémas combinatoires est qu’ils permettent de réaliser le traçage des traîtres de manière relativement efficace.
Schémas Algébriques

Boneh-Franklin [BF99] proposent un élégant schéma algébrique à clé publique avec une procédure de traçage de traîtres déterministe. Leur construction est basée sur la représentation logarithmique discrète et ils introduisent également une technique de traçage appelée linear space tracing. Les auteurs ont donné trois modèles de traçage: traçage sans boîte-noire, traçage avec boîte-noire à une clé, et traçage avec boîte-noire général, dans lequel ils proposent deux algorithmes efficaces de traçage pour les deux premiers modèles et un algorithme inefficace de traçage pour le dernier modèle. Les faiblesses de leur schéma sont que le nombre maximum de traîtres est borné et fixé dans la phase d’installation, et que la taille du chiffré est toujours linéaire en le nombre maximal de traîtres.

Naor-Pinkas [NP00] ont proposé un schéma algébrique basé sur la technique d’interpolation polynomiale. En dehors de l’héritage de l’algorithme de traçage du schéma Boneh-Franklin, le schéma Naor-Pinkas fournit également la fonctionnalité de révocation. Dans leur schéma, la taille de la clé secrète est constante, mais la taille du chiffré est toujours linéaire dans le nombre maximal d’utilisateurs révoqués. Le nombre maximum d’utilisateurs révoqués est limité et doit être fixé dans la phase d’installation. En outre, l’utilisateur doit connaître toutes les identités des autres utilisateurs pour chiffrer ou pour déchiffrer, donc la clé de chiffrement et la clé de déchiffrement sont toutes les deux linéaires en le nombre maximal d’utilisateurs dans le système. La sécurité sémantique du schéma Naor-Pinkas est atteinte sous l’hypothèse DDH, et le traçage des traîtres est le même que dans le schéma Boneh-Franklin.

Le premier schéma algébrique BE qui obtient une taille constante pour le chiffré et la clé secrète est proposé par Boneh, Gentry, et Waters [BGW05]. Une faiblesse de leur schéma est que la complexité temporelle de l’algorithme de chiffrement/déchiffrement est très élevée, et que la taille de la clé de déchiffrement est linéaire en le nombre maximal d’utilisateurs dans le système.

Les schémas BE basés sur l’identité ont été étudiés dans [Del07] où l’auteur propose un schéma dans lequel les deux clés secrètes et le chiffré sont de taille constante, et la taille de la clé de déchiffrement est linéaire en le nombre maximal d’utilisateurs légitimes. Cependant, la sécurité du schéma Delerablee07 est prouvée en modèle de l’oracle aléatoire et sous une hypothèse forte (type-q hypothèse).

Les schéma BE dynamiques ont été d’abord étudiés dans [DPP07] par Delerablee, Paillier et Pointcheval. Leur construction utilise la technique d’inversion pour construire un schéma dynamique BE avec une taille de clé secrète courte. Toutefois, le paramètre publique est linéaire en le nombre d’utilisateurs.

Nous considérons maintenant la collusion complète des traîtres dans les schémas de traçage de traîtres. Nous pouvons facilement construire un schéma trivial contre la collusion complète des traîtres via la technique de traçage linéaire, mais la taille de chiffré est linéaire en le nombre d’utilisateurs. Les schémas de traçage des traîtres bornés tels que le schéma Boneh-Franklin ou le schéma Naor-Pinkas ne sont pas plus efficaces que le schéma trivial lorsqu’on borne simplement le nombre de traîtres par le nombre d’utilisateurs. Construire un schéma contre la collusion complète des traîtres plus efficace que le schéma
trivial est resté longtemps un problème ouvert. Ce problème a été finalement résolu par Boneh, Sahai, et Waters [BSW06]. Leur idée principale est d’utiliser les couplages (pairing en anglais) sur les groupes bilinéaires dont l’ordre est composé pour organiser les utilisateurs dans le système comme les éléments d’une matrice, puis en appliquant la technique de traçage linéaire sur la matrice. Cela permet de réduire la taille du chiffré de $N$ à $\sqrt{N}$. Cependant, le schéma BSW06 (pour faire court) n’est pas encore efficace, à cause de l’utilisation d’un groupe d’ordre composé et du fait que la taille du chiffré est encore grande.

Boneh et Waters [BW06] améliorent le schéma BSW06 en fournissant des fonctionnalités de diffusion. Leur schéma (BW06 schéma pour faire court) est une combinaison du schéma BGW et du schéma BSW06, dans laquelle ils font usage de la technique de diffusion du schéma BGW et de la technique de traçage du schéma BSW06. Par ailleurs, le schéma BW06 permet une traçabilité publique dans laquelle on n’a pas besoin d’informations secrètes pour exécuter l’algorithme de traçage.

Schémas combinatoires


L’inconvénient principal des schémas à base d’arbres stateful vient du problème que les récepteurs doivent mettre à jour leur clé de déchiffrement à chaque fois qu’un utilisateur rejoint ou quitte le système. Pour résoudre ce problème, les schémas à base d’arbres stateless sont introduits [KRS99, GSW00, NNL01]. Les schémas à base d’arbres stateless les plus connus sont les schémas NNL [NNL01], dans lesquels les auteurs introduisent le paradigme subset-cover et proposent deux schémas de traçage et de révocation efficaces. Dans le premier schéma, appelé le schéma complete subtree (CS), la clé de déchiffrement et le chiffré sont de taille $O(\log N)$ et $O(r \log \frac{N}{r})$, respectivement. Le deuxième schéma, appelé le schéma subset difference (SD), bénéficie de la propriété intéressante que la longueur du chiffré est indépendante du nombre d’utilisateurs dans le système $(2r − 1)$, et que la taille de la clé de déchiffrement est de $O(\log^2 N)$. Il s’agit de schémas bien connus, qui ont été largement utilisés dans la pratique et ont servi de base pour la conception du système de protection de contenu de diffusion (appelé AACS [AAC]) pour le HD-DVD et pour les disques Blu-ray. Depuis une dizaine d’années, plusieurs schémas importants à base d’arbres stateless ont été introduits tels que le schéma d’Halevy et Shamir [HS02] et le schéma de Asano [Asa02]. En fait, Halevy et Shamir introduisent la méthode layered subset difference (LSD), qui est une amélioration de la method SD dont les tailles de chiffré et de clé de déchiffrement sont de $d(2r − 1)$ et $O(d \log^{1+\frac{1}{2}} N)$, respectivement. Asano [Asa02]
utilise la technique d’extraction d’une clé principale pour obtenir une taille constante de la clé secrète, et utilise la méthode Power Set en conjonction avec une structure arborescente a-aire pour réduire davantage la taille du chiffre.

Dodis et Fazio [DF02] introduisent une manière de transformer les schémas NNL à clé secrète en des schémas à clé publique en utilisant l’identity-based encryption (IBE) et le hierarchical IBE (HIBE). C’est aussi le premier article à montrer comment combiner la structure arborescente avec la cryptographie basée sur l’identité. Leurs schémas sont appelés DF-CS (pour celui correspondant à la transformation du schéma CS en utilisant un schéma IBE comme le schéma Waters IBE [Wat05]) et DF-SD (pour celui correspondant à la transformation du schéma SD en utilisant un schéma HIBE comme le schéma BBG [BBG05]). Les schémas obtenus sont meilleurs que les schémas originaux CS et SD.

Récemment, sur les schémas BE, deux nouveaux types d’attaque (attaque par des pirates évolutifs [KP07] et Pirates 2.0 [BP09]) ont été proposés pour souligner certaines faiblesses des schémas du paradigme de subset-cover, ou plus généralement les schémas combinatoires. Parceque ces types d’attaques diminuent sérieusement l’efficacité des schémas ou même brisent la sécurité des schémas, il est nécessaire de construire de nouveaux schémas du paradigme de subset-cover (ou plus généralement des schémas combinatoires) qui résistent bien à ces deux types d’attaques. À cette fin, un certain nombre de schémas sont proposés. Dans [JL09], les auteurs proposent une variante du schéma SD qui se défend bien contre une attaque de pirates évolutifs, mais leur schéma diminue sérieusement l’efficacité du schéma original SD. Dans [PT11], nous proposons une variante du schéma CS qui résiste bien aux pirates évolutifs et l’efficacité de notre schéma peut être comparable au schéma original CS. Dans une suite de travaux [ZZ11, DdP11, DdP13, PT11], les auteurs proposent des méthodes qui résistent à des formes particulières de Pirates 2.0. Dans [PT13], nous allons plus loin en introduisant une méthode pour résister à l’attaque Pirates 2.0 dans le modèle de fuite bornée (bounded leakage model en anglais). Notre méthode est basée sur la cryptographie résistant aux fuites des clés.

Récemment dans [PPT13], nous introduisons un modèle généralisé pour la diffusion de données chiffrées qui est appelé diffusion de données chiffrées multi-chaîne. Dans ce contexte, le chiffreur peut chiffrer plusieurs messages pour plusieurs ensembles d’utilisateurs légitimes “en même temps”. Dans les applications pratiques telles que les systèmes de télévision à péage, le centre peut diffuser plusieurs chaînes à la fois, ce qui contribue à améliorer l’efficacité (avec un en-tête constant pour de nombreuses chaînes), et à éviter le problème de zapping dans les systèmes de télévision à péage (les utilisateurs prennent du temps lors de la commutation des chaînes). Notre schéma est une variante du schéma BGW [BGW05] et l’efficacité de notre schéma peut être comparable au schéma original.

**Nos Contributions**

Dans cette thèse, nous avons trois contributions dans le domaine de la diffusion de données chiffrées. Dans la première contribution [GNPT13], nous proposons un schéma de traçage de traître optimal dans modèle de traçage sans boîte-noire et dans le modèle de traçage avec boîte-noire à une clé. Dans la deuxième contribution [PT11, PT13], nous généralisons la notion de WIBE [ACD+06, ABC+11], ce qui nous aide à construire deux types de schémas
qui sont relativement efficaces et qui résistent bien aux deux types d’attaques récents sur les schémas BE, à savoir l’attaque de pirates évolutifs et l’attaque Pirates 2.0. Dans la troisième contribution [PPT13], nous étudions un modèle généralisé pour la diffusion de données chiffrées qui peut être appliqué en pratique, par exemple dans les systèmes de télévision à péage. Nous appelons ce modèle diffusion de données chiffrées multi-chaîne. Nous discutons maintenant plus en détails nos résultats.

**Un Schéma Optimal dans Modèle de Traçage Sans Boîte-noire et Modèle de Traçage avec Boîte-noire à une Clé**

Pour le traçage des traîtres, nous rappelons que trois modèles de traçage sont considérés: modèle de traçage sans boîte-noire, modèles de traçage avec boîte-noire à une clé, et modèle de traçage avec boîte-noire général. Le dernier modèle est évidemment le plus important car il couvre toutes les stratégies d’adversaires, mais les deux premiers modèles sont également utiles car ils couvrent de nombreux scénarios pratiques. Dans cette thèse, nous proposons une schéma optimal dans le modèle de traçage sans boîte-noire et dans le modèle de traçage boîte-noire à une clé. L’idée principale de notre méthode est de construire un schéma de telle sorte que la collusion complète des traîtres ne puisse pas produire une nouvelle clé secrète valide, donc les traîtres doivent mettre au moins une de leurs clés secrètes dans le décodeur pirate pour produire un décodeur pirate.

**Nouvelles Approches pour Résister aux Attaques Avancées**

Comme rappelé ci-dessus, deux nouveaux types d’attaque (attaque de pirates évolutifs et Pirates 2.0) sont proposés pour exploiter certaines faiblesses des schémas du paradigme de subset-cover, ou plus généralement les schémas combinatoires. Ces attaques exploitent le fait que dans ce genre des schémas, soit la clé secrète contient plusieurs sous-clés ([NNL01, HS02]) soit la clé secrète peut être utilisée pour dériver d’autres sous-clés ([Asa02, GR04, NK05, AK05]). Par conséquent, chaque sous-clé, ce qui correspond à un nœud interne dans l’arbre, peut être partagée entre plusieurs utilisateurs. Si un utilisateur ne publie des sous-clés d’un certain niveau (pour vérifier qu’il y a beaucoup d’utilisateurs qui possèdent également cette sous-clé), cet utilisateur ne peut pas être tracé alors que les pirates peuvent collecter des sous-clés différentes pour construire efficacement un décodeur pirate.

Comme les schémas du paradigme de subset-cover ont été largement mis en œuvre dans la pratique, il est nécessaire de construire les nouveaux schémas de ce paradigme qui peuvent résister à ces deux types d’attaque. Avec cette motivation, nous proposons deux schémas du paradigme de subset-cover qui résistent bien à ces deux types d’attaque. Dans notre premier schéma, nous intégrons un schéma WIBE au schéma CS pour former le premier schéma de traçage et de révocation basé sur l’identité (IDTR) [PT11]. La clé secrète dans le schéma IDTR contient une seule sous-clé et ne peut pas être utilisée pour dériver d’autres sous-clés. L’efficacité du schéma IDTR est comparable au schéma CS. Cependant, la taille de la clé secrète est encore longue (linéaire dans la profondeur de WIBE). Pour l’améliorer, nous généralisons la notion de WIBE (appel GWIBE) qui nous aide à construire une variante du schéma IDTR dont la clé secrète est de taille constante.
Dans le deuxième schéma, nous montrons d’abord que tous les schémas existant ne peuvent résister à une forme particulière de Pirates 2.0 qui attaque dans le modèle de fuite bornée. Nous étudions donc l’attaque Pirates 2.0 dans le modèle fuite bornée pour trouver un lien entre l’attaque Pirates 2.0 et la résistance aux fuites des clés [PT13]. Dans cet objectif, nous définissons d’abord formellement l’attaque Pirates 2.0 dans le modèle de fuite bornée, puis proposons un schéma qui résiste à l’attaque Pirates 2.0 dans ce modèle.

**Diffusion multi-chaîne de Données Chiffrées**

L’objectif de la diffusion de données chiffrées est de permettre à un centre d’envoyer un contenu à un groupe arbitraire d’utilisateurs. Actuellement, les systèmes les plus efficaces fournissent des entêtes de taille constante, qui encapsulent des clés de session. Cependant, dans la pratique, et notamment dans le système de télévision à péage, les fournisseurs doivent envoyer des entêtes différentes pour des groupes d’utilisateurs différents. Chaque entête est donc spécifique à chaque groupe, une pour chaque chaîne: en conséquence, l’entête globale est linéaire en le nombre de chaînes. En outre, lorsque l’on veut zapper et regarder une autre chaîne, on doit obtenir une nouvelle en-tête et la déchiffrer pour obtenir la nouvelle clé de session: soit les entêtes sont envoyées assez fréquemment soit on doit stocker toutes les entêtes même si on regarde une seule chaîne car sinon, le temps de zapping devient trop long.

Afin de surmonter ces faiblesses, nous étudions un modèle généralisé pour la diffusion de données chiffrées que nous appelons **diffusion multi-chaîne de données chiffrées**. Dans ce contexte, le centre peut chiffrer plusieurs messages à plusieurs ensembles d’utilisateurs légitimes “**en même temps**”. On peut faire face aux problèmes ci-dessus en encapsulant plusieurs clés de session correspondant à différentes chaînes dans un seul en-tête. Nous construisons cette nouvelle primitive et nous l’appelons la **diffusion multi-chaîne de données chiffrées - MCBE**: on peut s’attendre à un surcoût global moins important et à un temps de zapping court parce que le décodeur dispose déjà des informations pour déchiffrer toutes les chaînes disponibles. Notre schéma est une variante du schéma BGW [BGW05], avec un en-tête global de taille constante, indépendamment du nombre de chaînes.

**Nos publications:**


- Duong Hieu Phan and Viet Cuong Trinh. **Key-Leakage Resilient Revoke Scheme Resisting Pirates 2.0 in Bounded Leakage Model.** *In A. Youssef, A.*
Chapter 1

Broadcast Encryption and Tracing Traitors

Contents

1.1 The State of the Art ........................................... 20
  1.1.1 State of the Art of Broadcast Encryption ............. 20
  1.1.2 State of the Art of Tracing Traitors .................. 25
1.2 Our Contributions ............................................. 28
  1.2.1 An Optimal Scheme in the Non-black-box Tracing Model and in
        the Single-key black box Tracing Model ................. 28
  1.2.2 New Approaches to Resist Advanced Attacks .......... 28
  1.2.3 Multi-Channel Broadcast Encryption: A Generalization of
        Broadcast Encryption .................................... 29
1.3 Definitions and Security Notions ........................... 30
  1.3.1 Broadcast Encryption ................................... 30
  1.3.2 Tracing Traitors ....................................... 33

Cryptography has been used for a very long time and a lot of traces of using cryptography in ancient times are found such as scytale transposition cipher and Caesar cipher. The primary goal of cryptography is to allow senders and receivers to communicate securely over an insecure channel. When using cryptography, the sender starts with ordinary data which is called plaintext and relies on it to produce something unreadable, which is called ciphertext. The sender then sends the ciphertext to the receiver via a public channel. We call encryption when the sender turns the plaintext into a ciphertext and we call decryption when the receiver turns the ciphertext back to the plaintext.

During the first period of development of cryptography, the standard model for hiding information is symmetric encryption (SKE for short) in which both the sender and the receiver agree on a shared secret key (called symmetric key), and the message is then encrypted under this shared secret key. The main advantage of SKE is the speed of encryption and decryption. This is the reason why until now SKE is still a main topic of research in cryptography, and it is used a lot in practice. However, the inconvenience
of SKE is that the sender and the receiver have to meet to agree on a shared secret key before sending messages. Moreover, it requires the use of a different key for each pair of individuals, which leads to millions of keys in an organization with thousands of people, and thus quickly becomes unmanageable.

In the 70’s, the invention of public key encryption remedied the inconveniences of SKE. In fact, public key encryption (PKE for short) allows a sender and a receiver to send messages without the need for sharing any secret key. Its main principle is to give the possibility for anyone to generate a mathematically related pair of keys, one for encryption which is called public key and one for decryption which is called secret key. Its main property is that one can publish the public key so that anyone can use it to do the encryption, and without the secret key nobody can decrypt. This principle was introduced for the first time by Diffie and Hellman in 1976, and later in 1977 the first practical public key encryption scheme was proposed by Rivest, Shamir, and Adleman (the well-known RSA scheme). The main disadvantage of PKE is that it is relatively slow, in comparison with SKE. Fortunately, one can combine a PKE with a SKE in a very efficient way. This is called hybrid method in which, in order to encrypt a long message, one first encrypts a random short symmetric key (session key) by using a PKE and then encrypts the long message with the session key by using a SKE. The receiver first computes the session key and then uses it to recover the long original message.

The discover of public-key encryption starts the second period of development of cryptography often referred to as the era of modern cryptography. Besides PKE, more and more new primitives have been introduced such as digital signatures, interactive proofs, etc. Along with these new primitives, formal treatment for the security has been extensively considered. Specifically, formalization of the security notions and attack models have been proposed in order to design schemes with proofs of security (the security of the scheme can be proved). This avoids the traditional cycle in designing a scheme, namely finding attacks and then modifying the scheme to resist the new attacks, because it guarantees that the scheme is secure under some well accepted assumptions. In 1982, the formal security notions for an encryption scheme was proposed by Goldwasser and Micali, where the notion of semantic security was introduced, which is a computational version of perfect security with regards to poly-time adversary.

Besides the main objectives of cryptography (SKE, PKE, digital signature, interactive proofs), other primitives or functionalities are required for new practical applications such as:

- Security in cloud computing: computation on encrypted information to which the most important primitive is fully homomorphic encryption.

- Key-leakage resilient cryptography: users can leak part of their secret keys while the scheme remains secure (countermeasure to some kinds of practical attack such as side channel attacks)

- Multi-user applications: generalization from one-to-one to one-to-many with many corresponding practical uses. For example, in file sharing in encrypted file systems one wants to securely share a file with many users, in encrypted mail systems one wants to securely send an email to many receivers, in social networks one wants
to privately send a message to a group, in pay-TV systems the center wants to broadcast a channel to a set of subscribers, etc.

In this thesis, we focus on multi-user applications, specifically we deal with broadcast encryption (BE for short). Broadcast encryption is introduced for the first time by Berkovits [Ber91], in which the authors consider the issue of broadcasting a message to any subset of legitimate users. Later on, Fiat and Naor in [FN94] give a formal broadcast encryption paradigm. In the context of BE, a sender is able to encrypt a message to a target set of legitimate users and to prevent non-legitimate users from recovering the broadcasted information. Moreover, even if all non-legitimate users collude, they don’t obtain any information about the content sent by the sender.

The BE schemes can be divided into two types of schemes: inclusive schemes and exclusive schemes (or revoking schemes). In the former, the sender directly determines the set of legitimate users (the target set). In the later, the sender directly determines the set of non-legitimate users, and the target set is then defined as all but non-legitimate users (who are also called the revoked users).

Almost all practical applications fall into the two cases: either the number of users in the target set is small or the number of users in the target set is large. Therefore, both inclusive schemes and exclusive schemes can be applicable to many practical applications. For example, inclusive schemes are suitable for email application. In fact, each time one only needs to send an email to a small group of people, therefore the target set is quite small. In contrast, exclusive schemes are a good choice for applications in which the target set is quite large such as in pay-TV systems. In pay-TV systems, the center often broadcasts to all subscribers in the system, and the number of revoked users (who do not pay a monthly subscription) is relatively small.

More formally, in order to send a message to a target set, the sender broadcasts a full header which generally consists of three parts: the first one is called the id-header which is a bit string that describes the target set, the second one is called the key-header which distributes the session key corresponding to the set of legitimate users, and the last one is called the message body which is an encryption of the message under the session key. Concerning the id-header, let $N$ be the number of users in the system, $s$ be the number of users in the target set, and $r$ be the number of users in the revoked set, normally, one needs $N$ bits to describe the target set (for inclusive schemes) or the set of revoked users (for exclusive schemes). However, almost all practical applications have small $s$ or $r$, therefore one can apply a technique in which one just needs $s \log N$ bits to describe the target set or $r \log N$ bits to describe the set of revoked users, and this technique is the same in all BE schemes. Moreover, in case the difference between two successive target set (or set of revoked users) is small and the receivers are keeping state, id-header only needs to transmit this difference. Regarding the message body, the message is encrypted under the session key by using a symmetric encryption, which leads to the fact that the techniques to optimize the size of message body in all BE schemes are the same, therefore id-header and message body often do not count when comparing the efficiency of two BE schemes.

Concerning the construction of a BE scheme, there are two obvious ways. In the first way, each user is provided with a different symmetric key by the center. The center then
encrypts the session key by using the symmetric key of each user in the target set. In this scheme, the \( id \)-header is 0, decryption key and decryption time are both constants. However, this scheme is impractical because the ciphertext is linear in the size of the target set. In the second way (known as the power-set scheme), each subset of users is assigned a key and each user stores the keys of subsets that he belongs to and thus user’s storage is \( O(2^N) \). In this scheme, the \( key \)-header is empty since the \( id \)-header already defines the session key, thus the length of the \( key \)-header is 0, the decryption time is constant. However, because the user’s storage is too big, \( O(2^N) \), this scheme is also impractical. The main goal in designing a BE scheme is to optimize parameters, namely \( key \)-header, decryption keys, encryption keys, and the time complexity of encryption and decryption. In these parameters, the most important one is the length of the \( key \)-header, as bandwidth is usually costly and sometimes limited. Moreover, the sender, like the broadcast center in pay-TV systems for example, usually has large storage and computational resources as opposed to the receivers, so that the decryption key size and the decryption time should also be considered when designing a BE scheme.

In broadcast encryption, a legitimate user possesses a decryption box (or the decoder) containing a decryption key. This decryption box is used to decrypt the encrypted data to recover the real data, for examples, in the pay-TV systems or Positioning / Navigation systems. The decryption box can be a tamper resistant device (smart card), a firmware for an electronic appliance, or a software on a personal computer. The problem is what will happen if legitimate users can extract their decryption keys in their decryption boxes and then sell them for someone to produce a pirate decryption boxes? Tracing traitors, introduced by Chor et al. [CFN94, CFNP00], aims to deal with this problem. When getting a pirate decryption box, the authority should be able to run a tracing algorithm to identify at least one of the users (hereafter called traitors) who collude to produce this pirate decryption box. This strongly discourages users from redistributing their decryption keys.

If a scheme provides multicast (the sender always broadcasts to all users) and tracing traitors, we call this scheme a traitor tracing scheme. If it further provides broadcast functionality (or revocation functionality), we call this scheme a trace and revoke scheme. In this thesis, we use the general term BE scheme to refer to all these kind of schemes. In a BE scheme, each user is assigned to a different secret key (sometimes called private key). If a user needs to know the public key to decrypt a ciphertext then the decryption key contains both the secret key and the public key, otherwise the decryption key is genuinely the secret key.

1.1 The State of the Art

1.1.1 State of the Art of Broadcast Encryption

We now give an overview of prior works on broadcast encryption. BE schemes were first introduced by Berkovits [Ber91] in which he derived a one-time broadcast encryption scheme from a secret sharing scheme. In 1993, Fiat and Naor [FN94] formalized BE schemes and proposed two private-key BE schemes, the first one using one-way functions (OWFs)
and hash trees, the second one based on root extraction modulo a composite number. Many BE schemes have been introduced and they can be divided into two categories: algebraic schemes and combinatorial schemes. In algebraic schemes, algebraic properties allows the size of the secret key to remain short and does not need to contain sub-keys. This helps algebraic schemes to resist well some kinds of advanced attack such as pirate evolution attack [KP07] and Pirates 2.0 [BP09]. In combinatorial schemes, in contrast, the size of the secret key is often long and contains many sub-keys. This is the reason why many combinatorial schemes are susceptible to advanced attacks. However, one advantage of combinatorial schemes is that they allow tracing traitors in a relatively efficient way.

**Algebraic Schemes**

Boneh and Franklin [BF99] proposed an elegant algebraic public key traitor tracing scheme with a deterministic tracing procedure (Boneh-Franklin scheme for short). Their construction is based on the discrete logarithmic representation for multicast and they also introduce the linear space tracing technique for tracing traitors. The secret key size of the Boneh-Franklin scheme is constant, but the weaknesses of their scheme are that the size of the ciphertext is always linear in the maximum number of traitors, and that the maximum number of traitors is fixed at the beginning (during the setup phase). The semantic security and the tracing security of their scheme hold under the decisional Diffie-Hellman assumption.

Naor and Pinkas [NP00] proposed an algebraic scheme based on the polynomial interpolation technique (Naor-Pinkas scheme for short). Apart from inheritance of the tracing algorithm from the Boneh-Franklin scheme, the Naor-Pinkas scheme also provides revocation functionality. In their scheme, the size of the secret key is constant, but the size of the ciphertext is still linear in the maximum number of revoked users. The maximum number of revoked users is limited and needs to be fixed in the setup phase. Moreover, users need to know all the identities of other users to encrypt or decrypt, thus the encryption key and the decryption key are both linear in the maximum number of users in the system. The semantic security of the Naor-Pinkas scheme holds under the DDH assumption, and tracing traitors is the same as in the Boneh-Franklin scheme.

Boneh, Gentry, and Waters [BGW05] proposed the first broadcast encryption scheme (BGW scheme for short) which achieves the nice property that both the ciphertext and the secret key are of constant size, but the weakness of their scheme is that the encryption and decryption time complexity are high. Moreover, the public key is linear in the maximum number of users in the system ($N$), and users need the public key to perform encryption and decryption (the maximum number of users in the system is fixed in the setup phase). This might become impractical in some settings since the storage/time complexity may be excessive for low-cost devices. The authors also proposed a second scheme in which they trade-off encryption key size, decryption key size, and ciphertext size so that all are of sub-linear size ($O(\sqrt{N})$). They also gave a construction which achieves CCA security based on the CHK-transform [CHK04] and uses a secure signature scheme or MAC. An improvement of the BGW scheme was proposed by Phan et al. [PPSS12], in which the scheme further provides dynamic property and achieves CCA security by using a hash function instead of a signature.
Identity-based broadcast encryption was first investigated by Abdalla et al. [ADML+07], Delerablee [Del07], and later on by Gentry and Waters [GW09], and Attrapadung and Libert [AL10]. In [Del07], the author proposed a scheme (Delerablee07 scheme for short) supporting revocation in which both the secret key and the ciphertext are of constant size. The public key is shorter than the one in the BGW scheme in the sense that it is only linear in the maximum number of users in the target set. Moreover, while the the maximum number of users in the BGW scheme is polynomial in the size of the security parameter, the maximum number of users in [Del07] can be exponential in the size of the security parameter (the identity-based property). However, the security of the Delerablee07 scheme can only be proven in the random oracle model and under a stronger assumption (q-type assumption in comparison to decisional bilinear Diffie-Hellman exponent assumption in the BGW scheme).

Dynamic Broadcast Encryption was first investigated in [DPP07] by Delerablee, Paillier and Pointcheval. In their scheme, the total number of users do not need to be fixed during the setup and when a new user joins the system, the encryption key just needs $O(1)$ updates. Moreover, new users can also decrypt all previously distributed messages. Their scheme is thus partially dynamic and is suitable for some applications, like DVD encryption.

We now consider the full collusion of traitors in traitor tracing schemes. It was a challenging open problem to construct a scheme that resists full collusion of traitors. We can easily construct a trivial scheme which achieves full collusion of traitors using the linear tracing technique, but the ciphertext size is linear in the number of users. All bounded traitor tracing schemes such as the Boneh-Franklin scheme or the Naor-Pinkas scheme are not more efficient than the trivial scheme when setting the bound of the number of traitors as the total number of users in the system. It was open for a long time to construct a scheme supporting full collusion of traitors that is more efficient than the trivial scheme. This problem was solved by Boneh, Sahai, and Waters [BSW06]. Their main idea is to use pairing in a composite order group to organize the users in the system as elements of a matrix and then to apply the linear tracing technique to the matrix. This helps reducing the ciphertext size from $N$ to $\sqrt{N}$. However, their scheme (BSW06 scheme for short) is still far from practical, because of the use of parings on composite order groups and because the ciphertext size is still large.

Boneh and Waters [BW06] further improved the BSW06 scheme by providing broadcast functionality. Their scheme (BW06 scheme for short) is a combination of the BGW scheme and the BSW06 scheme, in which they make use of the broadcast technique from the BGW scheme and the tracing technique from the BSW scheme. Moreover, the BW06 scheme supports public traceability in which one does not need secret information to run the tracing algorithm. This scheme is very interesting from a theoretical point of view as it is the first algebraic trace and revoke scheme supporting full collusion of traitors with sub-linear ciphertext size. It is, however, hard to implement in practice, because of the use of parings on composite order groups, and the large ciphertext size and secret key size (both of them are always of $O(\sqrt{N})$ independently of the effective size of collusion is, i.e. even if there are effectively few traitors).
Combinatorial schemes

Combinatorial schemes include tree-based schemes and code-based schemes. Tree-based schemes were independently proposed by Wallner et al. [WHA99] and Wong et al. [WGL98]. The authors employ a logical key tree structure in which for a system of \( N \) users they construct a tree with \( N \) leaves (each user is assigned a leaf), and for each node in the tree, a key is drawn at random. All receivers have to update their decryption keys by using information given in broadcasted messages every time a user is revoked or a new user joins the system (the whole update takes \( 1 + 2 \log N \) messages). These are thus stateful tree-based schemes and can only be implemented in practical applications where the set of privileged users is rarely updated. Several methods which combine tree-based structure with other techniques have been proposed: Sherman and McGrew [SM03] combined a tree-based structure with a one-way function, Canetti et al. [CMN99a] combined a tree-based structure with a pseudo-random generator, and Kim et al. [KPT00] combined a tree-based structure with a Diffie-Hellman key exchange scheme. These schemes aim to reduce the cost of revocation (from \( 1 + 2 \log N \) messages to \( 1 + \log N \) messages).

In order to solve the re-keying problem, stateless tree-based schemes have been introduced [KRS99, GSW00, NNL01]. Among these schemes, the NNL schemes [NNL01] seem to be the most practical ones. In fact, they have been used as a basis to design the widely spread content protection system for HD-DVDs and Blu-ray disks called AACS [AAC]. In [NNL01], the authors introduce the subset-cover framework and propose two efficient private-key trace and revoke schemes. The first one is called complete subtree (CS) scheme whose decryption key and ciphertext are of \( O(\log N) \) and \( O(r \log \frac{N}{r}) \) size, where \( N \) is the number of users in the system and \( r \) is the number of revoked users. The second one, called subset difference (SD) scheme, enjoys an interesting property: the length \((2r - 1)\) of the ciphertext is independent of the number of users in the system and the size of the decryption key is \( O(\log^2 N) \). Some important improvements of the NNL schemes were also proposed by Halevy and Shamir [HS02] and by Goodrich, Sun, and Tamassia [GST04]. Halevy and Shamir introduced the layered subset difference (LSD) method which is an improvement of the SD method in which the size of the ciphertext and the size of the decryption key are \( d(2r - 1) \) and \( O(d \log^{1+\frac{1}{d}} N) \) respectively. In [GST04], by combining the technique in CS and SD, the authors further proposed a scheme which gives ciphertexts of size \( 4r - 2 \) and decryption keys of size \( O(\log N) \) at the expense of the decryption time \( (O(N) \) in comparison with \( O(\log N) \) in SD).

Several interesting stateless tree-based schemes with constant size secret key are also proposed [Asa02, GR04, NK05, AK05]. In [Asa02], the author makes use of the master key technique [CT90] to reduce the length of the secret key to \( O(1) \). Later on, in [GR04] the authors improve the Asano’s result on the storage of public information and the computational complexity at the receiver by making use of the accumulator-based method of Akl and Taylor [AT83]. They also introduce layers in a similar fashion as in the LSD method. Nojima and Kaji [NK05] improve the CS method by using a trapdoor one-way permutation to reduce the size of the secret key to \( O(1) \), Asano and Kamio further reduce the computational cost of the Nojima-Kaji construction by using a Rabin tree. However, all constant size secret key schemes above have the disadvantage that these schemes only achieve the weak security notion of key intractability but not of key indistinguishability.
1.1. THE STATE OF THE ART

All tree-based schemes mentioned above are private-key BE schemes. However, the public-key BE schemes are desirable for some applications in practice. Dodis and Fazio [DF02] thus introduce a way to transform the NNL schemes from a private-key setting to a public-key setting by using IBE and HIBE. This is also the first paper to show how to combine tree-based structure and identity-based cryptography. Their schemes are denoted by DF-CS (transforming CS scheme to public-key setting by using an IBE scheme such as Waters IBE in [Wat05]) and DF-SD (transforming SD scheme to public-key setting by using an HIBE scheme such as the BBG scheme [BBG05]). The resulting schemes are competitive with the original CS and SD schemes. In [PT11], we go further by integrating WIBE scheme [ACD+06, ABC+11] into the CS scheme and this results in the first identity-based trace and revoke scheme (IDTR scheme for short) whose ciphertext size and secret key size are comparable to the that of original scheme CS. We further generalize the notion of WIBE (GWIBE for short) which helps us to improve the secret key size to be constant. [DdP11, DdP13] show a way to combine tree-based structure with algebraic schemes by integrating the Naor-Pinkas scheme [NP00] into the CS scheme or the SD scheme. However, because they integrate the Naor-Pinkas scheme into the CS or the SD scheme, the resulting schemes are not as efficient as the original schemes (CS or SD) but also suffer from the same weakness as the Naor-Pinkas scheme.

A large number of combinatorial schemes are based on the use of some special codes. These are codes that have a tracing property such as fingerprinting codes or collusion secure codes. The idea is to combine many sub-schemes supporting a small number of users to construct a general scheme supporting a large number of users. Each user in the general scheme, who is assigned a codeword, receives a set of sub-keys, each of which corresponds to a sub-scheme. The assignment of sub-keys for each user depends on his codeword. This technique allows to reduce tracing traitors in the general scheme from tracing traitors in the sub-schemes and therefore tracing traitors is very efficient. In fact, the efficiency of the general black box tracing is the main advantage of code-based schemes. Several elegant code-based schemes were introduced: Kiayias and Yung [KY02b] propose a scheme with constant rate, [BP08, BN08] achieve constant size ciphertexts, [ADML+07] show a way to combine code-based schemes with identity-based cryptography. Recently, Ngo, Phan, and Poincheval [NPP13] introduced the first trace and revoke scheme based on codes. One of the main shortcomings of code-based schemes remains that the size of secret key is large (linear in the length of codewords).

Let us now discuss about the security notions for a BE scheme. Various notions have been investigated. While the standard notion for confidentiality is always the semantic security, the attack models are categorized at various levels. The reason is that achieving both high efficiency and security against strong attacks is a very challenging problem. Selective security (selectively IND – CPA) is considered in [BGW05, Del07] in which the adversary must decide the target set that he wishes to attack before the setup phase. Adaptive security (adaptively IND – CPA) is first considered in [GW09] in which the adversary can adaptively corrupt users before receiving the challenge ciphertext. Chosen ciphertext security IND – CCA and dynamic BE scheme are respectively investigated in [BGW05] and [DPP07]. Overall, in [PPS11], the authors give a global picture of notions for BE schemes, and clarify the relations among these notions. Models for tracing traitors are considered in the Boneh-Franklin scheme, in which the general black box tracing model is the strongest tracing model. However, there is not yet any practical scheme achieving
CHAPTER 1. BROADCAST ENCRYPTION AND TRACING TRAITORS

this strongest tracing model so far.

Finally, two new types of attacks against BE schemes (pirate evolution attack [KP07] and Pirates 2.0 [BP09]) were proposed to point out some weaknesses of schemes in the subset-cover framework, or more generally of combinatorial schemes. Since these types of attacks seriously decrease the efficiency of the schemes or even break their security, it is necessary to build new schemes in the subset-cover framework (or more generally the combinatorial schemes) which can resist well these two types of attack. To this aim, a number of schemes has been proposed. In [JL09], the authors propose a method to defend against pirate evolution attack in the subset difference framework, but their method decreases seriously the efficiency of the original scheme. In [PT11], we propose a method to resist well pirate evolution attack in the complete subtree framework and the efficiency of our scheme is comparable to that of the original scheme. [ZZ11, DdP11, DdP13, PT11] propose methods which resist a particular form of Pirates 2.0 in the bounded leakage model. In [PT13], we go further by introducing a method resisting well Pirates 2.0 in the bounded leakage model. Our method is based on leakage resilient cryptography.

Recently in [PPT13], we introduced a generalized model for broadcast encryption which is called multi-channels broadcast encryption. In this context, the encryptor can encrypt several messages to several target sets “at the same time”. In practical applications such as pay-TV systems, the broadcaster can use many channels at once, which improves the efficiency (one constant header for many channels), and avoids the zapping problem in pay-TV systems (switching channels often takes time). Our scheme is a variant of the BGW scheme [BGW05] and its efficiency is comparable to that of the original scheme.

1.1.2 State of the Art of Tracing Traitors

As mentioned above, tracing traitors is a functionality of broadcast encryption. However, tracing traitors has been investigated in the literature as an independent line of research. In this thesis, a main part has been devoted to tracing traitors. We will thus discuss in detail tracing traitors in this section.

Traitor tracing scheme was first introduced by Chor et al. [CFN94, CFNP00]. Their schemes are combinatorial schemes and require both storage and decryption time of complexity \(O(t^2 \log^2 t \log(N/t))\), and encryption time of complexity \(O(t^3 \log^4 t \log(N/t))\), where \(N\) is the maximum number of users in the system and \(t\) is the maximum number of traitors. Explicit combinatorial construction, which efficiency is better than [CFN94] in case \(t\) and \(N\) are small, is later suggested by Stinson and Wei in [SW98].

In [NP98, CFNP00], the authors introduced the notion of threshold traitor tracing scheme. In this notion, the tracing algorithm can identify at least a traitor if the pirate decryption box (pirate decoder) can decrypt a valid ciphertext with probability greater than a given threshold \(\beta\). The decryption time complexity of [CFNP00] is constant, storage is \(O(t/\beta \log(t/\varepsilon))\), and encryption time complexity is linear in \(t\), where \(\varepsilon\) is the probability of successfully tracing one of the traitors.

Public traceability was put forth in [CPP05]. Unlike the classical tracing algorithms, the tracing algorithm here does not need to know the secret information of the system. Everyone can run the tracing algorithm, so that by running the tracing algorithm in many
1.1. THE STATE OF THE ART

parallel computers, the authority can reduce the time required to identify the traitors.

Tracing models were considered in [BF99], in which Boneh and Franklin proposed an algebraic public key traitor tracing scheme, and consider three tracing models: non-black-box, single-key black box, and general black box. These models are defined as follows: for the non-black-box tracing model, the tracing algorithm needs at least one decryption key embedded in the pirate decoder as input. In the model of single-key black box, the tracing algorithm requires that the pirate decoder has at most one decryption key embedded and it is assumed that the pirate decoder always uses this key to decrypt the ciphertext as in the normal decryption algorithm. In the model of general black box, the tracing algorithm should work for any strategy of the pirate decoders by only having access to pirate decoders in a black-box manner.

Boneh and Franklin also introduced the linear space tracing technique for tracing traitors, which has been subsequently used in many other works such as [NP00, TSN06, JKL09]. In this technique, we consider a set of $l$ vectors from $d_1$ to $d_l$ where $d_i \in \mathbb{F}_q^{2t}$ ($l \geq 2t + 2$). Any $2t$ set of $l$ vectors are linearly independent. The main idea of the linear space tracing technique is that given a vector $d$ which is in the linear span of at most $t$ vectors of set $l$, there exists an efficient algorithm finding the unique representation of these $t$ vectors. In fact, this algorithm is based on error correcting codes: one reduces the algorithm of finding these $t$ vectors to the algorithm of fixing $t$ error positions in a Reed-Solomon code. Therefore, in order to make use of this technique one should create a set of $l$ codewords satisfying the following properties:

1. Any $2t$ codewords among $l$ codewords are linearly independent, $l \geq 2t + 2$. Each secret key depends on a codeword.

2. Choose $l$ codewords in order to make use of the advantages of error correcting codes as above.

3. Finally, it is required that: Given $t$ secret keys of $t$ traitors, an adversary cannot build a new secret key $d$ which is not in the linear span of these $t$ secret keys.

Concerning the efficiency of their scheme, the secret key size is constant and the tracing algorithms for the two former tracing models (non-black-box, single-key black box) work quite efficiently. However, the main weakness of the Boneh-Franklin scheme is that the ciphertext size is always linear in the maximum number of traitors, and the maximum number of traitors is bounded and fixed at the beginning. Moreover, the tracing algorithm for the general black box model is inefficient, as the time complexity is exponential in the maximum number of traitors.

Some improvements have been proposed for the Boneh-Franklin scheme. Tonien and Safavi in [TSN06] and Junod, Karlov, and Lenstra in [JKL09] managed to improve the tracing algorithm for non-black-box tracing model and single-key black box tracing model. However, the ciphertext size is always linear in the maximum number of traitors, and the maximum number of traitors is bounded and fixed at the beginning. Recently in [GNPT13], we proposed an optimal public key traitor tracing scheme in the non-black-box tracing model and in the single-key black box tracing model.
As mentioned in the previous section, the first traitor tracing scheme supporting full collusion of traitors and achieving sub-linear size ciphertexts was introduced by Boneh, Sahai, and Waters [BSW06]. For tracing traitors, their main idea is to use pairings in composite order groups to organize the users in the system as elements of a matrix, and then to apply the linear tracing technique on this matrix as follows: they rely on Index Hiding property which defines that an adversary without a key $K_i$ cannot distinguish between an encryption to index $i$ (target set is $[i, N]$) and one to index $i + 1$ (target set is $[i + 1, N]$). Based on this property, the tracer can identify a secret key $K_i$ embedded in the pirate decoder if the pirate decoder can distinguish between an encryption to index $i$ and one to index $i + 1$. To this aim, the tracing algorithm can use the Chernoff bound to estimate the difference between the probability that the pirate decoder successfully decrypts an encryption to index $i$ and an encryption to index $i + 1$. The tracing algorithm checks if the difference is non-negligible, then it states that user $i$ is a traitor.

For efficiency, the secret key in their scheme is of constant size but the ciphertext is always of sub-linear size ($O(\sqrt{N})$) whatever the effective size of collusion is, i.e. even if there are few effective traitors. Their tracing algorithm is secret tracing. Boneh and Waters [BW06] further improved the BSW06 scheme by providing broadcast functionality and allowing the tracing algorithm to be run publicly.

We now consider two important combinatorial trace and revoke schemes in [NNL01]. Their constructions are based on the subset-cover framework. In this framework, the set of users $\mathcal{N}$ can be partitioned into $t$ subsets $(S_1, S_2, \ldots, S_t) \subset \mathcal{N}$. Each subset $S_i$, $1 \leq i \leq t$ is associated with a key. Each user $u \in S_i$ should be able to decrypt the ciphertext which is encrypted with the key of $S_i$, and users $v \notin S_i$ cannot decrypt, even if all of them collude. When encrypting, one first partitions $\mathcal{N}/\mathcal{R}$ into $w$ disjoint subsets $(S_{i_1}, S_{i_2}, \ldots, S_{i_w})$ such that $S_{i_1} \cup S_{i_2} \cup \cdots \cup S_{i_w} = \mathcal{N}/\mathcal{R}$, where $\mathcal{R}$ is set of revoked users. Then a session key $K$ is chosen randomly and is encrypted $w$ times under the keys corresponding to $w$ subsets $(S_{i_1}, S_{i_2}, \ldots, S_{i_w})$.

For tracing traitors, they achieve a relaxed black-box traceability: given a pirate decoder which can decrypt successfully a valid ciphertext with probability at least $p$, the tracing algorithm either outputs the identity of one of the traitors, or a subset partition that can be used to broadcast to all legitimate users and make the pirate decoder useless (i.e., successfully decrypt with negligible probability).

The important parameter of a traitor tracing scheme is the transmission rate which is defined as the ratio between the size of ciphertexts and the size of plaintexts. Kiayias and Yung in [KY02a] designs a traitor tracing scheme with constant transmission rate. All schemes mentioned above have a ciphertext size at least $t$ times longer than the embedded plaintext, which means that the transmission rate is linear in $t$, where $t$ is the maximum number of traitors. In [KY02a], Kiayias and Yung in fact prove that in the case of a long enough plaintext, it is possible to obtain ciphertexts with constant expansion rate. Their method is based on collusion-secure fingerprint codes [BS98, Tar03]. Recently, two traitor tracing schemes with constant size of ciphertexts have been proposed [BP08, BN08]. Their schemes are based on collusion-secure fingerprint codes with erasure [SNW03, Sir07]. Interestingly, the schemes with constant transmission rate [BP08, BN08, KY02a] also support an efficient tracing algorithm in the general black box tracing model. However, the main shortcoming of the code-based schemes remains that the user’s key size is large.
1.2 Our Contributions

In this thesis, we have made three contributions to the domain of broadcast encryption. In the first contribution [GNPT13], we propose an optimal public key traitor tracing scheme in the non-black-box tracing model and in the single-key black box tracing model. In the second contribution [PT11, PT13], we build two schemes which are relatively efficient and which resist well two types of recent attacks on BE schemes: pirate evolution attack and Pirates 2.0. In the third contribution [PPT13], we study a generalized model for broadcast encryption which can be applied in practice to Pay-TV systems. We call this direction multi-channel broadcast encryption. We now discuss our results in greater details.

1.2.1 An Optimal Scheme in the Non-black-box Tracing Model and in the Single-key black box Tracing Model

For tracing traitors, we recall that three tracing models are considered: the non-black-box tracing model, the single-key black box tracing model, and the general black box tracing model. The last model is evidently the most important one because it covers all adversarial strategies, but the two former models are also useful as they cover many practical scenarios. In this thesis, we propose an optimal public key traitor tracing scheme in the non-black-box tracing model and in the single-key black box tracing model. The main idea of our method is to build a scheme in such a way that the full collusion of traitors cannot produce a new valid secret key, so that traitors must put at least one of their secret keys into the pirate decoder. Interestingly, our method also answers the open question left in [BF99] in which it was mentioned: “it seems reasonable to believe that there exists an efficient public key traitor tracing scheme that is completely collusion resistant. In such a scheme, any number of private keys cannot be combined to form a new key. Similarly, the complexity of encryption and decryption is independent of the size of the coalition under the pirate’s control. An efficient construction for such a scheme will provide a useful solution to the public key traitor tracing problem.”

1.2.2 New Approaches to Resist Advanced Attacks

Recently, two new types of attack (pirate evolution attack and Pirates 2.0) were proposed, pointing out some weaknesses of subset-cover framework based schemes, or more generally of combinatorial schemes. These attacks exploit the fact that in this kind of scheme either the secret key contains many sub-keys (CS, SD, LSD, ... schemes) or the secret key can be used to derive many other sub-keys ([Asa02, GR04, NK05, AK05]). Therefore, each sub-key, which corresponds to an internal node in the tree, can be shared among many users. If a user only publishes sub-keys of a certain level (to ensure that there are many users who also share that sub-key), this user cannot be traced but pirates can efficiently collect different sub-keys to build a practical decoder.
Since schemes from the subset-cover framework have been widely implemented in practice, it is necessary to build new schemes in this framework that can resist these two types of attacks. Based on this motivation, we propose two methods to fight these attacks against the subset-cover framework. Our first method is to integrate a WIBE scheme into a CS scheme to form the first identity-based trace and revoke scheme (IDTR) [PT11]. The secret key in the IDTR scheme contains only one sub-key and cannot be used to derive any other sub-keys. The efficiency of the IDTR scheme is comparable to that of the original scheme CS. However, the size of the secret key is still long, it is in fact the same as that of the CS scheme. To improve the size of the secret key, we generalize the notion of WIBE (call GWIBE) which allows us to build a variant of IDTR scheme whose secret key is of constant size. In the second method, we show that all existing methods to fight Pirates 2.0 attack only consider a particular form of Pirates 2.0 attack in the bounded leakage model. We thus investigate the Pirates 2.0 attack in the bounded leakage model to find out a connection between Pirates 2.0 attack and leakage resilient cryptography [PT13]. To this aim, we first formally define a Pirates 2.0 attack in the bounded leakage model, and then propose a key-leakage resilient scheme (KIDTR scheme for short) which resists Pirates 2.0 attack in the bounded leakage model.

1.2.3 Multi-Channel Broadcast Encryption: A Generalization of Broadcast Encryption

The aim of broadcast encryption is to allow a broadcaster to send a content to a large arbitrary group of users at once. Currently, the most efficient schemes provide constant-size headers, that encapsulate ephemeral session keys under which the payload is encrypted. However, in practice, and namely for pay-TV, providers have to send various contents to different groups of users. Headers are thus specific to each group, one for each channel: as a consequence, the global overhead is linear in the number of channels. Furthermore, when one wants to zap to and watch another channel, one has to get the new header and decrypt it to learn the new session key: either the headers are sent quite frequently or one has to store all the headers, even if one watches one channel only. Otherwise, the zapping time becomes unacceptably long.

To overcome these weaknesses, we study a generalized model for broadcast encryption which we call multi-channel broadcast encryption. In this context, the broadcaster can encrypt several messages to several target sets “at the same time”, therefore one can deal with the above problems by encapsulating several ephemeral keys for various groups and thus various channels in a simple header. We construct such a new primitive and call it Multi-Channel Broadcast Encryption – MCBE: one can expect for a much shorter global overhead and a short zapping time since the decoder already has the information to decrypt any available channels at once. Our candidates are private variants of the BGW scheme, with a constant-size global header, independently of the number of channels.

Our publications:

• D. H. Phan and V. C. Trinh. Identity-based trace and revoke schemes. In X. Boyen and X. Chen, editors, ProvSec 2011: 5th International Conference on
1.3 Definitions and Security Notions

1.3.1 Broadcast Encryption

In this section we describe the definition for a broadcast encryption scheme. Formally, such a scheme consists of four probabilistic algorithms:

- **Setup**(\(\lambda\)): Takes as input a security parameter \(\lambda\), generates the global parameters \(\text{param}\) of the system, and returns a master secret key \(\text{MSK}\) and an encryption key \(\text{EK}\). It also creates an empty list \(\text{list}\). If the scheme allows public encryption, \(\text{EK}\) is public (known as \(\text{MPK}\)), otherwise \(\text{EK}\) is kept private, and can be seen as part of \(\text{MSK}\).

- **Extract**(\(\text{MSK}, id, \text{list}, \text{param}\)): Takes as inputs the master secret key, a user identifier \(id\), list, and \(\text{param}\). If \(id\) is in the set of possible valid users and \(id \notin \text{list}\), outputs the secret key \(SK_{id}\) for user \(id\). Identifier \(id\) is then appended to list. Else, outputs \(\bot\).

- **Encrypt**(\(\text{EK}, \text{list}, S\)): Takes as inputs \(\text{EK}\), list, and a target set \(S\), outputs a key-header \(H\) and a session key \(K \in K\), \(S\) can be seen as id-header. A broadcast consists of the full header \(\text{Hdr} = (S, H, \text{payload})\), in which the \(\text{payload}\) is the encryption of a message encrypted under the session key \(K\). The \(\text{payload}\) is known as the message body.

- **Decrypt**(\(SK_{id}, \text{Hdr}, \text{param}\)): A user \(id\) takes as inputs his secret key, \(\text{Hdr}, \text{param}\). If \(id \in S\), outputs the session key \(K\), which is then used to decrypt the \(\text{payload}\).


For the *exclusive* schemes (or revoking schemes): list is the set of all users, when encrypting one uses $\text{Encrypt}(EK, \text{list}, R)$, $R$ is the set of revoked users and the target set $S = \text{list} \setminus R$.

For correctness, we require that for all $S \in \text{list}$ and $id \in S$, if $(\text{param}, \text{MSK}, EK) \leftarrow \text{Setup}(\lambda)$, $SK_{id} \leftarrow \text{Extract}(\text{MSK}, id, \text{list}, \text{param})$, and $(\text{Hdr}, K) \leftarrow \text{Encrypt}(EK, \text{list}, S)$, one then should get $\text{Decrypt}(SK_{id}, \text{Hdr}, \text{param}) = K$.

If the Extract algorithm can be run just one time at the Setup phase only, we say that the scheme is *static*. Otherwise, we say that the scheme is *dynamic* BE, and the Extract algorithm is called the Join algorithm. The scheme is called *fully dynamic* if nothing needs to be updated after a new user joins the system, otherwise it is called *partially dynamic*. Moreover, as already mentioned above, for all the BE schemes, the $id$-header and the message body are similar, we thus define a BE scheme by focusing only on the key-header part, which can be seen as a key encapsulation mechanism (KEM). To this aim, the Encrypt algorithm above just needs to output a key-header $H$ and a session key $K \in K$. The Decrypt algorithm will return the session key $K$.

### Key Encapsulation Mechanisms and Data Encapsulation Mechanism

As mentioned above, a BE scheme includes two independent parts: a key encapsulation mechanisms (KEM) and a data encapsulation mechanism (DEM). The session key is first encrypted in the key-header by using KEM, the message is then encrypted in the message body under this session key by using DEM. Until now, all the BE schemes can be written as $\text{KEM} + \text{DEM}$. For example, considering the BE schemes such as the ones from [BGW05, GW09], the session key is a random group element and is encrypted in the key-header by using KEM. The session key is then multiplied to the message, this can be seen as the symmetric encryption DEM in which the session key is the symmetric key. If KEM is IND – CCA$_2$ secure (we formally define this notion below) then we can easily achieve a IND – CCA$_2$ secure BE scheme by binding all the components of the ciphertext together. Moreover, the DEM in all BE schemes are similar, we thus only need to concentrate on KEM when considering the efficiency and security for BE schemes.

### Security Notions

As already mentioned above, when considering the security of a BE scheme, we just need to consider the security for the KEM, where the KEM is said secure if the adversary attacking KEM cannot distinguish two keys (*key indistinguishability*, denoted IND) in a key encapsulation.

We define the security of a broadcast encryption scheme by the following game between an attacker $\mathcal{A}$ and a challenger, in the Real-or-Random setting:

**Setup($\lambda$):** The challenger runs the Setup($\lambda$) algorithm to generate the global parameters param of the system, and returns a master key MSK and an encryption key EK. It also initiates an empty list list. If the scheme is asymmetric, EK is given to $\mathcal{A}$, otherwise it is part of the MSK, and thus kept secret. Corruption and decryption lists $\Lambda_C, \Lambda_D$ are set to empty lists.

**Query phase 1.** The adversary $\mathcal{A}$ adaptively asks the following types of queries:
1. Corruption query for the user with identifier \(id\): the challenger runs \(\text{Extract}(\text{MSK}, id, \text{list}, \text{param})\) and forwards the resulting secret key to the adversary. The user \(id\) is appended to the corruption list \(\Lambda_C\);
2. Decryption query on the full header \(\text{Hdr} = (S, H, \text{payload})\) with a user identifier \(id' \in \text{list}\). The challenger answers with \(\text{Decrypt}(SK_{id'}, \text{Hdr}, \text{param})\). The pair \((\text{Hdr}, S)\) is appended to the decryption list \(\Lambda_D\);
3. Encryption query (if \(EK\) is private) for the target sets \(S\). The challenger answers with \(\text{Encrypt}(EK, \text{list}, S)\).

**Challenge.** The adversary \(\mathcal{A}\) outputs a target set \(S^*\) which he wishes to attack.

The challenger runs \(\text{Encrypt}(EK, \text{list}, S^*)\) and gets \((\text{Hdr}^*, K^*)\). Next, the challenger picks a random \(b \leftarrow \{0, 1\}\). If \(b = 1\), it picks a random \(K^* \leftarrow \mathcal{K}\). It outputs \((\text{Hdr}^*, K^*)\) to \(\mathcal{A}\).

Note that if \(b = 0\), \(K^*\) is the real key, encapsulated in \(\text{Hdr}^*\), and if \(b = 1\), \(K^*\) is random, independent of the header.

**Query phase 2.** The adversary \(\mathcal{A}\) continues to adaptively ask queries as in the first phase.

**Guess.** The adversary \(\mathcal{A}\) eventually outputs its guess \(b' \in \{0, 1\}\) for \(b\).

We say the adversary wins the game if \(b' = b\), but only if \(S^* \cap \Lambda_C = \emptyset\) and \((\text{Hdr}^*, S^*) \notin \Lambda_D\). We then denote by \(\text{Succ}^{\text{ind}}(\mathcal{A}) = \Pr[b' = b]\) the probability that \(\mathcal{A}\) wins the game, and its advantage is

\[
\text{Adv}^{\text{ind}}(\mathcal{A}) = 2 \times \text{Succ}^{\text{ind}}(\mathcal{A}) - 1
\]

\[
= \Pr[1 \leftarrow \mathcal{A}|b = 1] - \Pr[1 \leftarrow \mathcal{A}|b = 0].
\]

The security notion weaker than *key indistinguishability* is *key intractability*: it only requires that the adversary cannot compute the secret key as opposed to distinguishing it from random. There are some models of attack that we need to clarify:

- If the adversary \(\mathcal{A}\) must output the target set \(S^*\) which he wishes to attack before the **Setup** phase, the security game is called selective security. This is only meaningful in **static BE** schemes since the adversary needs to know the set of users to choose the target set before the **Setup** phase.

- If the adversary \(\mathcal{A}\) can run the corruption queries in both **Phase 1** and **Phase 2**, the security game is called adaptive security. We said the security game is adaptive-1 if \(\mathcal{A}\) can only run the corruption queries in the **Phase 1**.

- If the adversary \(\mathcal{A}\) can run the encryption queries (if \(EK\) is private), the security game is called chosen plaintext attack (denoted **IND – CPA**).

- If the adversary \(\mathcal{A}\) can run the encryption queries (if \(EK\) is private) and decryption queries, the security game is called chosen ciphertext attack (denoted **IND – CCA**). If \(\mathcal{A}\) can only run decryption queries in **Phase 1** the game is denoted **IND – CCA\(_1\)**, else it is denoted **IND – CCA\(_2\)**.
1.3.2 Tracing Traitors

A trace and revoke scheme consists of the algorithms in a general BE scheme as described in the previous section and of a new algorithm of tracing traitors. Traitor tracing scheme is a particular form of trace and revoke schemes in which the target set is always set to be the set of all users. In this section, we give the tracing security models for a traitor tracing scheme.

**General black box tracing in BE:** In the broadcast encryption scheme, we add an additional algorithm $\text{Trace}^D(S_D, EK, MSK)$: the traitor tracing algorithm interacts in a black-box manner with a pirate decoder $D$ which is built from a certain set $T$ of traitors. The algorithm takes as inputs a set $S_D \subset \text{list}$ (could be adversarially chosen), $EK$, $MSK$, and outputs a set $T_D$.

More precisely, under the conditions:

- There are at most $t$ traitors: $|T| \leq t$;
- The target set contains at least a traitor. Otherwise the pirate decoder cannot decrypt, due to the semantic security of the BE scheme;
- $D$ is "efficient" in decrypting ciphertexts, i.e. it successfully decrypts with some non-negligible probability for the target set $S_D$.

then the tracing algorithm outputs at least a traitor: $\emptyset \neq T_D \subseteq T \cap S_D$;

For the formal security game we say that general black box tracing security holds if, for any $PPT$ $A$, the probability $A$ wins is negligible in the following game:

- In the setup phase, the challenger runs $\text{Setup}(\lambda)$ algorithm to get a master key $MSK$ and gives the public parameters to the adversary $A$.
- In the query phase, $A$ may adaptively ask corrupt query for user identifier $id$, and gets $SK_{id}$; $id$ is then appended to the set $T$.
- At some point $A$ outputs a pirate decoder $D$ and a set $S_D$ which contains at least a corrupted user. The challenger then runs $\text{Trace}^D(S_D, EK, MSK) \rightarrow T_D$. We say that $A$ wins if $T_D = \emptyset$ or $T_D \notin T \cap S_D$.

The General black box model is of course the most desired model in tracing traitors because it covers all strategies of the adversary, but it is very challenging to construct an efficient scheme in this model. Therefore, two weaker models than the general black box model, namely the single-key black box model and the non-black-box model, are considered in which one can construct efficient schemes. These two models are considered because they are quite practical. In fact, in practice, the pirate often uses a legitimate box and puts a new valid secret key in this box. This case is covered by the single-key black box model. Moreover, if the tracer can also open the pirate decoder to extract the pirate secret key, it is then covered by the non-black-box model.
1.3. DEFINITIONS AND SECURITY NOTIONS

**Single-key black box tracing in BE:** The *single-key black box* model is the same as the *general black box* model, except that in the *single-key black box* model it is required that $D$ contains a valid secret key and it is assumed that the box always uses this key to decrypt the ciphertext as in the normal decryption algorithm.

**Non-black-box tracing in BE:** The *non-black-box* model is the same as in the *general black box* model, except that the tracing algorithm takes as additional input a valid secret key embedded in $D$.

Note that if the tracing algorithm does not rely on any secret information, then it is called *publicly traceable*. A weaker form of tracing traitor is also considered in [NNL01], in which the tracer does not always output at least a traitor, but rather either outputs a traitor or finds a new strategy to disable the pirate decoder while still being able to broadcast to all other legitimate users normally.

**Full access black box tracing vs minimal access black box tracing**

1. In the full access black box tracing model, the tracer can query the pirate decoder on a ciphertext $C$: if $C$ is a well-formed ciphertext, he will always receive the corresponding plaintext $M$, otherwise, the pirate decoder can return an arbitrary output (it can return a signal indicating that the ciphertext $C$ is invalid or can maliciously return a random message $M'$).

2. In the minimal access black box tracing model, the tracer queries the pirate decoder on a pair $(C, M)$ and only receives a signal: *valid* if the ciphertext $C$ is a valid encryption of $M$, *invalid* if otherwise.
Chapter 2

An Efficient Construction of Tracing Traitors

Contents

2.1 Algebraic Schemes .................................................. 37
  2.1.1 Preliminaries .................................................... 37
  2.1.2 Algebraic Schemes .............................................. 39
  2.1.3 Traceability in Broadcast Encryption ......................... 41

2.2 An Optimal Scheme in the Non-Black-Box Tracing Model
  and in the Single-key Black Box Tracing Model ................. 47
  2.2.1 Motivation ..................................................... 48
  2.2.2 Construction .................................................. 50
  2.2.3 Security ........................................................ 51
  2.2.4 Traitor Tracing ............................................... 52

In this chapter, we first consider the well-known algebraic methods to construct a broadcast encryption scheme and we also figure out the advantage and disadvantage of each method. We then focus on the traceability in broadcast encryption. Finally, we introduce our new method for tracing traitors in the non-black-box model and in the single-key black box model and we also make a comparison between our method and other known methods.

We look at the first algebraic broadcast encryption scheme in [BGW05], which achieves constant size for both the ciphertexts and the secret keys. The weaknesses of this scheme are the time complexity of the encryption, the time complexity of decryption, and the decryption key size.

We consider the first dynamic broadcast encryption scheme in [DPP07]. Their construction uses the inversion technique to achieve a dynamic broadcast encryption scheme with small secret key. However, the public parameters still contains a linear number of group elements in the number of users.
Identity-based broadcast encryption was investigated in the Delerablee07 scheme [Del07], the author proposed a scheme in which both the secret keys and the ciphertexts are of constant size, and the decryption key size is only linear in the maximum number of users in the target set. However, the security of the Delerablee07 scheme is proven in the random oracle model and under a strong assumption (q-type assumption).

In the next part, we focus on algebraic schemes supporting tracing traitors. These schemes are divided into two categories: bounded traitors schemes and full collusion of traitors schemes.

For the bounded traitors schemes, the maximum number of traitors is bounded and often fixed in the setup phase. In this category of scheme, the Boneh-Franklin scheme [BF99] is one of the most prominent one, in which their construction is based on the discrete logarithmic representation for multicast and they also introduce the linear space tracing technique for tracing traitors. The authors considered three tracing models: the non-black-box, the single-key black box, and the general black box, in which they proposed two efficient tracing algorithms for two former models and an inefficient one for the last model. The shortcomings of their scheme are that the maximum number of traitors are bounded and fixed in the setup phase, and the ciphertext size is always linear in the maximum number of traitors.

Another important bounded traitors scheme is the Naor-Pinkas scheme [NP00], which is based on threshold secret sharing for broadcasting and which also makes use of the linear space tracing technique for tracing traitors. Unlike the Boneh-Franklin scheme, their scheme provides both revocation and tracing traitors. However, their scheme still cannot deal with the shortcomings of the Boneh-Franklin scheme.

Concerning the full collusion of traitors schemes, the maximum number of traitors is unbounded. As already mentioned in the previous section, the BSW06 scheme [BSW06] is the first traitor tracing scheme supporting the full collusion of traitors and achieving sub-linear size ciphertexts. Their method is based on bilinear maps in groups of composite order, in which they organize the users in the system as elements in a matrix and then to apply the linear tracing technique in the matrix. This helps reducing the ciphertext size from $O(N)$ to $O(\sqrt{N})$. The weakness of this method is that the ciphertext size is large (always of $O(\sqrt{N})$) whatever the effective size of collusion is, i.e. even if there are few effective traitors. Moreover, their tracing algorithm needs secret information, thus it is secret tracing. Boneh and Waters [BW06] further improved the BSW06 scheme by providing broadcast functionality. Their scheme is a combination of the BGW scheme and the BSW06 scheme, in which they make use of the broadcast technique from the BGW scheme and make use of the tracing technique from the BSW06 scheme. Unlike the tracing algorithm in the BSW06 scheme, the tracing algorithm in the BW06 scheme is publicly traceable (one does not need the secret information to run the tracing algorithm).

Overall, until now the construction of a practical scheme supporting tracing traitors in the general black box model is still an open problem. Moreover, even the problem of construction of an optimal scheme supporting tracing traitors in both the non-black-box model and the single-key black box model hasn’t been solved yet. In the Boneh-Franklin scheme, they gave a suggestion to deal with the later problem by mentioning that: "it seems reasonable to believe that there exists an efficient public key traitor tracing scheme
that is completely collusion resistant. In such a scheme, any number of private keys cannot be combined to form a new key. Similarly, the complexity of encryption and decryption is independent of the size of the coalition under the pirate’s control. An efficient construction for such a scheme will provide a useful solution to the public key traitor tracing problem”. Our method, which is introduced in the last section of this chapter, gives an answer to this question. In fact, we proposed an efficient public key traitor tracing scheme whose parameters are all of constant size, and the full collusion of traitors cannot produce a new key. Our proposed scheme is moreover dynamic.

2.1 Algebraic Schemes

2.1.1 Preliminaries

We start this section by recalling the computational Diffie-Hellman (CDH) assumption and the decisional Diffie-Hellman (DDH) assumption in a cyclic group \( G \). We then discuss the bilinear maps and several hardness assumptions in bilinear groups.

CDH and DDH Assumptions

**Definition 2.1.1 (CDH Assumption)** The \((t, \varepsilon) - CDH\) assumption says that for any \( t \)-time adversary \( A \) that is given \((g, g^r, h) \in G\), its probability to output \( h^r \) is bounded by \( \varepsilon \):

\[
\text{Succ}_{cdh}(A) = \Pr[A(g, g^r, h) = h^r] \leq \varepsilon.
\]

**Definition 2.1.2 (DDH Assumption)** The \((t, \varepsilon) - DDH\) assumption says that for any \( t \)-time adversary \( A \), given \((g, g^r, h) \in G\), cannot distinguish \( h^r \) and a random value \( T \) in \( G \) with advantage greater than \( \varepsilon \):

\[
\text{Adv}_{DDH}(A) = \left| \Pr[A(g, g^r, h, h^r) = 1] - \Pr[A(g, g^r, h, T) = 1] \right| \leq \varepsilon.
\]

Bilinear Maps

Recently, the field of Pairing-Based Cryptography has been strongly developed. The main idea of it is the construction of a mapping between two useful cryptographic groups that allows for new cryptographic schemes based on the reduction of one problem in one group to a different, usually easier problem in the other group. Typically, the first of these two groups is referred to as a gap group, in which the decisional Diffie-Helman problem is easy (due to reducing to an easy problem in the second group) while the computational Diffie-Helman problem still remains hard. The known implementations of these pairings, which are based on the Weil and Tate pairings, involve fairly complex mathematical structures. Fortunately, basing on abstract bilinear maps with their group structure and mapping properties, a lot of interesting schemes have been built.

Let \( G \) and \( G_T \) denote two finite multiplicative abelian groups of large prime order \( p > 2^\lambda \) where \( \lambda \) is the security parameter. Let \( g \) be a generator of \( G \) of order \( p \). We assume that
there exists a bilinear map \( e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T \) that satisfies the three properties:
(1) Bilinearity: \( e(g^a, g^b) = e(g,g)^{ab} \), for all \( a,b \in \mathbb{Z}_p \)
(2) Non-degenerate: \( e(g^a, g^b) = 1 \) iff \( a = 0 \) or \( b = 0 \),
(3) \( e(g^a, g^b) \) is efficiently computable.

The maps \( e \) is then called an admissible bilinear maps and \((p, \mathbb{G}, \mathbb{G}_T, e(\cdot, \cdot))\) is then called a bilinear map group system. There are many mathematical structures that verify these three properties such as scalar product on euclidean space or multiplication in a ring defines a pairing \( e(x, y) = xy \). However, the structures that are relevant in cryptography should also satisfy some requirements as mentioned above: while DDH problem is easy in \( \mathbb{G} \), CDH problem remains hard in \( \mathbb{G} \). We will moreover recall in the next paragraph some other standards assumptions in bilinear maps, which have been used in broadcast encryption. The maps in which these assumptions reasonable are mainly constructed from the Weil and Tate pairings.

**Hardness Assumptions in Bilinear Groups** The bilinear Diffie-Hellman Exponent assumption (BDHE assumption for short) and the decisional bilinear Diffie-Hellman Exponent assumption (DBDHE assumption for short), which are introduced by Boneh, Boyen, and Goh [BBG05].

**Definition 2.1.3 (BDHE Assumption)** The \((t, n, \varepsilon) - \text{BDHE} \) assumption says that for any \( t \)-time adversary \( A \) that is given \((g, h, g^{a_1}, \ldots, g^{a_n}, g^{a_{n+2}}, \ldots, g^{a_{2n}}) \in \mathbb{G}^{2n+1} \), its probability to output \( e(g, h)^{a_{n+1}} \in \mathbb{G} \) is bounded by \( \varepsilon \):
\[
\text{Succ}^{\text{bdhe}}(A) = \Pr[A(g,h, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}) = e(g_{n+1}, h)] \leq \varepsilon.
\]

**Definition 2.1.4 (DBDHE Assumption)** The \((t, n, \varepsilon) - \text{DBDHE} \) assumption says that for any \( t \)-time adversary \( A \) that is given \((g, h, g^{a_1}, \ldots, g^{a_n}, g^{a_{n+2}}, \ldots, g^{a_{2n}}) \in \mathbb{G}^{2n+1} \), and a candidate to the BDHE problem, that is either \( e(g, h)^{a_{n+1}} \in \mathbb{G} \) or a random value \( T \), cannot distinguish the two cases with advantage greater than \( \varepsilon \):
\[
\text{Adv}^{\text{bdhe}}(A) = \left| \Pr[A(g,h, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, e(g_{n+1}, h)) = 1] - \Pr[A(g,h, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, T)) = 1] \right| \leq \varepsilon.
\]

**q-type assumption:** Let \((p, \mathbb{G}, \mathbb{G}_T, e(\cdot, \cdot))\) a bilinear map group system and \( g \in \mathbb{G} \) be a generator of \( \mathbb{G} \), and set \( g_T = e(g,g) \in \mathbb{G}_T \). Let \( s,n \) be positive integers and \( P, Q \in \mathbb{F}_p[X_1, \ldots, X_n]^s \) be two \( s \)-tuples of \( n \)-variate polynomials over \( \mathbb{F}_p \). Thus, \( P \) and \( Q \) are just two lists containing \( s \) multivariate polynomials each. We write \( P = (p_1, p_2, \ldots, p_s) \) and \( Q = (q_1, q_2, \ldots, q_s) \) and impose that \( p_1 = q_1 = 1 \). For any function \( h : \mathbb{F}_p \rightarrow \Omega \) and vector \((x_1, \ldots, x_n) \in \mathbb{F}_p^n \), \( h(P(x_1, \ldots, x_n)) \) stands for \((h(p_1(x_1, \ldots, x_n)), \ldots, h(p_s(x_1, \ldots, x_n))) \in \Omega^s \). We use a similar notation for the \( s \)-tuple \( Q \). Let \( f \in \mathbb{F}_p[X_1, \ldots, X_n] \). It is said that \( f \) depends on \((P, Q)\), which denotes \( f \in \langle P, Q \rangle \), when there exists a linear decomposition
\[
f = \sum_{1 \leq i,j \leq s} a_{i,j} \cdot p_i \cdot p_j + \sum_{1 \leq i \leq s} b_i \cdot q_i, \quad a_{i,j}, b_i \in \mathbb{Z}_p
\]
Let $P,Q$ be as above and $f \in \mathbb{F}_p[X_1,\ldots,X_n]$. The general decisional Diffie-Hellman Exponent problem ($\langle P,Q,f \rangle - \text{GDDHE problem for short}$) is defined as follows.

**Definition 2.1.5** ($\langle P,Q,f \rangle - \text{GDDHE}$) \cite{BGW05}.

Given $H(x_1,\ldots,x_n) \in \mathbb{G}^* \times \mathbb{G}_T^*$ as above and $T \in \mathbb{G}_T$ decide whether $T = g_T^{f(x_1,\ldots,x_n)}$.

The $\langle P,Q,f \rangle - \text{GDDHE}$ assumption says that it is hard to solve the $\langle P,Q,f \rangle - \text{GDDHE}$ problem if $f$ is independent of $(P,Q)$.

### 2.1.2 Algebraic Schemes

#### BGW Scheme \cite{BGW05}

The authors introduced a public key broadcast encryption scheme which achieves constant size for both the ciphertexts and the secret keys. In their scheme, in the setup phase first picks a random generator $g \in \mathbb{G}$ and a random scalar $\alpha \in \mathbb{Z}_p$, it then computes $g_i = g^{\alpha^i}$ for $i = 1,2,\ldots,N,N+2,\ldots,2N$. Next, it picks a random scalar $\gamma \in \mathbb{Z}_p$ and sets $v = g^{\gamma} \in \mathbb{G}$.

The public key is $\text{MPK} = (g_1,\ldots,g_N,g_{N+2},\ldots,g_{2N},v)$, whereas the secret key of user $i \in \{1,\ldots,N\}$ is $d_i = v^{\gamma^i}$. These secret keys are sent by the Extract algorithm.

In the encryption step, pick a random scalar $r \in \mathbb{Z}_p$, and set the session key $K = e(g_{N+1},g)^r$, where $e(g_{N+1},g)$ can be computed as $e(g,N,g_1)$ from $\text{EK}$.

Next, set: $\text{Hdr} = (g^{\gamma},(v \cdot \prod_{j \in S}g_{N+1-j})^r)$, and output $(\text{Hdr},K)$.

In the decryption, a decrypter with inputs are $(S,\text{Hdr},i,d_i,\text{EK})$, first parses $\text{Hdr} = (C_1,C_2)$, then computes the session key $K = e(g_i,C_2)/e(d_i \cdot \prod_{j \in S,j \neq i}g_{N+1-j+i},C_1)$.

Concerning the efficiency of their scheme, the ciphertexts are of constant size ($\text{Hdr}$ contains just 2 elements). However, in order to decrypt the decrypter needs to know all the values $g_i$, $i = 1,2,\ldots,N,N+2,\ldots,2n$. Therefore, the decryption keys is linear in the maximum number of users in the system. To overcome this weakness, the authors also introduced a second scheme which gives a trade-off between the ciphertext text size and the decryption key size, where both of them are of $\sqrt{N}$ size. For the security, their scheme achieves selectively IND – CPA under the DBDHE assumption. They also considered the chosen ciphertext attack (IND – CCA), in which they gave a construction which achieves IND – CCA security based on the CHK-transform \cite{CHK04} and using a secure signature scheme or MAC.

#### Delerable-Paillier-Pointcheval-07 Scheme \cite{DPP07}

The authors introduced the first partially dynamic public key broadcast encryption scheme whose efficiency can be comparable with others existing schemes. Their public key BE scheme is partially dynamic (which is defined in the section 1.3.1) because the encryption keys of their scheme need to be updated after a new user joins the system. Their scheme makes use of the inversion technique, in which each user is assigned to a secret element and a public label, the label is related to this secret element and is used to identify this user. When performing an encryption to a given set of revoked users, a broadcastedr manages to use the labels of these revoked users when creating the ciphertext and the session key. When decrypting, users who are not in the revoked set can use their secret keys to cancel out redundant
parts and recover the session key. For users in the revoked set, when using their secret keys to decrypt, he will create additional redundancy and therefore cannot recover the session key.

They proposed three schemes in which two schemes are public key broadcast encryption schemes and one scheme is private-key broadcast encryption scheme.

- The first public key broadcast encryption scheme achieves constant size decryption key, and the ciphertext size is linear in the number of revoked users. However, the encryption key is linear in the number of users in the system. This construction achieves partially dynamic adaptive-1 IND – CPA under the GDDHE assumption.

- The second public key broadcast encryption scheme achieves constant size ciphertext. However, the encryption key and decryption key are both linear in the number of users in the system. This construction also achieves partially dynamic adaptive-1 IND – CPA under the GDDHE assumption.

- The private-key broadcast encryption scheme achieves constant size for both encryption key and decryption key, the ciphertext size is linear in the number of revoked users. This construction achieves fully dynamic adaptive-1 IND – CPA under the GDDHE assumption.

**Delerablee-07 Scheme** [Del07] The author introduced an identity-based broadcast encryption scheme with constant size for both ciphertext and secret key. Their technique is quite similar to the inversion technique in Delerablee-Paillier-Pointcheval-07 scheme. In fact, the user’s identity in their scheme plays the role as the label in the Delerablee-Paillier-Pointcheval-07 scheme. However, unlike the Delerablee-Paillier-Pointcheval-07 scheme, this is an inclusive BE scheme, thus the broadcaster uses the identities of receivers to create the ciphertext, and if anyone uses a secret key of user outside the target set to decrypt, he will create additional redundancy. Their scheme is detailed as follows.

They use a cryptographic hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ to map a user’s identity to an element in $\mathbb{Z}_p$, master key is $(g, \gamma)$, public key is $(w, v, h, h^{\gamma}, \ldots, h^{\gamma^m})$ where $w = g^\gamma, v = e(g, h)$, the secret key of user is $g^1_{1, \gamma + H(ID)} + H(ID)$. For encryption, assume the target set is $ID_1, \ldots, ID_s$, choose a random $k$, the ciphertext and the session key $K$ are computed:

$$C_1 = w^{-k}, C_2 = h^{\prod_{i=1}^s(\gamma + H(ID_i))}, K = v^k$$

For decryption, compute

$$K = (e(C_1, h^t) \cdot e(g^{\gamma + H(ID)}, C_2))^{1_{\prod_{j=1, j \neq i} H(ID_j)}}$$

where

$$t = \frac{1}{\gamma} \cdot \left( \prod_{j=1, j \neq i}^s (\gamma + H(ID_j)) - \prod_{j=1, j \neq i}^s H(ID_j) \right)$$

As in the construction we see that their scheme achieves constant size for both the secret keys and the ciphertexts. For the public key (or the decryption key because the decrypter
needs the public key to decrypt), their scheme is better than the BGW scheme in the sense that it is only linear in the maximum number of user in the target set, it will be the same as in the BGW scheme if we need to design a scheme in which a broadcaster can broadcast a message to all users in the system. Their scheme achieves selectively IND–CPA security under the GDDHE assumption and using the random oracle.

2.1.3 Traceability in Broadcast Encryption

Bounded Traitors Schemes

Linear space tracing technique is an important technique for tracing traitors. In fact, many important public key traitor tracing schemes are already based on it for tracing traitors, we can name a few such schemes [BF99, TSN06, JKL09, NP00]. In this section, we first recall the linear space tracing technique, we then in the last section consider some important algebraic bounded traitors schemes, which gives us an overview about the existing algebraic bounded traitors schemes.

**Linear Space Tracing Technique** In this technique, we consider a set of $l$ vectors from $d_1$ to $d_l$ where $d_i \in \mathbb{F}_q^{2t}$ ($l \geq 2t + 2$). Any $2t$ set of $l$ vectors are linearly independent. The main idea of the linear space tracing technique is that given a vector $d$ which is in the linear span of at most $t$ vectors of set $l$, there exists an efficient algorithm finding the unique representation of these $t$ vectors. In fact, this algorithm is based on error correcting codes: one reduces the algorithm of finding these $t$ vectors to the algorithm of fixing $t$ error positions in a Reed-Solomon code. The reduction is detailed as follows.

- Build a matrix $A$

$$A = \begin{pmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & 2 & 3 & \cdots & l \\
1^2 & 2^2 & 3^2 & \cdots & l^2 \\
\vdots \\
1^{l-2t-1} & 2^{l-2t-1} & 3^{l-2t-1} & \cdots & l^{l-2t-1}
\end{pmatrix}$$

We see that any vectors in the span of rows of $A$ corresponding to a polynomial $f$ of degree at most $l - 2t - 1$ evaluated at the $(1, \cdots, l)$. Therefore the matrix $A$ is a representation of a Reed-Solomon codes. We choose the degree of $f$ is $l - 2t - 1$ because we want to fix $t$ error positions of a Reed-Solomon codes with the length of the vector $l$.

Denoted $b_1, \ldots, b_{2t}$ as the basis of the linear space of vectors satisfying $A\boldsymbol{x} = 0$ mod $q$. Let $b_1, \ldots, b_{2t}$ be the columns of a matrix, we get a matrix $B : l \times 2t$ as follows:

$$B = \begin{pmatrix}
| & | & | & | \\
| b_1 \ b_2 \ b_3 \ \cdots \ b_{2t} | \\
| | \ | \ | \\
| | \ | \ | \\
| | \ | \ | \\
\end{pmatrix}$$
2.1. ALGEBRAIC SCHEMES

- Let the set of \( l \) vectors above as the set of rows of the matrix \( \mathbf{B} \), then there exists a unique vector \( w \) of hamming weight at most \( t \) such that \( \overline{\mathbf{w}} \cdot \mathbf{B} = \overline{\mathbf{d}} \), where \( w \) will imply the set of \( t \) vectors. We find a vector \( v \) in which \( \tau \cdot \mathbf{B} = \overline{\mathbf{d}} \). Therefore, \( (\tau - \overline{\mathbf{w}}) \cdot B = 0 \).

- Since the rows of matrix \( \mathbf{A} \) span the space of vectors which are orthogonal to the columns of \( B \), \((\overline{\mathbf{v}} - \overline{\mathbf{w}})\) is a linear span of the rows of \( \mathbf{A} \), thus \((\overline{\mathbf{v}} - \overline{\mathbf{w}}) = (f(1), f(2), \ldots, f(l)) \). Moreover \( \overline{\mathbf{w}} \) is a vector of hamming weight at most \( t \), therefore \((f(1), f(2), \ldots, f(n)) \) equals \( \overline{\mathbf{v}} \) in all but \( t \) components.

Let \( g \) be a polynomial of degree at most \( t \) and \( g(i) = 0 \) for all \( i = 1, ..., l \), \( f(i) \neq v_i \) (where \( v_i \) is the \( i \)’th component of \( \overline{\mathbf{v}} \)). Then for all \( i = 1, ..., l \) we have \( f(i)g(i) = g(i)v_i \). The polynomial \( f \cdot g \) has degree of at most \( l - t - 1 \). Thus, we get \( l \) equations from \( i = 1 \) to \( l \) in \( l \) variables (the variables are the coefficients of the polynomials \( f \cdot g \) and \( g \), where the leading coefficient of \( g \) is 1). Let \( h \) and \( g \) be a solution where \( g \) is a non-zero polynomial of degree at most \( t \), \( h \) is a polynomial of degree at most \( l - t - 1 \). Observe that whenever \( f(i) = v_i \) (i.e. at \( l - t \) points) we have \( h(i) = g(i)v_i = g(i)f(i) \). Since \( h \) and \( f \cdot g \) are polynomials of degree at most \( l - t - 1 \) and \( h \) equals \( f \cdot g \) at \( l - t \) points, so \( h = f \cdot g \). It follows that \( f = h/g \) (because \( f, h, g \in \mathbb{F}_q(x) \)).

Therefore, in order to make use of this technique one should create a set of \( l \) codewords satisfying the following properties:

1. Any \( 2t \) codewords among \( l \) codewords are linearly independent, \( l \geq 2t + 2 \). Each secret key depends on a codeword.

2. Choose \( l \) codewords in order to make use of the advantages of error correcting codes as above.

3. Finally, it is required that: Given \( t \) secret keys of \( t \) traitors, an adversary cannot build a new secret key \( d \) which is not in the linear span of these \( t \) secret keys.

Boneh-Franklin Scheme [BF99] In their public key traitor tracing scheme, the authors use discrete log representation for broadcasting and make use of linear space tracing technique for tracking traitors. In fact, in the setup, the authority randomly chooses \( \alpha_1, \ldots, \alpha_{2t} \) then computes \( y = \prod h_i^{\alpha_i} \), the public key is \((y, h_1, \ldots, h_{2t}) \). To compute the secret key of user \( i \)’th, the authority first computes a representation of \( y \) with respect to the base \((h_1, \ldots, h_{2t}) \). To do that, they multiply a codeword \( \gamma^i = (\gamma_1, \ldots, \gamma_{2t}) \) and a random element \( \theta_i \), the secret key of user \( i \)’th is then assigned to \( \theta_i \) and \( \gamma^i \) is published.

For encryption, to broadcast a message \( M \), a broadcaster first chooses a random \( a \) then computes the ciphertext \( C = (M \cdot y^a, h_1^a, \ldots, h_{2t}^a) = (S, H_1, \ldots, H_{2t}) \).

For decryption, user \( i \)’th first computes \( U = \prod_{j=1}^{2t} H_j^{\gamma^i_j} \) then recover message \( M = S/U^{\theta_i} \).

We also note that given any representation \((\delta_1, \ldots, \delta_{2t})\) of \( y \) with respect to the base \((h_1, \ldots, h_{2t}) \), one can decrypt the ciphertext by computing \( M = S/\prod_{j=1}^{2t}(h_j^a)^{\delta_j} \).

**Tracing traitor:** They consider three tracing models: the non-black-box, the single-key black box, and the general black box.
• **Non-black-box tracing:** They make use of the linear space tracing techniques for tracing traitor. To this aim, they construct their scheme as follows.

Let \( N \) be the number of the users in the system, \( N \geq 2t + 2 \). They give a proof that given any \( t \) representations to the adversary, the adversary cannot create a new representation which is not a convex combination of these \( t \) representations. Their proof is based on the DDH assumption. Based on this proof, if \( \overline{d} \) is a representation of \( y \) found in the pirate decoder then \( \overline{d} \) must lie in the linear span of the representations \( \overline{d}_1, \ldots, \overline{d}_t \), where \( \overline{d}_1, \ldots, \overline{d}_t \) are representations corresponding to \( t \) traitors keys, or \( \overline{d} \) must be a linear span of \( t \) codewords. From that, by choosing these \( N \) codewords exactly the same as in the linear spacing technique above, the tracing algorithm that given \( \overline{d} \) as input will output \( \overline{d}_1, \ldots, \overline{d}_t \).

• **Single-key black box tracing:** Let \( \overline{d} = (\delta_1, \ldots, \delta_{2t}) \) be a representation of \( y \) in the pirate decoder, we now show that how the tracer can identify \( \overline{d} \) and then uses \( \overline{d} \) to find back \( t \) traitors as in the non-black-box tracing algorithm above.

The basic idea to identify \( \overline{d} \) here is that the tracer creates \( 2t \) invalid ciphertexts and then queries the pirate decoder to get the results. They based on the DDH assumptions prove that the pirate decoder cannot distinguish a valid ciphertext and an invalid ciphertext, therefore pirate decoder always outputs the result. By using the invalid ciphertext \( C = (S, h_{z1}^{\delta_1}, \ldots, h_{zt}^{\delta_{2t}}) \), where the non-constant vector \( z \) is chosen by the tracer, pirate decoder will respond \( M = S/\prod_{j=1}^{2t}(h_j^{\delta_j})^{\delta_j} \). With \( 2t \) queries such that the tracer can solve for \((h_{z1}^{\delta_1}, \ldots, h_{zt}^{\delta_{2t}})\), from that and with knowing the discrete log of the \( h_i \)'s base \( g \), tracer can compute \((g^{\delta_1}, \ldots, g^{\delta_{2t}})\) then identify \((\delta_1, \ldots, \delta_{2t})\).

• **General black box tracing - Black box confirmation:** They construct a black box confirmation algorithm in which if inputs are a set of suspects \( T_{\text{suspect}} \) and a set of traitors \( T_D \) and \( T_D \subseteq T_{\text{suspect}} \), the output of the algorithm is at least a traitor \( \overline{d} \). From that, by using the \( T_{\text{suspect}} \) is equal to the set of all users in the system and \( T_D \) is any set of \( t \) users they will identify all traitors. However the time complexity of the algorithm is high, in fact the running time is of \( O((\frac{N}{t})t^2) \).

In the Boneh-Franklin scheme, the maximum number of traitors (\( t \)) needs to be fixed in the setup phase because it uses the discrete log representation. The secret key is just one element but ciphertext is linear in \( t \). The semantic security of the scheme achieves IND-CPA security, and both the semantic security and the tracing security hold under the DDH assumption. For tracing traitors, two non-black-box tracing algorithm and single-key black box tracing algorithm are efficient, but general black box tracing algorithm is inefficient. The detailed comparison with other methods is provided in the table 2.1.

**Some Improvements of the Boneh-Franklin Scheme [TSN06, JKL09]** In [TSN06], the authors proposed a public key traitor tracing scheme which is similar to the Boneh-Franklin scheme, except that in the Boneh-Franklin scheme they use Reed-Solomon codes to construct the decryption keys and in [TSN06] the authors use the cover-free family to construct the decryption keys. By this way, the time complexity of the single-key
black box tracing algorithm in [TSN06] is better \( O(\frac{t^2}{\log t} \log N) \) in comparison with \( O(N) \) at the expense of the size of secret keys \( O(t) \) in comparison with \( O(1) \).

In [JKL09], the authors went further by dissecting the way Reed-Solomon codes are used to watermark the secret keys. They find out that using Berlekamp-Massey algorithm instead of the Berlekamp-Welch algorithm will improve the time complexity of non-black-box tracing algorithm and single-key black box tracing algorithm. The reason behind it is that the time complexity of the Berlekamp-Massey algorithm can be reduced if it is used only for tracing. In fact, the time complexity of non-black-box tracing algorithm and single-key black box tracing algorithm now are independent of the number of users in the system (they are of \( O(t^2) \) size).

Similar to the Boneh-Franklin scheme, some shortcomings have not been solved: the maximum number of traitors is bounded and need to be fixed in the setup phase, and the ciphertext size is still linear in \( t \). The detailed comparisons of both the above methods are provided in the table 2.1.

Polynomial Interpolation based Schemes: Naor-Pinkas-00 [NP00] Moni Naor and Benny Pinkas designed a scheme which provides both revocation functionality and tracing traitors. In their scheme, the revocation functionality is based on threshold secret sharing technique and the tracing traitors uses the linear space tracing technique. The main idea of \( t \)-out-of-\( N \) of threshold secret sharing technique is that one divides a secret into \( N \) shares and from at least \( t \) shares one can recover it but from any \( t-1 \) shares one cannot recover it. To this aim, Naor-Pinkas use a polynomial \( P \) of degree \( t-1 \) with \( P(0) = s \) (secret) and \( P(i) \) is \( i \)th share. Then, by using the polynomial interpolation, given any \( t \) shares one can easily compute \( s \), but from any \( t-1 \) shares it is easy to prove that one cannot compute \( s \).

For the revocation, in the setup the authority chooses a global polynomial \( P \) of degree \( t-1 \), then for each user \( i \) chooses and publishes a random element \( x_i \), the secret key for user \( i \) is \( P(x_i) \), all the values \( g^{P(x_i)} \) and \( g^{P(0)} \) are published. To revoke a set of users \( (1, 2, \ldots, t-1) \), a broadcaster chooses a random element \( r \) then sets the session key \( K = g^{rP(0)} \), in the ciphertext he adds \( t-1 \) elements \( g^{rP(x_j)}, j \in 1, \ldots, t-1 \). By this way, a user, who is not in the revoked set, can have enough \( t \) shares to recover the session key, and the revoked users even colluding have maximum \( t-1 \) shares. Tracing traitor uses the same technique as in [BF99].

Concerning the efficiency of their method, the secret key is of constant size, but the ciphertext is linear in the maximum number of traitors. Moreover, because of the use a global polynomial \( P \) of degree \( t-1 \), the maximum number of traitors \( (t-1) \) needs to be fixed in the setup. In order to decrypt, a decrypter (based on the polynomial interpolation) needs to know all the values \( x_i \), decryption key thus is linear in the number of users in the system. The semantic security of their scheme holds under the DDH assumption, and the tracing traitors is the same as in the Boneh-Franklin scheme. The detailed comparison of their scheme with other schemes is provided in the table 2.1.
CHAPTER 2. AN EFFICIENT CONSTRUCTION OF TRACING TRAITORS

Full Collusion of Traitors Schemes

The main shortcoming of bounded traitors schemes is that if the number of traitors is bigger than a given threshold, the security of the schemes will be broken. Full collusion of traitors schemes deal with this problem. In this section, we consider the full collusion of traitors schemes which achieves sub-linear ciphertext size so far.

BSW06 Scheme [BSW06] A trivial way to construct a BE scheme is that the ciphertext contains many components, and each component is an encryption of a message or session key to which only a corresponding receiver can decrypt. Therefore, when we modify or replace this component, only the corresponding receiver can recognize the difference and it has no effect on the capacity of decryption of other receivers. The shortcoming of this method is that the ciphertext size is large, it in fact is linear in the number of receivers. Obviously, in order to achieve a sub-linear ciphertext, a component in the ciphertext must be shared among many receivers, but the side effect of this idea is that when we want to prevent a particular receiver from decrypting, many other receivers will be affected.

Boneh-Sahai-Waters in [BSW06] introduced a method to deal with this problem, in which they assign each user to a position in a matrix, by this way each user is identified by a couple of parameters: row position and column position. All users in the same row will share the same component in the ciphertext but have different column positions, thus many users can share the same component in the ciphertext and one still can prevent a particular user from decrypting.

For concrete construction, they number users in the system from 1 to $N$, and then use the composite order groups to construct a Private Linear Broadcast Encryption (PLBE) system. The main idea of PLBE is that each user has a unique position $(x, y)$ in a matrix: $\sqrt{N} \times \sqrt{N}$, in which $x$ represents for row component and $y$ represents for column component. At each time of encryption, an encryptor will encrypt a message to a specific position $(i, j)$. A user with the position $(x, y)$ will be able to decrypt the message if $x > i$ or $x = i, y \geq j$. Obviously, the sender can choose any set $[i, N]$ as the target set, and when one encrypts to the position $(1, 1)$ then all users in the system are able to decrypt.

Notice that, when one encrypts to the position $(1, 1)$ one just needs the public key ($\text{Encrypt}_{\text{PLBE}}(\text{MPK}, M)$), otherwise one needs to know the secret parameters ($\text{TrEncrypt}_{\text{PLBE}}(\text{MSK}, M, i)$).

Tracing Traitors: The basic idea of their tracing algorithm is that they first organize the users in the system as elements in a matrix as above, then they rely on Index Hiding property to do tracing. The Index Hiding property is defined that an adversary without a key $K_i$ cannot distinguish between an encryption to index $i$ (target set is $[i, N]$) and one to index $i + 1$ (target set is $[i+1, N]$). Based on this property, the tracer can identify a secret key $K_i$ embedded in the pirate decoder if the pirate decoder can distinguish between an encryption to index $i$ and one to index $i + 1$. To this aim, the tracing algorithm can use the Chernoff bound to estimate the difference between the probability of pirate decoder to decrypt an encryption to index $i$ and an encryption to index $i + 1$, then check if the difference is non-negligible it will state that user $i$’th is a traitor.
2.1. ALGEBRAIC SCHEMES

The traitor tracing system is detailed as follows.

- **Setup** runs Setup\(_{PLBE}\) with the same parameters, and outputs MPK as the public key, MSK as the secret tracing key, and the secret keys for users are the same as in PLBE scheme.

- **Encrypt** and **Decrypt** run algorithms Encrypt\(_{PLBE}\) and Decrypt\(_{PLBE}\) respectively with the same parameters.

- **Trace\(_D\)(MSK,\(\epsilon\))** Take as inputs \(D\) is a black box oracle \(D\), tracing key MSK, and \(\epsilon\) is the probability success of \(D\), output a set of traitors.
  
  1. For \(i = 1\) to \(N + 1\), do the following:
     - Let \(\text{cnt} \leftarrow 0\)
     - Repeat the following steps \(W \leftarrow 8\lambda(N/\epsilon)^2\) times.
       i. Sample \(M\) from the finite message space at random
       ii. Let \(C \leftarrow \text{TrEncrypt}_{PLBE}(\text{MSK}, M, i)\)
       iii. Call oracle \(D\) on input \(C\) and if \(D(C) = M\) then \(\text{cnt} \leftarrow \text{cnt} + 1\)
     - Let \(\hat{p}_i \in [0, 1]\) be the fraction of times that \(D\) decrypted the ciphertexts correctly. That is, \(\hat{p}_i = (\text{cnt}/W)\)
  2. Let \(S\) be the set of all \(i \in 1, ..., N\) for which \(\hat{p}_i - \hat{p}_{i+1} \geq \epsilon/(4N)\).
  3. Output the set \(S\) as the set of guilty colluders.

Concerning the efficiency of their method, this is a public key traitor tracing scheme because one just needs the public key to encrypt a message to the position \((1, 1)\) but the tracing algorithm is secret tracing. The secret key is of constant size but ciphertext is of \(O(\sqrt{N})\) size. Their scheme enjoys the full collusion of traitors property and supports the highest level of traitor tracing model: the general black box model. Their scheme achieves IND-CPA security and general black box tracing security under the Decision 3-party Diffie-Hellman, Bilinear Subgroup Decision, and Subgroup Decision assumptions.

**BW06 Scheme** [BW06] A trace and revoke system provides both revocation and tracing traitors. To construct a trace and revoke system we can combine a revocation system with a traitor tracing system. By this way, when a traitor \(u\) is identified, trace and revoke system will revoke \(u\) by changing the target set \(S\) to \(S \setminus \{u\}\) (or adding traitor \(u\) to revoked set). However, we cannot use such a naive way to combine a revocation system and a traitor tracing system, we will show the reason why by considering the following simple attack: we consider a scenario of a trace and revoke system which has two sub-systems revocation and tracing traitors. Each user in the trace and revoke system has the same index in its two sub-systems. When one wants to encrypt a message \(M\), the message \(M\) will be randomly divided into two pieces \(M_1, M_2\) such that \(M_1 \cdot M_2 = M\). The first message is encrypted by using the revocation system, and the second one is encrypted by using the traitor tracing system. In order to decrypt a ciphertext, a decryptor needs to have secret keys in the both sub-systems. We assume that two decryptors Alice and Bob collude to produce a pirate decoder by using the Alice’s revocation key and Bob’s traitor tracing key. The tracing algorithm thus will identify Bob is as a traitor and then use the
revocation system to revoke Bob, that means Bob’s revocation key is revoked but Bob’s traitor tracing key. This leads to the fact that the pirate decoder still works and moreover in the next time the tracing algorithm still identifies Bob be as a traitor.

In order to avoid this kind of attack, a secret key of user in the trace and revoke system must be simultaneously used in both sub-systems (revocation and tracing traitors). Based on this idea, Boneh-Waters in [BW06] managed to create the secret keys of two sub-system which are intertwined by multiplying the secret keys of two sub-systems together, and make these secret keys to be randomized for each user.

For concrete construction, they first construct an ABE system by combining the broadcast system in [BGW05] with the traitor tracing system in [BSW06], then they construct their trace and revoke system based on the ABE system. In fact, in ABE system one can simultaneously broadcast a message to a subset \( S \) (using the broadcast technique of [BGW05]) and a set \([i,N]\) (using the broadcast technique of [BSW06]), therefore only users who belong to both subsets \( S \) and \([i,N]\) can decrypt the ciphertext.

**Trace and Revoke** Their trace and revoke system is detailed as follow.

1. **Setup**: Same as in ABE.

2. **Encrypt**: Using \( \text{Encrypt}_{\text{ABE}}(S, \text{MPK}, 1, M) \).

3. **Decrypt**: Same as in ABE.

4. **Tracing Algorithm**: Same as in [BSW06].

5. **Revoke**: Suppose \( T \) is a set of traitors that **Tracing Algorithm** identified. To revoke \( T \), the system uses \( \text{Encrypt}_{\text{ABE}}(S', \text{MPK}, 1, M) \) where \( S' = S \setminus T \).

Concerning the efficiency of their method, their scheme enjoys the full collusion of traitors property and supports the highest level of traitor tracing model: the general black box model, moreover their tracing algorithm is publicly traceable (because the algorithm \( \text{Encrypt}_{\text{ABE}}(S, \text{MPK}, i, M) \) just needs \( \text{MPK} \)). However, the ciphertext and secret key are always of \( O(\sqrt{N}) \) size whatever the number of the traitors in the system is. Their scheme achieves adaptive-1 IND—CPA security and general black box tracing security under the (Modified) 3-party Diffie-Hellman, Bilinear Subgroup Decision, and Diffie-Hellman Subgroup Decision assumptions.

### 2.2 An Optimal Scheme in the Non-Black-Box Tracing Model and in the Single-key Black Box Tracing Model

In this section we present our optimal public key traitor tracing scheme in the non-black-box tracing model and in the single-key black box tracing model [GNPT13].
2.2. AN OPTIMAL SCHEME IN THE NON-BLACK-BOX TRACING MODEL AND IN THE SINGLE-KEY BLACK BOX TRACING MODEL

2.2.1 Motivation

The general black box tracing model is evidently the most desired model as it covers all the possible strategies of the pirate. However, as mentioned above, all the schemes in this model are still quite impractical. Moreover, it seems a very difficult and challenging problem to achieve a practical general black box tracing, it’s of practical interest in considering the weaker models of the non-black-box tracing and the single-key black box tracing. These models are also very practical, there are many scenarios that these models are suitable, as also discussed in [TSN06, JKL09].

Let us explain some details in the context of pay-TV. In the majority of the existing systems, each user has been provided a Set-Top box (STB) and a smartcard. The secret key of the user is stored in the smartcard which has the role of decrypting the session key for every crypto-period (between 2 and 10 seconds), this session key is then transmitted to the STB for decrypting the content. The pirate always wants to minimize the cost of distribution of his solution and in practice, he really wants to try to produce new pirate smartcard to be used in already deployed STBs. It is thus necessary that these pirate smartcard are compatible with the STBs in the fields (including the legitimate STBs). As a consequence, the smartcard should preserve the functionality of the legitimate smartcard and it has to embed a pirate but valid key in the memory. It is often in reality that the authority can reverse this key in the memory of the pirate smartcard and the scenario exactly falls in the non-black-box tracing model. Even if the tracer cannot reserve the memory of the pirate smartcard, we argue that the single-key pirate tracing model is suitable. Indeed, in the modern CAS (Conditional Access Systems), the session key is delivered at the last moment so that there is only a small delay between the time the smartcard decrypts the session key and the time the decoder receive the encrypted content. Therefore, if the pirate card (which is evidently cannot more performant than a legitimate smartcard) always try to decrypt the session key with different, say two, keys, it will fails to decrypt the content in time and will give the STB the session key after the encrypted content arrive for that crypto-period. One could wonder what happens if the pirate decoder only try to detect the presence of tracing algorithm from time to time. Fortunately, the single-key black box tracing algorithms, as in Boneh-Franklin schemes and in our scheme, only need to ask just one query and the decoder is resettable in practice, this strategy of pirate does not work. All in all, we would like to argue that the non-black-box tracing model and the single-key black box tracing model, though much weaker than the general black box tracing model and cannot thus cover all the strategies of the pirate, are still very practical. In fact, there are quite a lot of interesting works that only concentrate on these models, namely [TSN06, JKL09, ADVW12].

In a theoretical point of view, it’s also a very interesting problem to consider non-black-box tracing because there is still no optimal solution, far from that, in spite of many efforts. Indeed, the Boneh-Franklin is efficient with respect to the non-black-box tracing and single-key black box tracing but its ciphertext size is still linear in the number of traitors. Tonien-Safavi in [TSN06] and the Junod-Karlov-Lenstra in [JKL09] manage to improve the tracing algorithm but the ciphertext size is always linear in the number of traitors. A side effect of this high ciphertext size in the number of traitors is that these schemes cannot be used with full collusion because in this later case, these schemes are worse than the trivial scheme of assigning each user an independent key. Agrawal et. al.
[ADVW12] go one step further by achieving an intermediate level between bounded tracing (when one assumes a maximum number of traitors) and full collusion: they allow the pirate to collect up to $t$ keys and get some bounded partial information about the others keys. We notice that the authors in [ADVW12] only considers the non-black-box tracing model and therefore a full collusion resistant scheme in the non-black-box tracing model satisfies immediately their security notion proposed. All in all, there is still an important gap between the efficiency of all these schemes and an optimal solution: the ciphertext size depends on the number of traitors and none of them can deal with full collusion. Our objective is to close this gap.

Our method is inspired from Boneh-Franklin’s open problem left in [BF99]. In fact, Boneh-Franklin mentioned: “it seems reasonable to believe that there exists an efficient public key traitor tracing scheme that is completely collusion resistant. In such a scheme, any number of private keys cannot be combined to form a new key. Similarly, the complexity of encryption and decryption is independent of the size of the coalition under the pirate’s control. An efficient construction for such a scheme will provide a useful solution to the public key traitor tracing problem”. We resolve this question in the affirmative way, by constructing a very efficient scheme with all parameters are of constant size and moreover dynamic.

We also give here the detailed comparison table between our scheme and other schemes. We notice that our scheme is the only scheme that allows minimal access single-key black box tracing. $N$ denotes the number of users in the system, $t$ denotes the maximum number of the traitors, and needs to be fixed in the setup, $n \geq 3t + 1$, $\tilde{O}(N) = O(N \log N \log \log N), \hat{O}(t^2) = O(t^2 \log t \log \log t), p$ is the order of the group $G$. Partially signifies that the pirate can collect up to $t$ keys and get some bounded partial information about the others keys.

![Table 2.1: Comparison between our public key traitor tracing scheme and some previous schemes.](image)

<table>
<thead>
<tr>
<th>Pri/Pu key</th>
<th>Ci-text</th>
<th>Enc/Dec</th>
<th>Non/Gen-b-box</th>
<th>Full/Mi-S-key</th>
<th>Revo</th>
<th>F-Col/Dyn</th>
</tr>
</thead>
<tbody>
<tr>
<td>[BF99]</td>
<td>$O(1)/O(t)$</td>
<td>$O(t)$</td>
<td>$O(t)/O(t)$</td>
<td>$\tilde{O}(N)/O(2^t)$</td>
<td>$\tilde{O}(N)/-$</td>
<td>$-$</td>
</tr>
<tr>
<td>[TSN06]</td>
<td>$O(t)/O(t)$</td>
<td>$O(t)$</td>
<td>$O(t)/O(t)$</td>
<td>$O(\frac{t^2}{\log t} \log N)/O(2^t)$</td>
<td>$O(\frac{t^2}{\log t} \log N)/-$</td>
<td>$-$</td>
</tr>
<tr>
<td>[JKL09]</td>
<td>$O(1)/O(t)$</td>
<td>$O(t)$</td>
<td>$O(t)/O(t)$</td>
<td>$\hat{O}(t^2)/O(2^t)$</td>
<td>$\hat{O}(t^2)/-$</td>
<td>$-$</td>
</tr>
<tr>
<td>[ADVW12]</td>
<td>$O(n)/O(N)$</td>
<td>$O(n)$</td>
<td>$O(n)/O(n)$</td>
<td>$O(N)/-$</td>
<td>$-$</td>
<td>Partial/-$</td>
</tr>
<tr>
<td>[NP00]</td>
<td>$O(1)/O(N)$</td>
<td>$O(t)$</td>
<td>$O(t)/O(t)$</td>
<td>$\tilde{O}(N)/O(2^t)$</td>
<td>$\tilde{O}(N)/-$</td>
<td>Yes</td>
</tr>
<tr>
<td>[BGW05]</td>
<td>$O(1)/O(N)$</td>
<td>$O(1)$</td>
<td>$O(N)/O(N)$</td>
<td>$-/-$</td>
<td>$-/-$</td>
<td>Yes</td>
</tr>
<tr>
<td>[BSW06]</td>
<td>$O(1)/O(\sqrt{N})$</td>
<td>$O(\sqrt{N})$</td>
<td>$O(\sqrt{N})/O(1)$</td>
<td>$-/-O(N^3)$</td>
<td>$-/-$</td>
<td>$-$</td>
</tr>
<tr>
<td>[BW06]</td>
<td>$O(\sqrt{N})/O(\sqrt{N})$</td>
<td>$O(\sqrt{N})$</td>
<td>$O(N)/O(\sqrt{N})$</td>
<td>$-/-O(N^3)$</td>
<td>$-/-$</td>
<td>Yes</td>
</tr>
<tr>
<td>[DPP07]</td>
<td>$O(1)/O(N)$</td>
<td>$O(t)$</td>
<td>$O(t^2)/O(t)$</td>
<td>$-/-$</td>
<td>$-/-$</td>
<td>Yes</td>
</tr>
<tr>
<td>[Del07]</td>
<td>$O(1)/O(N)$</td>
<td>$O(1)$</td>
<td>$O(N)/O(N)$</td>
<td>$-/-$</td>
<td>$-/-$</td>
<td>Yes</td>
</tr>
<tr>
<td>Ours</td>
<td>$O(1)/O(1)$</td>
<td>$O(1)$</td>
<td>$O(1)/O(1)$</td>
<td>$O(1)/-$</td>
<td>$O(\log N)/O(N)$</td>
<td>$-$</td>
</tr>
</tbody>
</table>

Table 2.1: Comparison between our public key traitor tracing scheme and some previous schemes.
2.2. AN OPTIMAL SCHEME IN THE NON-BLACK-BOX TRACING MODEL AND IN THE SINGLE-KEY BLACK BOX TRACING MODEL

2.2.2 Construction

We now present the construction of our public key traitor tracing scheme.

Let \((p, \mathbb{G}, \mathbb{G}_T, e(\cdot, \cdot))\) a bilinear map group system and \(g \in \mathbb{G}\) be a generator of \(\mathbb{G}\), our scheme is constructed as follows:

**Setup\((\lambda)\).** The algorithm chooses \(e_1, e_2, v \xleftarrow{} \mathbb{Z}_p\) then sets \(d_1 = e_1^{-1}, d_2 = e_2^{-1}\)

The master key \(\text{MSK}\) is \((e_1, e_2, v)\). The system public keys \(\text{MPK}\):

\[
(g^{d_1}, e(g, g)^{d_1}, g^{d_1d_2}, e(g, g), e(g, g)^v, e(g, g)^{d_2v})
\]

**Join\((i, \text{MSK})\).** For each user \(i\) chooses \(a_i \xleftarrow{} \mathbb{Z}_p\) such that \(a_i \neq -1, -v, d_2 - 1\). The secret key for user \(i\) is set as: \(A_i = g^{e_1(a_i + v)}, B_i = \frac{1}{a_i + 1} - e_2\). We call the secret keys in the case \(a_i = -v\) or \(a_i = d_2 - 1\) are special keys. The users in the system can be assigned to all secret keys in the secret key space except these special keys. Note that the special key, in the case \(a_i = -v\), is not useful for decryption.

**Encrypt\((M, \text{MPK})\).** Encryptor picks a random \(k\) in \(\mathbb{Z}_p\), then computes:

\[
C_1 = g^{d_1k}, C_2 = e(g, g)^{d_2k}, C_3 = g^{d_1d_2k},
\]

\[
C_4 = e(g, g)^{k-v}, C_5 = e(g, g)^{d_2k-v}, C_6 = e(g, g)^{-k} \cdot M
\]

Finally, outputs \(C = (C_1, C_2, C_3, C_4, C_5, C_6)\).

**Decrypt\((A_i, B_i, C)\).** User \(i\)'th first computes:

\[
\frac{e(A_i, C_1)}{C_4} \cdot C_2^{-1} \cdot \left(\frac{e(A_i, C_3)}{C_5}\right) B_i = e\left(g^{e_1(a_i + v)}, g^{d_1k}\right) \cdot e(g, g)^{d_2k\left(\frac{1}{a_i + 1} - e_2 - 1\right)}.
\]

\[
\cdot \left(\frac{e\left(g^{e_1(a_i + v)}, g^{d_1d_2k}\right)}{e(g, g)^{d_2k-v}}\right)^{\frac{1}{a_i + 1} - e_2} =
\]

\[
e(g, g)^{k-a_i} \cdot e(g, g)^{d_2k^{-1}} \cdot e(g, g)^{-k} \cdot e(g, g)^{-d_2k} \cdot e(g, g)^{k-a_i \left(\frac{d_2}{a_i + 1} - 1\right)} = e(g, g)^{-k}.
\]

then outputs \(M = C_6 / e(g, g)^{-k}\).

**Intuition about our construction** In the decryption, we emphasize that the crucial element is \((\frac{e(A_i, C_3)}{C_5})B_i\). We remark that, though a pirate can perform a linear combination on the elements \(A_i\) in his collected keys, there is no way for the pirate to exploit the combination of his keys to do a linear combination for the elements \((\frac{e(A_i, C_3)}{C_5})\) because \(C_5\) is changed for each encryption. Therefore the well-known pirate’s strategy of making a linear combination on the collected keys do not work for our scheme. The next section is devoted for formal analysis of security.
2.2.3 Security

In this section we will prove that our scheme above is fully dynamic IND-CPA secure.

Definition 2.2.1 (GDDHE Assumption) The \((t, \varepsilon) - \text{GDDHE}_1\) assumption says that for any \(t\)-time adversary \(A\) that is given input \((g, g^x, g^{x^y}, g^{kx}, g^{ky}, g^{kxy})\) cannot distinguish between a value \(e(g, g)^k \in \mathbb{G}_T\) or a random value \(T \in \mathbb{G}_T\), where \(x, y, k \in \mathbb{Z}_p, g \in \mathbb{G}\), with advantage greater than \(\varepsilon\):

\[
\text{Adv}^{\text{GDDHE}_1}(A) = \left| \Pr[A(\text{input}, e(g, g)^k) = 1] - \Pr[A(\text{input}, T) = 1] \right| \leq \varepsilon.
\]

It is not hard to see that \(\text{GDDHE}_1\) assumption is a special case of \((P, Q, f) - \text{GDDHE}\) assumption. Indeed, we set \(P = (p_1 = 1, p_2 = X, p_3 = Y, p_4 = XY, p_5 = KX, p_6 = KY, p_7 = KXY), Q = (g_1 = 1), f = K\). Suppose that \(f\) is not independent to \((P, Q)\), i.e., one can find \(a_8 \neq 0\) such that the following equation holds for all \(X, Y, K \in \mathbb{Z}_p\)

\[
a_8 f = \sum_{1 \leq i, j \leq 7} a_{ij} \cdot p_i \cdot p_j + b_1 \cdot q_1
\]

\[
\iff a_8 K = (KX + KY + KXY)(a_1 + a_2 X + a_3 Y + a_4 XY + a_5 KX + a_6 KY + a_7 KXY)
\]

\[
\iff a_8 = (X + Y + XY)(a_1 + a_2 X + a_3 Y + a_4 XY + a_5 KX + a_6 KY + a_7 KXY)
\]

\[
\iff (X + Y + XY)(a_1 + a_2 X + a_3 Y + a_4 XY + a_5 KX + a_6 KY + a_7 KXY) - a_8 = 0
\]

This implies that the constant term \(a_8 = 0\) which is a contradiction with the requirement that \(a_8 \neq 0\). Therefore, \(f\) is independent to \((P, Q)\).

Theorem 2.2.2 Under the GDDHE\(_1\) assumption, our scheme is fully dynamic IND-CPA secure.

Proof: We first prove that our scheme is IND-Ady-CCA\(_0\)-secure by assuming that there exists an adversary \(B\) who is successful in breaking the IND-CPA secure of our scheme, we prove that there also exists an adversary \(A\) which attacks the GDDHE\(_1\) assumption with the same advantage.

We show that \(A\) can simulate the interaction with \(B\) and then use the output of \(B\) to break the GDDHE\(_1\) assumption as follow:

In the setup, \(A\) receives the inputs from his challenger:

\[
(g, g^x, g^{x^y}, g^{kx}, g^{ky}, g^{kxy}, T)
\]

and needs to distinguish \(T\) is either \(e(g, g)^k\) or a random value in \(\mathbb{G}_T\).

In the next step, \(A\) provides the inputs for \(B\) as follow:

He chooses randomly \(z \in \mathbb{Z}_p\), implicitly sets \(d_1 = zy, d_2 = x, v = y\), then computes the public key:

\[
g^{d_1} = (g^y)^z, e(g, g)^{d_2} = e(g, g^x), g^{d_1, d_2} = (g^{x^y})^z, e(g, g), e(g, g)^v = e(g, g^y),
\]
2.2. AN OPTIMAL SCHEME IN THE NON-BLACK-BOX TRACING MODEL AND IN THE SINGLE-KEY BLACK BOX TRACING MODEL

\[ e(g, g)^{dz-x} = e(g, g^{xy}) \]

In the challenge phase, \( B \) outputs two messages \( M_0 \) and \( M_1 \). \( A \) chooses randomly a bit \( b \in \{0, 1\} \) then computes the challenge ciphertext as follow:

- \( C_1 = (g^{xy})^z = g^{dz-k} \)
- \( C_2 = e(g, g^{kx}) = e(g, g)^{dz-k} \)
- \( C_3 = (g^{kxy})^z = g^{dz+k} \)
- \( C_4 = e(g, g^{ky}) = e(g, g)^{d_z-k} \)
- \( C_5 = e(g, g^{kxy}) = e(g, g)^{d_z-k} \)
- \( C_6 = \frac{1}{T} \cdot M_b \)

then gives it to \( B \).

\( B \) outputs its guess \( b' \) for \( b \). If \( b' = b \) the algorithm \( A \) outputs 0 (indicating that \( T = e(g, g)^k \)). Otherwise, it outputs 1 (indicating that \( T \) is random in \( G_T \)).

As the simulation of \( A \) is perfect, \( A \) can thus break GDDHE assumption with the same advantage that \( B \) can break the IND-CPA-secure.

We also note that in our scheme any new user is able to join the system at any time, our scheme is thus dynamic. Overall, our scheme is fully dynamic IND-CPA-secure.

2.2.4 Traitor Tracing

In this section we will show that our scheme is secure in the non-black-box tracing model and single-key black box tracing model.

Non-Black-Box Tracing

**Definition 2.2.3 (GDDHE2 Assumption)** The \((t, \varepsilon) - \text{GDDHE}_2\) assumption says that for any \( t \)-time adversary \( A \) that is given \((b_1, \ldots, b_l, \text{input})\) in which \( b_1, \ldots, b_l \) are random in \( \mathbb{Z}_p \) and \( \neq 0 \),

\[ \text{input} = \left( g^{d_1}, g^{d_1d_2}, g^{\frac{1}{d_1^2}}, g^{\frac{1}{d_1(d_2+1)}}, \ldots, g^{\frac{1}{d_1^2(d_2+1)}} \right) \]

its probability to output a value \( g^{d_2} \in \mathbb{G} \), where \( d_1, d_2 \in \mathbb{Z}_p, g \in \mathbb{G} \), is bounded by \( \varepsilon \):

\[ \text{Succ}^{\text{GDDHE}_2}(A) = \Pr[A(b_1, \ldots, b_l, \text{input}) = g^{d_2}] \leq \varepsilon. \]

We show that this assumption holds in the generic group, the details can be found in our paper [GNPT13]. Next, we recall the definition of Modified-\( l \) - SDH assumption from [Boy07].

**Definition 2.2.4 (Modified-\( l \) - SDH Assumption)** Given \( g, g^a \in \mathbb{G} \) and \( l - 1 \) pairs \( \langle w_j, g^{1/(a+w_j)} \rangle \in \mathbb{Z}_p \times \mathbb{G} \) for a fixed parameter \( l \in \mathbb{N} \). Output another pair \( \langle w, g^{1/(a+w)} \rangle \in \mathbb{Z}_p \times \mathbb{G} \).  

**Theorem 2.2.5** Under the GDDHE2 assumption and Modified-\( l \) - SDH assumption, our scheme is secure in the non-black-box tracing model.

**Proof:** It is sufficient for us to show that the collusion of any number of traitors cannot derive a new valid secret key. Then, the proof is automatically followed since at least a
traitor’s key must be embedded in the pirate decoder and when the tracer reverse this key, the identity of the corresponding traitor is revealed.

To prove that the collusion of any number of traitors cannot derive a new valid secret key, we first prove that they cannot derive a special key $A, B$ in which $a = d_2 - 1$, we then prove that they also cannot derive any new valid secret key that differs from this special key.

**Lemma 2.2.6** Under the GDDHE\(_2\) assumption, the collusion of any number of traitors cannot derive a special key $A, B$ in which $a = d_2 - 1$.

**Proof:** Assume that there is an adversary $B$ which takes as inputs $l$ traitors’ keys, for any number $l$, the system public key, and successfully derive a special key $A, B$ in which $a = d_2 - 1$. We construct an algorithm $A$ which can simulate the interaction with $B$ and then use the output of $B$ to break the GDDHE\(_2\) assumption as follow:

In the setup, $A$ receives the inputs from his challenger:

$$b_1, \ldots, b_l, g^{d_1}, g^{d_1d_2}, g^\frac{1}{\pi_1}, g^{\frac{d_2-b_1d_2-1}{(b_1d_2+1)}}, \ldots, g^{\frac{d_2-b_ld_2-1}{(b_ld_2+1)}}$$

And needs to output the value $g^{\frac{d_2}{\pi_1}}$.

In the next step, $A$ first chooses randomly $v \in \mathbb{Z}_p$, then provides the inputs for $B$ as follow:

- $A$ provides a secret key $A_i, B_i, i = 1, \ldots, l$ for $B$ by setting $B_i = b_i = \frac{1}{a_i+1} - e_2$, therefore implicitly $a_i = \frac{d_2-b_id_2-1}{(b_id_2+1)}$, then computes

$$A_i = g^{\frac{d_2-b_id_2-1}{(b_id_2+1)}} \cdot g^{\frac{1}{\pi_1}} = g^{\frac{2}{\pi_1}} \cdot g^{\frac{v}{\pi_1}} = g^{e_1(a_i+v)}$$

where $e_1 = d_1^{-1}, e_2 = d_2^{-1}$. Note that because $b_i, d_1, d_2, v$ are randomly chosen in $\mathbb{Z}_p$, the resulted secret key is also chosen in the same distribution as in the joint algorithm.

- For the public key, $A$ computes:

$$g^{d_1}, e(g, g)^{d_2} = e(g^{d_1d_2}, g^\frac{1}{\pi_1}), g^{d_1d_2}, e(g, g) = e(g^{d_1}, g^\frac{1}{\pi_1}), e(g, g)^v, e(g, g)^{v-d_2} = e(g^{d_1d_2}, g^\frac{v}{\pi_1})$$

When $B$ outputs the special secret key $A, B$ in which $a = d_2 - 1$

$$A = g^{e_1(v+d_2-1)}, B = 0$$

then $A$ outputs

$$\frac{A \cdot g^{\frac{1}{\pi_1}}}{g^{\frac{v}{\pi_1}}} = g^{\frac{d_2}{\pi_1}}$$

As a result, the probability that the collusion of any number of traitors can derive a special key $A, B$ in which $a = d_2 - 1$ is the same as the probability that a $t$-time adversary $A$ who breaks the security of the GDDHE\(_2\) assumption.
2.2. AN OPTIMAL SCHEME IN THE NON-BLACK-BOX TRACING MODEL AND IN THE SINGLE-KEY BLACK BOX TRACING MODEL

Lemma 2.2.7 Under the Modified—l — SDH assumption, the collusion of any number of traitors cannot derive any new valid secret key that differs from the special key above.

Proof: Assume that there is an adversary $B$ which takes as inputs $l−2$ traitors’ keys, for any number $l$, the system public key, and successfully derive a new valid secret key which is different from these $l−2$ traitors’ keys and the special key above. We construct an algorithm $A$ which can simulate the interaction with $B$ and then use the output of $B$ to break the Modified—l — SDH assumption as follow:

In the setup, $A$ receives the inputs from his challenger:

$$(w_1, \ldots, w_{l−1}, g, g^{a_1}, g^{a_2}, \ldots, g^{a_{l+w−1}})$$

In the next step, $A$ provides the inputs for $B$ as follow:

He first chooses randomly $e_1, v \in \mathbb{Z}_p$, then implicitly sets $e_2 = \alpha + w_1$ thus $g^{1+w_1} = g^{\alpha_2}$. $A$ can easily compute the system public keys and gives them to $B$.

To compute $A_i$, $B_i, i = 2, \ldots, l−1$, $A$ sets $B_i = \frac{1}{a_i} − e_2 = w_i − w_1$ thus $a_i = \frac{1}{e_2+w_i−w_1}−1$ and

$$A_i = (g^{\frac{1}{a+w}})^{e_1} \cdot g^{e_1(v−1)} = g^{e_1(\frac{1}{e_2+w_i−w_1}+v−1)} = g^{e_1(a+v)}$$

Note that $\alpha = e_2 − w_1$.

When $B$ outputs a new secret key

$$A = g^{e_1(a+v)}, B = \frac{1}{a+1} − e_2$$

where $a \neq −1, d_2 − 1, a_2, \ldots, a_{l−1}$, then $A$ outputs $w = B + w_1 = \frac{1}{a+1} − e_2 + w_1$ thus

$$a = \frac{1}{e_2+w−w_1}−1, \text{ and}$$

$$\frac{g^{\alpha+w}}{(g^{a+v})^{e_1}} = g^{\frac{\alpha+v}{a+1}} = g^{\frac{1}{e_2+w−w_1}−1+1} = g^{\frac{1}{e_2+w−w_1}} = g^{\frac{1}{a+w}}$$

Note that $a \neq −1, d_2 − 1, a_2, \ldots, a_{l−1}$ thus $w \neq w_1, \ldots, w_{l−1}$.

As the simulation of $A$ is perfect, $A$ can thus break Modified—l — SDH assumption with the same advantage that $B$ can successfully derive a new valid secret key.

Single-key Black Box Tracing

Definition 2.2.8 (GDDHE$_3$ Assumption) The $(t, \varepsilon) −$ GDDHE$_3$ assumption says that for any $t$-time adversary $A$ that is given a pair $(b, \text{input})$ in which $b \neq 0$ is random in $\mathbb{Z}_p$ and

$$\text{input} = \left( g, g^{d_1}, g^{d_2}, g^{\frac{1}{d_1}} + g^{d_2}, g^{d_3−b_2d_2−1}, g^{d_1d_2}, e(g, g)^{d_2}, e(g, g)^k \right)$$

cannot distinguish between a value $e(g, g)^{kd_2} \in \mathbb{G}_T$ and a random value $T \in \mathbb{G}_T$, where $d_1, d_2, v, k \in \mathbb{Z}_p, g \in \mathbb{G}$, with an advantage greater than $\varepsilon$:

$$\text{Adv}_{\text{GDDHE}_3}(A) = \left| \Pr[A(b, \text{input}, e(g, g)^{kd_2}) = 1] - \Pr[A(b, \text{input}, T) = 1] \right| \leq \varepsilon$$

54
We notice that, unlike the Modified $l − SDH$ assumption, this is a static assumption. We show that this assumption holds in the generic group, the details can be found in our paper [GNPT13].

**Theorem 2.2.9** Under the GDDHE$_3$ assumption, our scheme is secure in the single-key black box tracing model.

**Proof:** We note that in the single-key black box tracing model, there are two separate functions which are called the key-builder and the box-builder. In the first one, the traitors will collude to derive a new valid secret key. In the second one, one receives this new secret key and build a pirate decoder based on it.

In our proof we first prove that the pirate decoder takes as inputs a secret key and the public key, cannot distinguish a probe ciphertext and a well-form ciphertext, therefore it will run the decryption algorithm normally. Finally, we present a tracing algorithm in which the tracer creates a probe ciphertext and then queries the pirate decoder on this probe ciphertext. After the pirate decoder outputs the answer, the tracer can identify the secret key that pirate decoder is using to decrypt.

Assume that there is a pirates decoder $B$, on inputs a secret key and the public key, can successfully distinguish a probe ciphertext and a well-form ciphertext. We show that $A$ can simulate the interaction with $B$ and then use the output of $B$ to break the GDDHE$_3$ assumption:

In the setup, $A$ receives the inputs from his challenger:

\[
b, g, g^{d_1}, g^{d_1 d_2}, g_{v \Pi}^w, g_{v \Pi (d_2+1)}^{d_2 d_1 - 1}, g^{k d_1}, g^{k d_1 d_2}, e(g, g)^{d_2}, e(g, g)^k, T\]

with $b, d_1, d_2, v, k$ are randomly chosen in $\mathbb{Z}_p$, and needs to distinguish $T$ is $e(g, g)^{kd_2}$ or not.

In the next step, $A$ provides the inputs for $B$ as follow:

- $A$ provides a secret key for $B$ by setting $B = b = \frac{1}{a+1} - e_2$, therefore implicitly $a = \frac{d_2 - bd_2 - 1}{(bd_2+1)}$, then computes

\[
A = g_{d_1 (bd_2+1)}^{d_2 - bd_2 - 1} \cdot g_{v \Pi}^w = g_{v \Pi}^w \cdot g_{v \Pi}^w = g^{e_1 (a+v)}
\]

where $e_1 = d_1^{-1}, e_2 = d_2^{-1}$. Note that because $b, d_1, d_2, v$ are randomly chosen in $\mathbb{Z}_p$, the resulted secret key is also chosen in the same distribution as in the joint algorithm.

- For the public key, $A$ computes:

\[
g^{d_1}, e(g, g)^{d_2}, g^{d_1 d_2}, e(g, g), e(g, g)^v = e(g^{d_1}, g_{v \Pi}^w), e(g, g)^{v d_2} = e(g^{d_1 d_2}, g_{v \Pi}^w)
\]

$A$ next chooses a random message $M$ and uses $T$ to compute the challenge ciphertext and passes it to $B$:

\[
g^{k d_1}, T, g^{k d_1 d_2}, e(g_{v \Pi}^w, g^{k d_1}), e(g_{v \Pi}^w, g^{k d_1 d_2}), e(g, g)^{-k} \cdot M
\]
In the guess phase, if $B$ outputs 0 (indicating that this is well-form ciphertext) then $A$ outputs 0 (indicating that $T$ is $e(g,g)^{kd_2}$), and otherwise if $B$ outputs 1 (indicating that this is probe ciphertext) then $A$ also outputs 1 (indicating that $T$ is a random element). We also note that $B$ can maliciously output a random message $M'$ in the case he knows the challenge ciphertext is a probe ciphertext, however $A$ still knows the right answer of $B$ because he knows the real message $M$.

As the simulation of $A$ is perfect, $A$ can thus break $\text{GDDHE}_3$ assumption with the same advantage that $B$ can successfully distinguish a probe ciphertext and a well-form ciphertext. We can thus construct a single-key black box tracing algorithm as follow:

**Full Access Single-key Black Box Tracing Algorithm:** When a user $j$ joins the system, the tracer computes and stores the pair $(j,e(g,g)_{B_j})$ in a sorted table $\text{Tab}$. The tracing algorithm then works as follow:

1. The tracer picks random $k, r \in \mathbb{Z}_p$ then creates a probe ciphertext:
   \[
   C_1 = g^{kd_1}, C_2 = e(g,g)^{kd_2+r}, C_3 = g^{kd_1d_2}, C_4 = e(g,g)^{kv},
   \]
   \[
   C_5 = e(g,g)^{kd_2}, C_6 = M'.
   \]

2. Assume the decryption key $A_i, B_i$ is embedded in the pirate decoder. Then the tracer queries the pirate decoder on this probe ciphertext. The pirate decoder will compute:
   \[
   K = \frac{e(A_i, C_1)}{C_4} \cdot C_2^{B_i - 1} \cdot \left( \frac{e(A_i, C_5)}{C_6} \right)^{B_i} = e(g,g)^{\frac{kd_2}{\alpha_i + 1}} \cdot e(g,g)^{(kd_2+r)(\frac{1}{\alpha_i + 1} - \frac{1}{2}) - 1}
   \]
   \[
   = e(g,g)^{-k} \cdot e(g,g)^{r(B_i - 1)}
   \]
   Then outputs:
   \[
   C_6/K
   \]

3. The tracer first recovers $K$ then computes $e(g,g)^{B_i}$ since it knows $k, r$. Then the tracer simply verifies if the element $e(g,g)^{B_i}$ is in the table $\text{Tab}$ and eventually outputs the traitor. It is easy to see that our tracing algorithm never accuses any innocent user and the time complexity of our tracing security is $O(\log N)$. We also notice that, in our system, $N$ is the effective number of the actual users in the system.

**Minimal Access Single-key Black Box Tracing Algorithm:** In the setup phase, the tracer picks random $k, r \in \mathbb{Z}_p$ and a message $M$, then creates:

\[
C_1 = g^{kd_1}, C_2 = e(g,g)^{kd_2+r}, C_3 = g^{kd_1d_2}, C_4 = e(g,g)^{kv}, C_5 = e(g,g)^{kd_2}
\]

and store these values in a table $\text{Tab}$.

When a user $j$ joins the system, the tracer computes

\[
C_{6,j} = e(g,g)^{-k} \cdot e(g,g)^{r(B_j - 1)} \cdot M
\]

and stores the pair $(j, C_{6,j})$ in the table $\text{Tab}$.

The tracing algorithm then works as follow:
1. For each user’s indices $j$, the tracer queries the pirate decoder on a pair 

$$(C = (C_1, C_2, C_3, C_4, C_5, C_{6,j}), M)$$

2. Assume the decryption key $A_i, B_i$ is embedded in the pirate decoder. The pirate decoder will compute:

$$K = \frac{e(A_i, C_1)}{C_4} \cdot C_2^{B_i-1} \cdot \left( \frac{e(A_i, C_3)}{C_5} \right)^{B_i} = e(g, g)^{k d_2 a_i} \cdot e(g, g)^{(k d_2 + r)(\frac{1}{n+1} - \frac{1}{n+2})}$$

Then computes:

$$M' = C_{6,j}/K$$

3. At user’s indices $j$, if the tracer receives a signal valid which indicates that $C$ is a valid encryption of $M$, then the tracer outputs user’s indices $j$ is a traitor. It is easy to see that our tracing algorithm never accuses any innocent user and the time complexity of our tracing security is $O(N)$. We also notice that, in our system, $N$ is the effective number of the actual users in the system.
2.2. AN OPTIMAL SCHEME IN THE NON-BLACK-BOX TRACING MODEL AND IN THE SINGLE-KEY BLACK BOX TRACING MODEL
Chapter 3
Resistance to Advanced Attacks on Broadcast Encryption

Contents

3.1 Tree-based Schemes and Its Variants ........................................ 61
   3.1.1 Logical Key Tree .................................................... 62
   3.1.2 Subset-cover Framework ............................................. 62
   3.1.3 Combining tree-based scheme with master key technique ........... 65
   3.1.4 From Secret-key Setting to Public-key Setting ..................... 66
3.2 Pirate Evolution Attack - PEvoA ............................................. 68
   3.2.1 Introduction .......................................................... 68
   3.2.2 Fighting against PEvoA- The Method of Jin and Lotspiech ........ 69
3.3 Pirates 2.0 Attack ........................................................... 70
   3.3.1 Introduction .......................................................... 70
   3.3.2 Fighting against Pirates 2.0 ......................................... 71
3.4 The First Identity-based Trace and Revoke Schemes - IDTR ............. 72
   3.4.1 Framework ............................................................ 74
   3.4.2 Solutions .............................................................. 76
   3.4.3 Our First Construction ............................................... 77
   3.4.4 Resistance to Pirate Evolution Attack - PEvoA .................... 80
   3.4.5 Resistance to Pirates 2.0 ............................................ 82
   3.4.6 Our Second Construction ............................................. 82
3.5 Resisting Pirates 2.0 Attack in Bounded Leakage Model ............... 88
   3.5.1 Key-Leakage Resilient Revoke Scheme ............................. 89
   3.5.2 A Construction of Key-Leakage Resilient Revoke Scheme - KIDTR 90
   3.5.3 KIDTR is Immune to Pirates 2.0 in Bounded Leakage Model ..... 96
Combinatorial schemes can be divided into three main categories: tree-based schemes, code-based schemes, and schemes combining either tree-based schemes or code-based schemes with other techniques.

For tree-based schemes, there are two main categories of schemes: the first one is stateful tree-based schemes and the second one is stateless tree-based schemes. Stateful tree-based schemes were independently introduced by Wallner et al. [WHA99] and Wong et al. [WGL98]. Many techniques were combined with stateful tree-based schemes to improve the efficiency of them, to name a few here: Sherman and McGrew [SM03] combined a tree-based structure with a one-way function, Canetti et al. [CMN99a] combined a tree-based structure with a pseudo-random generator, and Kim et al. [KPT00] combined a tree-based structure with a Diffie-Hellman key exchange scheme.

The main disadvantage of stateful tree-based schemes is the re-keying problem which requires all receivers updating their decryption keys each time a user joins or leaves the system. In order to solve this problem, the stateless tree-based schemes were introduced [KRS99, GSW00, NNL01]. The most prominent stateless tree-based schemes are the NNL schemes [NNL01], in which the authors introduced the subset-cover framework and proposed two efficient private-key trace and revoke schemes. The first one is called complete subtree (CS) scheme whose decryption key and ciphertext are of $O(\log N)$ and $O(r \log N)$ size, where $N$ is the number of users in the system and $r$ is the number of revoked users. The second one, called subset difference (SD) scheme, enjoys an interesting property: the length $(2r-1)$ of the ciphertext is independent of the number of users in the system and the size of the decryption key is $O(\log^2 N)$. These are well-known schemes since they have been used widely in practice such as a basis to design the spread content protection system for HD-DVDs and Blu-ray disks called AACS [AAC]. Since then, several important stateless tree-based schemes have been introduced such as the scheme of Halevy and Shamir [HS02] and the scheme of Asano [Asa02]. In fact, Halevy and Shamir introduced the layered subset difference (LSD) method, which is an improvement of the SD method in which the sizes of ciphertext and decryption key are of $d(2r-1)$ and of $O(d \log^4 N)$ respectively. Asano [Asa02] made use of the master key technique to achieve the constant size secret key, and used Power Set Method in conjunction with an $a$-ary logical key tree structure to reduce the ciphertext size.

Dodis and Fazio [DF02] introduced a way to transform the NNL schemes from a secret-key setting to a public-key setting by using IBE and HIBE. This is also the first paper to show how to combine tree-based structure with the identity-based cryptography. Their schemes are denoted by DF-CS (transforming CS scheme to public-key setting by using an IBE scheme such as Waters IBE in [Wat05]) and DF-SD (transforming SD scheme to public-key setting by using an HIBE scheme such as the BBG scheme [BBG05]). The resulting schemes are competitive to the original CS and SD schemes.

As mentioned above, BE schemes can be divided into two main categories: algebraic schemes and combinatorial schemes, in which many combinatorial schemes have been used widely in the practice. However, two advanced attacks on BE schemes have been proposed recently. These attacks point out the weakness of combinatorial schemes, in particular the weakness of the subset-cover framework based schemes and code-based schemes. These advanced attacks are called pirate evolution attack [KP07] and Pirates 2.0 [BP09].
CHAPTER 3. RESISTANCE TO ADVANCED ATTACKS ON BROADCAST ENCRYPTION

Pirate evolution attack - PEvoA is an attack concept against a trace and revoke scheme that exploits the properties of the combined functionality of tracing and revocation in a tree-based scheme. Using a set of users’ keys, the pirate produces an initial pirate decoder. When this pirate decoder is seized, the pirate evolves the first pirate decoder by issuing a second version that succeeds in decrypting ciphertexts. The same step can be repeated again and again and the pirate continues to evolve a new version of the previous decoder.

Pirates 2.0 attack results from traitors collaborating in a public way. In other words, traitors do not secretly collude but display part of their secret keys in a public place; pirate decoders are then built from this public information. The distinguishing property of Pirates 2.0 attack is that traitors only contribute partial information about their secret key material which suffices to produce (possibly imperfect) pirate decoders while allowing them to remain anonymous. This type of attack has been shown to be a real threat for tree-based and code-based schemes.

In this chapter, we present two contributions to fight against PEvoA and Pirates 2.0 attack. In our first contribution, we proposed a method to resist well PEvoA and Pirates 2.0 attack in the subset-cover framework. Concerning the PEvoA attack, our idea is that we build a scheme in which the secret key contains just one sub-key, and from this sub-key one cannot derive any other sub-keys which can be used to decrypt the ciphertext. This leads to the fact that the attacker in PEvoA attack cannot evolve the pirate decoder to decrypt the ciphertext. Our schemes also resist a restricted model of Pirate 2.0 attack, in which each time a traitor contributes his secret information to the public domain he has to contribute entirely one of his sub-keys. The secret key in our schemes only contains one sub-key and from this sub-key one cannot derive any other sub-key, if a traitor contributes entirely his sub-key to the public domain, he will lose his anonymity. We notice that our considering model covers all existed attacks on the tree-based schemes so far, moreover all other known methods to fight against Pirates 2.0 [DdP11, DdP13, ZZ11] also only consider this model. For concrete construction, our scheme (in public-key setting) is constructed by integrating WIBE scheme [ACD+06, ABC+11] into the CS scheme, this results in the first identity-based trace and revoke scheme whose efficiency can be comparable to the original scheme CS.

Our second contribution is that we first showed that all existed methods fighting against Pirates 2.0 attack only consider a particular form of Pirates 2.0 attack in a special form of bounded leakage model. We thus investigated the Pirates 2.0 attack in the bounded leakage model to find out a connection between Pirates 2.0 attack and leakage resilient cryptography [PT13]. To this aim, we first formally defined a Pirates 2.0 attack in the bounded leakage model and then proposed a key-leakage resilient scheme (denoted KIDTR) which resists Pirates 2.0 attack in the bounded leakage model.

3.1 Tree-based Schemes and Its Variants

There are two main categories of tree-based schemes: the stateful tree-based schemes and the stateless tree-based schemes. In these categories, tree-based schemes are also combined with many techniques to improve the efficiency and to transform from the secret-key setting to public-key setting. In this section, we recall the logical key tree structure that
resulting in *stateful* tree-based schemes, subset-cover framework that resulting in *stateless* tree-based schemes, and several important combinations between tree-based schemes and other techniques such as pseudo-random generator, master key technique, identity-based cryptography, and polynomial interpolation.

### 3.1.1 Logical Key Tree

Wallner *et al.* [WHA99] and Wong *et al.* [WGL98] independently employ a *logical key hierarchy* to construct multicast encryption schemes. *Logical key hierarchy* is a complete binary tree with $N$ leaves, and in their schemes these $N$ leaves correspond to $N$ users in the system. In order to assign a secret key to each user, they first assign each node on the tree a random key, then the secret key of each user (who is a leaf) contains $\log N + 1$ keys on the path from his leaf to the root (including key at leaf). In order to revoke a user $u$, all keys associated with nodes on the path from $u$’s leaf to the root are changed. To this aim, denoted $\text{parent}(x)$ is parent of node $x$ and $\text{sib}(x)$ is sibling of node $x$, the center will start from the $u$’s leaf by encrypting a new key associated with node $\text{parent}(u)$ under the key associated with node $\text{sib}(u)$, thus only sibling of node $u$ knows the new key associated with node $\text{parent}(u)$. For the key associated with node above node $\text{parent}(u)$, that is node $\text{parent}(\text{parent}(u))$, the center encrypts twice, one under the new key associated with $\text{parent}(u)$ and one under the key associated with $\text{sib}(\text{parent}(u))$, thus both $\text{sib}(u)$ and $\text{sib}(\text{parent}(u))$ know the new key associated with node $\text{parent}(\text{parent}(u))$, etc. By this way, the total messages the center needs to revoke one user are $2 \log N + 1$ messages.

We also note that a new user can join the system if the current number of users is less than $N$, the center simply assigns the new user to an unoccupied leaf node on the tree. In the case the number of users exceeds $N$, a new root node is created. This increases the depth of the tree by one, and allows the possible number of users in the system to be doubled. These kinds of scheme are known as the *stateful BE* schemes.

#### Combination with pseudo-random generator

An important improvement on *logical key hierarchy* was proposed by Canetti *et al.* [CMN99b]. They improved the efficiency of the transmission complexity when updating users’ secret key by using a pseudo-random generator, the result is that revoking one user only costs $\log N + 1$ messages. The main idea is that a random seed $r$ is encrypted with the key of $\text{sib}(u)$. Then a pseudo-random generator that doubles the size of input with right output half $R$ and left output half $L$ is used to derive $R(r)$, $R(R(r))$, ... We assume that $L(r)$ is the new key associated with $\text{parent}(u)$, then it can only be derived by $\text{sib}(u)$. $R(r)$ is encrypted with the key of $\text{sib}(\text{parent}(u))$, and $L(R(r))$ is the new key of $\text{parent}(\text{parent}(u))$, $R(R(r))$ is encrypted with the key of $\text{sib}(\text{parent}(\text{parent}(u)))$, and the key $L(R(R(r)))$ is associated with $\text{parent}(\text{parent}(\text{parent}(u)))$ and so on.

### 3.1.2 Subset-cover Framework

In the *stateful BE* schemes mentioned in the previous section, in order to revoke one user, the center needs to broadcast $2 \log N + 1$ messages. Therefore to revoke $r$ users, the center
Aiming at solving this problem, Naor et al. [NNL01] propose two stateless BE schemes (CS and SD) which are based on a new framework, called the subset-cover framework. The efficiency of the SD scheme is later improved in [HS02].

**Subset-cover framework [NNL01]** Let $\mathcal{N}$ be the set of all users in the system, $|\mathcal{N}| = N$, and $\mathcal{R} \subseteq \mathcal{N}$ be a group of $|\mathcal{R}| = r$ revoked users. The goal of a revocation algorithm is to allow a user to transmit a message $M$ to all non-revoked users and all the revoked users cannot decrypt the message even they collude. Since the receivers are stateless, the output of the decryption should be based on the current message and the secret key only.

Figure 3.1: Complete subtree scheme - CS

In this framework, the set of users $\mathcal{N}$ can be partitioned into $t$ subsets $(S_1, S_2, \ldots, S_t) \subseteq \mathcal{N}$. Each subset $S_i$, $1 \leq i \leq t$ is associated with a key. Each user $u \in S_i$ should be able to decrypt the ciphertext which is encrypted by the key of $S_i$, and any users $v \notin S_i$ cannot decrypt, even if all of them collude.

In order to encrypt a message $M$ to all non-revoked users $\mathcal{N}/\mathcal{R}$, one first partitions $\mathcal{N}/\mathcal{R}$ into $w$ disjoint subsets $(S_{i_1}, S_{i_2}, \ldots, S_{i_w})$ such that $S_{i_1} \cup S_{i_2} \cup \cdots \cup S_{i_w} = \mathcal{N}/\mathcal{R}$. Then a session key $K$ is randomly chosen and is encrypted $w$ times under the keys corresponding to $w$ subsets $(S_{i_1}, S_{i_2}, \ldots, S_{i_w})$. Finally, message $M$ is encrypted under the session key $K$. It is easy to see that the ciphertext size will be linear in $w$.

**Complete subtree scheme CS [NNL01]** This scheme employs a complete binary tree with $N$ leaves corresponding to $N$ users. Similar to logical key hierarchy, each node $i$ on the tree is assigned a random key $L_i$, a secret key of a user then contains $\log N + 1$ keys on the path from his leaf to the root (including key at leaf). For example, in the figure 3.1, user 4 corresponding to node 11 possesses the keys associated with node 11, 5, 2, 1. In order to encrypt a message to users 2 and 3 corresponding to node 10, 11, the center uses the key associated with node 5.

Let $\mathcal{R}$ be the set of revoked users, in order to partition $\mathcal{N}/\mathcal{R}$ into $w$ disjoint subsets $(S_{i_1}, S_{i_2}, \ldots, S_{i_w})$ such that $S_{i_1} \cup S_{i_2} \cup \cdots \cup S_{i_w} = \mathcal{N}/\mathcal{R}$, one colors the leaves of the
3.1. TREE-BASED SCHEMES AND ITS VARIANTS

revoked users and all their ancestors black. These black nodes will form the Steiner tree $ST(R)$ of the set $R$ of revoked users. The subsets $(S_{i_1}, S_{i_2}, \ldots, S_{i_w})$ now are non-black subtrees. They prove that for $r$ revoked users, $w$ is at most $r \log \frac{N}{r}$. From the subset-cover framework, the ciphertext of CS scheme is of $r \log \frac{N}{r}$ size, and the secret key is of $\log N + 1$ size.

For tracing traitors, they achieve a relaxed black-box traceability: given a pirate decoder which can decrypt successfully a valid ciphertext with probability at least $p$, the tracing algorithm either outputs the identity of one of the traitors, or a subset partition that can be used to broadcast to all legitimate users and make the pirate decoder useless (i.e., successfully decrypt with negligible probability).

![Figure 3.2: The subset difference scheme - SD](image)

**Subset difference scheme SD** [NNL01] In the CS scheme, the ciphertext is still dependent on the maximum number of users in the system ($N$). To overcome this drawback, they designed a second scheme known as subset difference scheme in which the ciphertext is independent of the maximum number of users in the system.

In this scheme, $N$ users are the leaves on a complete binary tree as in the CS scheme. They define a subset $S_{i,j}$ which is a subtree rooted at $v_i$ minus the subtree rooted at $v_j$, that means the subtree $S_{i,j}$ does not contain any user who is descendant of $v_j$. For examples, the subtree $S_{2,10}$ in the figure 3.2 contains the users corresponding to nodes 8, 9, 11 not 10. Each node $i$ on the complete binary tree is assigned a label $L_i$. The label of node $v_j$ on the subtree $S_{i,j}$ is $L_{i,j}$ and it is derived from label $L_i$ as follows: using a pseudo-random sequence generator $G$ that triples the size of input: $G_L(L_i)$ is the label of left child of the subtree rooted at $v_i$, $G_R$ is right child, and $G_M(L_{i,j})$ is the key associated with the subset $S_{i,j}$.

Concerning a user $u$ in the subtree rooted at $v_1$, the secret key of $u$ contains the label $L_{i,k}$ where $v_k$ is the siblings of ancestors of $u$ in the subtree. For examples, the secret key of user corresponding to node 13 in the subtree rooted at node 3 contains the label $L_{3,12}$ and $L_{3,7}$. By this way, $u$ can compute any label $L_{i,k}$ in which $u$ is not a descendant of $v_k$ with the complexity at most $O(\log N)$, and from this label he can compute the key
Layered subset difference scheme - LSD [HS02] This scheme is an improvement of the SD scheme in terms of efficiency, in which they reduce the size of secret key and pay the cost on the ciphertext size. In fact, they propose a trade-off between secret key and ciphertext from $O(\log^2 N), 2r - 1$ in SD scheme to $O(d \log^{1+\frac{1}{d}} N), d(2r - 1)$ in their scheme.

The main idea of their method is that any subset $S_{i,j}$ in the SD scheme can be divided into two sub-subsets $S_{i,k}$ and $S_{k,j}$, in which $k$ is on the path from $i$ to $j$. They define layers and the special levels in which the root and every level of depth $k \cdot \sqrt{\log(N)}$ are called special levels, and the levels between them are called layers. Based on the layers and the special levels, they define the local subset and special subset. $S_{i,j}$ is a local subset if both $i, j$ lie on the same layer, and is a special subset if $i$ lies on a special level. They prove that all subsets $S_{i,j}$ in the SD scheme can be presented as an union of a local subset and a special subset. By this way, the labels memorized by each user will be reduced and the number of subsets in the partition will be increased, while any properties of their scheme remain the same as in the CS scheme.

3.1.3 Combining tree-based scheme with master key technique

In order to improve the efficiency of the tree-based scheme, particularly optimize the storage at receivers, Asano [Asa02] introduced a way to combine tree-based scheme with the master key technique [CT90], this results in a new revoke scheme whose secret key is of constant size. Their idea is that they use master key technique to achieve constant size secret key and Power Set Method in conjunction with an $a$-ary logical key tree structure (each internal node on the tree has $a$ children) to reduce the ciphertext size. The Power Set Method with $n$ users is defined as follows: let users be numbered from 1 to $n$, we define a subset $S_{b_1,...,b_n}$ where $b_i$ is a bit 1 if user $i$’th belongs to subset and 0 if not, therefore there are $2^n - 2$ such subsets (ignore the case $\sum b_i = 0$ or $= n$).

Let $N$ be the possible number of users, and be a power of $a$. They choose a logical $a$-ary tree, then they use the Power Set Method with $a$ elements for each internal node (including the root), thus each internal node has $2^a - 2$ subsets, the internal nodes are named from 1 to $\frac{N-1}{a-1}$ from left to right. Similar to a tree-based scheme, users in this scheme are leaves. For examples, in the case $a = 3$, the subsets of node $k$ are $S_{k,100}, S_{k,010}, S_{k,001}, S_{k,110}, S_{k,101}, S_{k,011}$.

Trusted center chooses two large primes $q_1, q_2$ and publishes $M = q_1 q_2$, it also chooses $(2^a - 2) \frac{N-1}{a-1} + 1$ primes $p_{k,b_1b_2,...,b_n}$ where $k = 1, \ldots, \frac{N-1}{a-1}$. Denote $\{b_1, \ldots, b_a\}$ by $B$, assign $p_{k,B}$ to a subset $S_{k,B}$ and publish this assignment. Let $T$ be the product of all $p_{k,B}$.

Trusted center chooses a random $K$, then the key associated to subset $S_{k,B}$ is computed
3.1. TREE-BASED SCHEMES AND ITS VARIANTS

$SK_{k,B} = K^{T/p_k,B \mod M}$. The child $i$'th of this node will get this key if it belongs to this subset (means $b_i = 1$), thus each child will get $2^a - 1$ keys. Each user $u_j$ at leaf now is assigned a master key $MK_j$, and $MK_j$ is computed as follows: let $w_j$ be the product of all keys which belong to nodes on the path from $j$ to the root (there are $(2^a - 1) \log_a N + 1$ such keys), $MK_j = K^{T/w_j}$. When encrypting, the center first partitions the $a$-ary tree to revoke the revoked users, then encrypts the session key by using the keys of root of each subtree in the partition. They prove that there are maximum $r(\frac{\log N}{\log a} + 1)$ subtrees in the partition, thus the ciphertext size of ther scheme is linear in $O(r(\frac{\log N}{\log a} + 1))$. For decryption, a decryptor $j$ finds an appropriate subset, then derives the subset key from its master key as follow:

$$MK_j^{w_j/p_k,B} = (K^{T/w_j})^{w_j/p_k,B} = K^{T/p_k,B} = SK_{k,B}$$

Note that, since user $j$ belongs to subset $S_{k,B}$ thus $p_k,B \mid w_j$.

Their scheme achieves constant size secret key but the shortcoming is that the complexity of generation of primes and decryption time are high. In order to deal with this problem, they also propose the second scheme which is a trade-off between the secret key size and the complexity. In fact, the secret key will be linear in $O(\frac{\log N}{\log a})$ and the complexity reduces to $2^a - 1$ multiplication operations.

3.1.4 From Secret-key Setting to Public-key Setting

All tree-based schemes mentioned above are private-key BE schemes. Dodis and Fazio [DF02] showed a way to transform the CS, SD schemes from the secret-key setting to the public-key setting by using IBE and HIBE. They proposed two schemes, denoted DF-CS which is a transformation from the CS scheme to public-key setting, where the best instance of this scheme is obtained by combining with Waters IBE in [Wat05], and denoted DF-SD which is a transformation from the SD scheme to public-key setting, where the best instance of this scheme is obtained by combining with the BBG scheme [BBG05].

Identity-based Cryptography  To warm up, we recall some definitions in identity-based cryptography. Identity-based encryption (IBE) allows a sender to encrypt a message using the receiver’s identity as a public key. By using the identities as public keys, it avoids the need to distribute public key certificates. There are many practical applications which are suitable to this primitive. For examples, email application where the receivers are often off-line and unable to present a public key certificate while the sender encrypts a message.

The definition for an IBE scheme as follow: the Setup takes as input a security parameter $\lambda$ and returns a public key and a master key. Extract takes as inputs the public key, master key, and an identity $ID \in \{0,1\}^*$, returns the secret key corresponding to identity $ID$. Encrypt takes as inputs the public key, message $M$, an identity $ID^*$ of user who is intended to receive this message, returns the ciphertext $C$. Decrypt takes as ciphertext $C$, secret key corresponding to identity $ID^*$, and the public key, returns the message $M$.

For construction, the breakthrough is made by Boneh and Franklin [BF01] in which they first proposed an efficient IBE scheme based on bilinear maps and the security based
CHAPTER 3. RESISTANCE TO ADVANCED ATTACKS ON BROADCAST ENCRYPTION

on random oracle. Since then, many constructions are introduced to improve the efficiency and the security, we can name a few such as [CHK03, BB04a, BB04b, Wat05, BBG05].

**HIBE** is an extension of IBE in which users are organized in a tree hierarchy. The identity of user specifies the position of user in the tree, and denoted as a tuple $\text{HID} = (ID_1, \ldots, ID_l)$ where each $ID_i$ is an identity as in IBE scheme, and is called a level. In a HIBE scheme, from secret key of parent, one can extract the secret keys of its children and its descendants. When encrypting a message to a user by using his identity, this user and all his ancestors can decrypt the corresponding ciphertext. Note that, a HIBE scheme can be easily constructed from a IBE scheme, and BBG scheme [BBG05] is the first construction which achieves the constant size ciphertext.

Identity-based encryption with wildcards [ACD+06] is a generalization of HIBE scheme where at the time of encryption, the sender can decide to make the ciphertext decryptable by a whole range of users whose identities match a certain pattern. Such a pattern is described by a vector $P = (P_1, \ldots, P_l) \in (\{0,1\}^* \cup \{\ast\})^l$, where $\ast$ is a special wildcard symbol. An identity $\text{HID} = (ID_1, \ldots, ID_l)$ matches a pattern $P$, denoted $\text{HID} \in_\ast P$, if and only if $l' \leq l$ and $\forall i = 1 \ldots l'': ID_i = P_i$ or $P_i = \ast$. By this way, any ancestor of a matching identity is also a matching identity.

For definition, a WIBE scheme consists of four algorithms (Setup, Extract, Encrypt, Decrypt). The Setup and Extract algorithms are the same as in HIBE scheme (since the users in WIBE scheme are organized in a tree hierarchy as in HIBE scheme). When encrypting, one wants to encrypt a message $m \in \{0,1\}^*$ intended for all identities matching pattern $P$, he computes $C \leftarrow \text{Encrypt}(\text{mpk}, P, m)$. Any of the intended receiver with identity $\text{HID} \in_\ast P$ can decrypt the ciphertext using its own decryption key. Some efficient constructions of WIBE schemes can be based on Waters HIBE [Wat05] and BBG [BBG05].

**Dodis and Fazio** [DF02] For the first scheme DF-CS, their main idea is that they label each node on the CS tree with a bit 0 or 1 depending on the node is left child or right child. By this way, each node $S_j$ has a unique identity $ID_{S_j}$ that is a bit string from the root to the node $S_j$. By using an IBE scheme, each node $S_j$ can extract a key $L_{S_j}$ corresponding to its identity $ID_{S_j}$. Now, each node on the tree has two parts: an identity which is published, and a key associated with this node as in the CS scheme. They also distribute the secret keys for users in the same way as in the CS scheme, this means that the number of keys that a user must store is still $O(\log N)$. When encrypting, an encryptor first partitions $\mathcal{N}/\mathcal{R}$ into $r \log \frac{N}{r}$ subsets $S_j$ as in the CS scheme, then using IBE encryption algorithm to encrypt the session key $K$ under identity of each subset $S_j$. By this way, the encryptor does not need to know the key associated with node $S_j$ as in the CS scheme, that means that the scheme is now in public-key setting. Each user in the subtree rooted at node $S_j$ can decrypt the session key $K$ because he knows the key $L_{S_j}$. The length of ciphertext is still linear in $O(r \log \frac{N}{r})$ as in the CS scheme. The security of the scheme depends on the security of underlying IBE scheme and the CS scheme. The best instance of this scheme is obtained when combining with Waters IBE in [Wat05], and is depicted in the table 3.1.

For the second scheme DF-SD, in the SD scheme they realize that from the label of parent node in a subtree, one can extract the label of child node in this subtree, this is
similar to scenario of a HIBE scheme, in which from the secret key of identity \( ID_i \) one can extract the secret keys of all identity \( ID_j \) where \( ID_j \) is prefix of \( ID_i \). Therefore, they combine a HIBE scheme with the SD scheme, they also label each node on the tree with a bit as in DF-CS scheme above. Considering a subtree \( S_{i,j} \), the hierarchical identifier for this subtree is defined by three levels \( ID_{S_{i,j}}, ID_{v_i,v_j}, 2 \), where \( ID_{S_{i,j}} = ID_{v_i} \) (bit string from root to node \( v_i \)), \( ID_{v_i,v_j} \) is bit string on the path from \( v_i \) to \( v_j \). By using a HIBE scheme, the secret key \( L_{i,j} \) corresponding to hierarchical identifier \( ID_{S_{i,j}}, ID_{v_i,v_j}, 2 \) is extracted from the secret key \( P_{i,j} \) of its parent \( ID_{S_{i,j}}, ID_{v_i,v_j}, 2 \). Moreover, \( P_{i,j} \) is also extracted from the secret key \( P_{i,l} \) of \( ID_{S_{i,j}}, ID_{v_{i,l}} \) where \( v_i \) is parent of \( v_j \). In order to distribute secret keys to users, they use the same way as in the SD scheme, considering a user \( u \) in the subtree \( S_{i} \), let \( v_1, \ldots, v_{i-1} \) be nodes on the path from \( u \) to \( v_i \) (parent and ancestors of \( u \)), \( v'_1, \ldots, v'_{i-1} \) are siblings of them. The secret key of user \( u \) will include the secret keys of hierarchical identifiers \( (ID_{S_{i,j}}, ID_{v_i,v'_i}), \ldots, (ID_{S_{i,j}}, ID_{v_{i-1},v'_{i-1}}) \). Thus the secret key size is still linear in \( O(\log^2 N) \). When encrypting, encryptor only needs to know the hierarchical identifier \( ID_{S_{i,j}}, ID_{v_{i,l}} \), \( 2 \) which are published thus the scheme is in public-key setting. The security of the scheme depends on the security of underlying HIBE scheme and the SD scheme. The best instance of this scheme is obtained when combining with BBG scheme [BBG05], and is depicted in the table 3.1.

### 3.2 Pirate Evolution Attack - PEvoA

#### 3.2.1 Introduction

Pirate evolution attack is a new kind of attack which decreases the efficiency of the system. In the initial phase, the adversary corrupts \( t \) traitor keys and uses this set of keys to create a master pirate box decoder \( B \). \( B \) spawns successively a sequence of pirate boxes decoders \( B_1, B_2, \ldots \). After a current version of pirate box \( B_i \) (\( i = 1, \ldots \)) is captured and rendered useless, the master box \( B \) spawns a new version of pirate box \( B_{i+1} \) that can decrypt the ciphertext encrypted at that time. The center continues to run the tracing algorithm to render \( B_{i+1} \) useless. The iteration repeats until \( B \) cannot spawn a new version of pirate box decoder. If \( B \) can spawn a greater number of pirate boxes than \( t \), the system takes a long time to eradicate pirates in this attack. In this case we say that the system is susceptible to the pirate evolution attack. We also can name a few schemes which are susceptible to this attack [NNL01, HS02, GST04], in particular, [Asa02, AK05] are also susceptible to this attack although the secret key of user just contains one sub-key, the reason is that each user can always possess (or derive) keys corresponding to intermediate nodes in the tree. Therefore, the traitors can easily produce many pirate boxes.

**Formalization of PEvoA.** We now give the formalization of a pirate evolution attack that covers all known attacks of this type.

**Definition 3.2.1 (PEvoA in Subset-Cover Systems)** In a subset-cover system with \( N \) users, each user \( u_i \), \( i = 1, \ldots, N \), possesses a set of decryption keys \( K_{u_i} = (K_{u_{i,1}}, \ldots, K_{u_{i,h}}) \). A master pirate box \( B \) built from \( t \) traitors \( (u_1, \ldots, u_t) \) should possess a set of decryption keys \( (K_{u_1}, \ldots, K_{u_t}) \).
A pirate evolution attack is an attack in which $B$ spawns successively a sequence of pirate boxes decoders $B_1, B_2, \ldots$ corresponding to each iterations $1, 2, \ldots$. The evolution from the iteration $i$ to the iteration $i + 1$ is defined as follows:

- At each iteration $i$, after the tracing algorithm creates a new partition $S_{i+1}$ which renders the pirate box $B_i$ useless, or revokes some set of users, $B$ spawns a new version $B_{i+1}$ of pirate box, where $B_{i+1}$ must contain at least a decryption key $d_{i+1}$ such that $B_{i+1}$ can decrypt the ciphertext encrypted under the new partition $S_{i+1}$ with probability higher than a given threshold $q$.
- The iteration repeats until $B$ cannot spawn a new version of the pirate box decoder.

The progress of $PEvoA$ can be formalized by an algorithm $evo[PEvoA] = PEvoA (K_{u_1}, \ldots, K_{u_t})$ where $u_1, \ldots, u_t$ are traitors:

1. $(K_{u_1}, \ldots, K_{u_N}) \leftarrow G(1^\lambda, \rho, N)$ where $\rho \leftarrow \text{coins}
2. K_t \subseteq (K_{u_1}, \ldots, K_{u_N})$ and $|K_t| = t$
3. $l = 0$
   repeat $l = l+1$
   \[ B_l(d_{l,1}, \ldots, d_{l,y_l}) \leftarrow B(K_t) \text{ where } d_{l,1}, \ldots, d_{l,y_l} \text{ are decryption keys at iteration } l \]
   until $\text{Prob}[B_l(d_{l,j}, E^{B_1, \ldots, B_{l-1}}(m)) = m] < q$ where $j = 1, \ldots, y_l$
   $E^{B_1, \ldots, B_{l-1}}(m)$ is the ciphertext at iteration $l$, $q$ is given threshold, $m \in \mathbb{M}$
4. output $l$, the number of pirate box decoders that $B$ can spawn.

Definition 3.2.2 (Resistance to $PEvoA$) A system is susceptible to $PEvoA$ if the maximum number of pirate box decoders that $B$ can spawn is greater than the number of traitors $t$. The system is immune from $PEvoA$ if the maximum number of pirate box decoders is less than or equal to $t$.

3.2.2 Fighting against $PEvoA$- The Method of Jin and Lotspiech

Recently, [JL09] proposed a method to defend against pirate evolution attack in the subset difference scheme [NNL01], but their method decreases the efficiency of the original scheme.

The main idea of $PEvoA$ is that each user $u$ in the subset difference method possesses a set of decryption keys (long live keys) associated to $S_{i,j}$ ($i, j \in [1, \ldots, \log N]$), where $u$ is a descendant of $v_i$ but not $v_j$. Given to a master pirate box decoder $B$ a set of decryption keys of a traitor $u \in S_{i,j}$ ($u$ is not a descendant of $v_j$), $B$ can successively spawn a series of pirate boxes decoder $B_1, B_2, \ldots$ by using the long live keys of a series sets $S_{c,j}, \ldots$, where $v_c$ is a node on the path from $v_i$ to $v_j$. 

69
3.3. PIRATES 2.0 ATTACK

In [JL09], they proposed a method to fight against this attack, their method is quite simple, they only level down the encryption key $K_{s_{i,j}}$ to the desired level by, first, splitting $K_{s_{i,j}} = K_{s_{i,j_1}}, K_{s_{j_1,j_2}}, \ldots, K_{s_{j_h,j}}$ where $j_1, \ldots, j_h$ are nodes on the path from $i$ to $j$, then continuing to level down each subset $K_{s_{i,j_1}}, K_{s_{j_1,j_2}}, \ldots, K_{s_{j_h,j}}$ to the desired level as long as the combination of these subsets covers all users in the set $S_{i,j}$. They proved that if the tracing algorithm wants to detect and disable a traitor in only $b$ generations of pirate decoder, each subset needs to be expanded into multi-subsets; the overall expanded subsets are linear in $N^{1/b}$. Therefore, the ciphertext size will magnify to a factor of $O(N^{1/b})$.

By this way they can restrain master pirate box $B$ from spawning $B_i$. In exchange, however, the size of ciphertext will be magnified. If their scheme resists completely to pirate evolution attack, the size of ciphertext in their scheme must be linear in the number of users in the system. They stress the case that when generations of clone decoder is 2, the efficiency of their scheme is acceptable in practice.

3.3 Pirates 2.0 Attack

3.3.1 Introduction

Pirates 2.0 is a new type of attack against traitor tracing schemes. This type of attack results from traitors collaborating together in a public way and display part of their secret keys in a public place, allowing pirate decoders to be built from this public information. Traitors contribute part of their secret keys in an appropriate way in order to remain anonymous. Moreover, the traitors can contribute information at their discretion and they can publicly collude in groups of large size.

The basic idea behind Pirates 2.0 attack is that traitors are free to contribute some piece of secret data as long as several users of the system could have contributed exactly the same information following the same (public) strategy; this way, they are able to remain somewhat anonymous. The leakage information is formalized via extraction function which is any efficiently computable function $f$ on the space of the secret keys and a traitor $u$ is said to be masked by a user $u'$ for an extraction function $f$ if $f(sk_u) = f(sk_{u'})$. The anonymity level is meant to measure exactly how anonymous they remain. We now recall some definitions for Pirates 2.0.

**Traitor’s Strategy.** A traitor’s strategy is a publicly available probabilistic algorithm Contribute that traitors exeute to provide information to pirates, and all traitors use the same strategy.

**Definition 3.3.1 (Extraction Function)** An extraction function is an efficiently computable function $f$ that outputs information about the secret key.

**Definition 3.3.2 (Masked Traitor)** A traitor $t$ is said to be masked by a user $u$ for an extraction function $f$ if $f(sk_u) = f(sk_t)$.

70
Definition 3.3.3 (Anonymity Level) The level of anonymity of a traitor $u$ after a contribution $\bigcup_{1 \leq i \leq t} f_i(s_{k_u})$ is defined as the number $\alpha$ of users masking $u'$ for each of the $t$ extraction functions $f_i$ simultaneously:

$$\alpha = \#\{u' | \forall i, f_i(s_{k_u}) = f_i(s_{k_{u'}})\}.$$

Definition 3.3.4 (Pirates 2.0) A traitor tracing scheme is said to be vulnerable against a Pirates 2.0 attack if:

- there is a construction of a pirate decoder from information published by traitors in such a way that the traitor rest assured to have an anonymity level of $\alpha > 1$.
- the pirate is able to specify at least one target set $S$ so that the produced pirate decoder can decrypt ciphertexts for this target set $S$ with a non-negligible probability.

We realize that any scheme, which has the secret key of user containing many sub-keys and some sub-keys can be shared among many secret keys of users, will be vulnerable to a Pirates 2.0 attack. The reason behind it is that a traitor can publish the shared sub-key to the public domain while still remain anonymous because others users also possess this shared sub-key. For examples, in the complete subtree scheme [NNL01], each user is a leaf on the tree and possesses $\log N$ sub-keys, each sub-key corresponds to a node on the path from the user’s leaf to the root. By this way, users in the system can share many sub-keys because they share the same many ancestors. So that, if a user publishes a sub-key corresponding to one of these same ancestors he will remain anonymous because many other users possess the same this sub-key. Moreover, by using this sub-key, a pirate decoder is able to decrypt the ciphertext encrypted to this corresponding node. This strategy of attack can be used in the code-based scheme because in the code-based scheme the secret key of user contains many sub-keys, and many sub-keys can also be shared among many secret keys of users.

3.3.2 Fighting against Pirates 2.0

The method of [DdP11, DdP13]: The main idea in [DdP11, DdP13] is that they combine the Naor-Pinkas scheme with the CS scheme to form a new scheme. The secret key of user in this new scheme contains many sub-keys as in the CS scheme, however each sub-key is not shared among many users as in the CS scheme, therefore they can restrain the user from publishing his sub-keys to public domain, because if he does that he will immediately reveal his sensitive information. In fact, for each subset $S_i$ in the CS scheme (or $S_{i,j}$ in the SD scheme) they choose uniform at random a secret $t$-degree polynomial $P_i$ where $t$ is the maximum number of users who collude. Each user $u$ who belongs to subset $S_i$ in the CS scheme (or $S_{i,j}$ in the SD scheme) receives the pair $(i, P_i(I_u))$.

Encryption and decryption are the same as in the Naor-Pinkas scheme. For each subset $S_i$ in the CS scheme (or $S_{i,j}$ in the SD scheme) a broadcaster uses $t + 1$ elements $K_{i,0}, \ldots, K_{i,t}$ (or $K_{i,j,0}, \ldots, K_{i,j,t}$ in the SD scheme) which are computed from $g^{P_i(0)}, \ldots, g^{P_i(t)}$ and some other random elements (depending on the CS scheme or the SD
scheme). One can recover $K_{i,0}$ if he knows $t+1$ different values $K_{i,j}$. The broadcaster then encrypts the session key $K$ under $K_{i,0}$ and broadcasts the ciphertext including $t$ different values $K_{i,1}, \ldots, K_{i,t}$. In order to decrypt the ciphertext, each user who belongs to a subset $S_i$ can combine these $t$ different values with his value $(K_{i,u})$ to recover $K_{i,0}$ (or $K_{i,j,0}$ in the SD scheme) and then use $K_{i,0}$ to decrypt the ciphertext. The weakness of their scheme is the same as in the Naor-Pinkas scheme. Moreover, the secret key size is still the same as in the NNL schemes, but the ciphertext size of their schemes is a multiplied factor of $t$ bigger than the ciphertext size of the NNL schemes, $t$ is the maximum number of the traitors.

The method of [ZZ11]: Interestingly, we realize that the method in [ADML+07] can resist well to Pirates 2.0 attack. In fact, in [ADML+07], the authors combined a WIBE scheme with a code based scheme. This results in a scheme in which a secret key of a user just contains only one sub-key, and thus if a user publishes his sub-key he will lose his anonymity. The weaknesses of this method are that their method does not support revocability and because of using the code the efficiency of their scheme depends on the length of the code.

In [ZZ11], the authors proposed another method. In fact, their method is similar to the method in [ADML+07] except that they split the second level in [ADML+07] into $L$ levels ($L$ is the length of code). Note that like [ADML+07], their scheme does not support revocability and the efficiency of their scheme depends on the length of the code.

### 3.4 The First Identity-based Trace and Revoke Schemes - IDTR

In this section, we first present the idea to fight against PEvoA and Pirates 2.0 and then introduce two concrete constructions [PT11]. The main problem of schemes in the tree-based framework (or more generally in the combinatorial setting) is that a user’s key contains many sub-keys or from a secret key one can derive many other sub-keys. Each sub-key corresponds to an internal node on the tree. Consequently, each sub-key can be shared among many users. Thus, if a user only publishes sub-keys of a certain level (to ensure that there are many users who also share that sub-keys), this user cannot be traced but pirates can efficiently collect different sub-keys to build a practical decoder.

Our method of fighting against Pirates 2.0 and pirate evolution attack consists of two steps: making private keys indecomposable (the private keys are not composed of sub-keys) and avoiding the derivation of sub-keys of any internal node from the private keys.

We also remark that the considered model of PEvoA and Pirates 2.0 here is not the most general one. However, this model covers the main idea of the attacks presented in [KP07] and [BP09], it also covers the proposed attacks in these original papers. Moreover, the countermeasures mentioned in [JL09], [DdP11] and [ZZ11] are all under our model. The most general model for PEvoA and Pirates 2.0 would be the case that a pirate can combine partial information about sub-keys to break the whole scheme. This model is not considered here nor in the previous papers and it is still an open question to find out
CHAPTER 3. RESISTANCE TO ADVANCED ATTACKS ON BROADCAST ENCRYPTION

a countermeasures for the most general form of PEvoA and Pirates 2.0. It appears that this closely relates to the subject of key-leakage resilient cryptography. The link between Pirates 2.0 and key-leakage resilient will be detailed in the section 3.5.

For concrete constructions, we propose the first identity-based trace and revoke schemes which rely on the subset cover framework. To this aim, we first propose a generic construction which transforms an identity-based encryption with wildcard (WIBE) of depth \( \log(N) \) (\( N \) being the number of users) into an identity-based trace and revoke scheme by relying on the complete subtree framework (of depth \( \log(N) \)). This leads, however, to a scheme with \( \log(N) \) private key size (as in a complete subtree scheme). We improve this scheme by introducing generalized WIBE (GWIBE) and propose a second construction based on GWIBE of two levels. The latter scheme provides a nice feature of having constant private key size (just 3 group elements).

Besides the property of resisting Pirates 2.0 and pirate evolution attack, our schemes are also very competitive in terms of efficiency. The only previous traitor tracing scheme in the identity-based setting is much less efficient than ours, especially when the number of traitors is large. We give the comparison between our schemes and other schemes in Table 3.1.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>PEvoA</th>
<th>Private key</th>
<th>Public key</th>
<th>Ciphertext</th>
<th>EncTime</th>
<th>DecTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS[NNL01]</td>
<td>no/no</td>
<td>( O(\log(N)) )</td>
<td>0</td>
<td>( O(r \log(\frac{N}{\epsilon})) )</td>
<td>( O(r) )</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>SD[NNL01]</td>
<td>no/no</td>
<td>( O(\log^2(N)) )</td>
<td>0</td>
<td>( O(r) )</td>
<td>( O(1) )</td>
<td></td>
</tr>
<tr>
<td>[ADML+07]</td>
<td>-/yes</td>
<td>( O(1) )</td>
<td>( O(t^2 \log(N)) ) +( \log(1/\epsilon) )</td>
<td>( O(t^2 \log(N)) ) +( \log(1/\epsilon) )</td>
<td>( O(t^2 \log(N)) ) +( \log(1/\epsilon) )</td>
<td>( O(\log(N)) )</td>
</tr>
<tr>
<td>[DF02][CS]</td>
<td>no/no</td>
<td>( O(\log(N)) )</td>
<td>( O(\log(N)) )</td>
<td>( O(r) )</td>
<td>( O(\log(N)) )</td>
<td></td>
</tr>
<tr>
<td>[DF02][SD]</td>
<td>no/no</td>
<td>( O(\log^2(N)) )</td>
<td>( O(\log(N)) )</td>
<td>( O(r) )</td>
<td>( O(\log(N)) )</td>
<td></td>
</tr>
<tr>
<td>[DdP11][CS]</td>
<td>no/yes</td>
<td>( O(\log(N)) )</td>
<td>( O(t) )</td>
<td>( O(t \cdot r) )</td>
<td>( O(t) )</td>
<td></td>
</tr>
<tr>
<td>[DdP11][SD]</td>
<td>no/yes</td>
<td>( O(\log^2(N)) )</td>
<td>( O(t) )</td>
<td>( O(t \cdot r) )</td>
<td>( O(t) )</td>
<td></td>
</tr>
<tr>
<td>[ZZ11]</td>
<td>-/yes</td>
<td>( O(1) )</td>
<td>( O(t^2 \log(N)) ) +( \log(1/\epsilon) )</td>
<td>( O(t^2 \log(N)) ) +( \log(1/\epsilon) )</td>
<td>( O(t^2 \log(N)) ) +( \log(1/\epsilon) )</td>
<td>( O(\log(N)) )</td>
</tr>
<tr>
<td>[JL09]</td>
<td>yes/-</td>
<td>( O(\log^2(N)) )</td>
<td>0</td>
<td>( O(N^{1/6}) )</td>
<td>( O(N^{1/6}) )</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>IDTR(_1)</td>
<td>yes/yes</td>
<td>( O(\log(N)) )</td>
<td>( O(\log(N)) )</td>
<td>( O(\log(N)) )</td>
<td>( O(\log(N)) )</td>
<td></td>
</tr>
<tr>
<td>IDTR(_2)</td>
<td>yes/yes</td>
<td>( O(1) )</td>
<td>( O(\log(N)) )</td>
<td>( O(\log(N)) )</td>
<td>( O(\log(N)) )</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1: Comparison between our IDTR\(_1\), IDTR\(_2\) schemes and previous schemes. The CS scheme, the SD scheme, and the scheme in [JL09] are in the private-key setting. The schemes in [ADML+07] and [ZZ11] need to fix before the setup phase the maximum number of traitors \( t \), and the tracing procedure might have an error with probability \( \epsilon \). \( r \) denotes the number of revoked users, '-' denotes “actually not known”.

The definition of an identity-based trace and revoke scheme IDTR, as well as the adaptive security model for IDTR systems, can be found in our paper [PT11]. Our model follows the model of identity-based traitor tracing [ADML+07] in which each group of users has
3.4. THE FIRST IDENTITY-BASED TRACE AND REVOKE SCHEMES - IDTR

an identity and the broadcaster encrypts messages to a group by using the group’s identity. We also deal with revocation, the encryption depends thus on the group’s identity and the set of revoked users in that group.

3.4.1 Framework

**First Step: Making Private Keys Indecomposable**

Since the core of Pirates 2.0 attack is the public contribution of partial private key, a natural idea is to render private keys indecomposable. But this condition is not sufficient. Asano [Asa02] and [AK05] propose very interesting schemes with constant size private keys. Because private keys in these schemes are of constant size, one might think traitors cannot contribute a partial key information to the public and that Pirates 2.0 can then be excluded. However, this is not the case. Indeed, in order for a user $U$ to be able to decrypt a ciphertext at a node in the path from $U$ to the root, $U$ has to be able to derive the corresponding key at that node. This is the reason why these schemes are still vulnerable to Pirates 2.0 and pirate evolution attack. The user $U$ does not need to contribute his key but can compute and contribute any key corresponding to the nodes on the path from $U$ to the root.

**Second Step: No Intermediate Key Could be Derived From a Private Key**

The goal in this step is to construct a scheme that, along with the undecomposability of private keys, enjoying the following additional property: no key at internal nodes can be derived from a private key. This property ensures preventing a traitor to derive and then publish keys at internal nodes and therefore it can only contribute its own key which leaks its identity.

**Main Idea: Delegation of Ciphertexts**

Our main idea is to allow a sub-ciphertext at a node being publicly modifiable to be another sub-ciphertext at any descendant node. This allows a user, in possessing a private key at a leaf, to decrypt any sub-ciphertext at its ancestor node. We illustrate the idea in the figure 3.4. The sub-ciphertext $C$ could be publicly modified to be a ciphertext $C_U$ that can be decryptable by any user $U$ lying in the sub-tree rooted at the corresponding node $C$. At this stage, we see that the user $U$ could have just one key which cannot be used to derive sub-keys for other nodes but it can decrypt ciphertexts at any nodes in the path from $U$ to the root.

The construction of WIBE in [ACD+06] implicitly gives a delegation of ciphertexts as we need. This is indeed our first solution which is quite inefficient. We then introduce a generalized notion of WIBE and propose a much more efficient solution.
CHAPTER 3. RESISTANCE TO ADVANCED ATTACKS ON BROADCAST ENCRYPTION

Figure 3.3: Asano’s method: from a private key, a user $U$ can derive keys at any nodes on the path from its leave to the root.

Figure 3.4: Our method: it can be seen as a dual of Asano’s method where a ciphertext at a node can be modified to be decryptable with keys at lower nodes.
3.4. THE FIRST IDENTITY-BASED TRACE AND REVOKE SCHEMES - IDTR

3.4.2 Solutions

First Solution: Integration of WIBE into a Complete Subtree Scheme

In our framework 3.4.1, the first step is to propose a scheme so that a user is unable to derive any key corresponding to its ancestor node. This requirement fits perfectly hierarchical identity based encryption systems (HIBE for short). However, an HIBE system cannot be directly employed for dealing with the second step because a user at a lower level cannot decrypt a ciphertext at a higher level. Fortunately, identity-based encryption with wildcards [ACD+06] can be used to solve this issue. The main idea in constructions of current WIBE is to be able to modify the ciphertext so that a legitimate user can put information on the ciphertext to obtain a new ciphertext specifically crafted for him and then can decrypt this modified ciphertext as in HIBE system. This completes the second step.

Our first scheme incorporates a special WIBE with complete subtree structure. In the WIBE used in our first scheme, at each level in hierarchy, there are just two possible identities, say 0 and 1. If we want to encrypt with a node $S$, we simply put wildcards for lower levels of $S$ and consequent, any users in the subtree rooted by $S$ is able to decrypt. This is illustrated in figure 3.5.

The current constructions of WIBE are mainly based on HIBE. Therefore, a private key of a user cannot be used to derive keys of nodes at higher level. This completes the first step. The main idea in constructions of current WIBE is to be able to modify the ciphertext so that a legitimate user can put information on the ciphertext to obtain a new ciphertext specifically crafted for him and then can decrypt this modified ciphertext as in HIBE system. This completes the second step.
 CHAPTER 3. RESISTANCE TO ADVANCED ATTACKS ON BROADCAST ENCRYPTION

Main Solution: Integration of 2-levels of GWIBE into a Complete Subtree Scheme

To improve the private key size, we replace the WIBE by 2-levels GWIBE and the resulting scheme has a nice property, the constant size private key. Figure 3.7 illustrates our method.

3.4.3 Our First Construction

Our first method is we combine WIBE with the complete subtree method: each group $ID \in \{0, 1\}^*$ represents a binary tree and each user $id \in \{0, 1\}^l$ (we write $id = id_1 id_2 \cdots id_l$, $id_i \in \{0, 1\}$) in a group $ID$ is assigned to be a leaf of the binary tree rooted at $ID$. For encryption, we will use a WIBE of depth $l + 1$, each user is associated with a vector $(ID, id_1, \cdots, id_l)$. This is illustrated in Figure 3.6.

Concretely, the WIBE-IDTR system works as follows:

**Setup**($\lambda, N$): Take a security parameter $\lambda$ and the maximum number of user in each group $N$ (thus $l = \lceil \log_2 N \rceil$). Run the setup algorithm of WIBE with the security parameter $k$ and the hierarchical depth $L = l + 1$ which returns (MPK, MSK). The setup then outputs (MPK, MSK). As in the complete subtree method, the setup also defines a data encapsulation method $E_K : \{0, 1\}^* \rightarrow \{0, 1\}^*$ and its corresponding decapsulation $D_K$. The session key $K$ used will be chosen fresh for each message $M$ as a random bit string. $E_K$ should be a fast method and should not expand the plaintext.

**Extract**(MSK, $ID, id$): Run the key derivation of WIBE for $l + 1$ level identity $WID = (ID, id_1, id_2, \cdots, id_l)$ (the $j$-th component corresponds to the $j$-th bit of the identity $id$) and get the decryption key $d_{WID}$. Output $d_{ID, id} = d_{WID}$.

**Encrypt**(MPK, $ID, R_{ID}, M$): A sender wants to send a message $M$ to a group $ID$ with the revocation list $R_{ID}$. The revocation works as in the complete subtree scheme. Considering a group $ID$ with its revocation list $R_{ID}$, the users in $N_{ID} \setminus R_{ID}$ are partitioned into disjoint subsets $S_{i_1}, \ldots, S_{i_w}$ which are all the subtrees of the original tree (rooted at $ID$) that hang off the Steiner tree defined by the set $R_{ID}$.

Each subset $S_{i_j}, 1 \leq j \leq w$, is associated to an $l + 1$ vector identity $ID_{S_{i_j}} = (ID, id_{i_j, 1}, \ldots, id_{i_j, k}, *, *, \ldots, *)$ where $id_{i_j, 1}, \ldots, id_{i_j, k}$ is the path from the root $ID$ to
3.4. THE FIRST IDENTITY-BASED TRACE AND REVOKE SCHEMES - IDTR

the node $S_i$ and the number of wildcards $*$ is $l - k$. The encryption algorithm randomly chooses a session key $K$, encrypts $M$ under the key $K$ by using an symmetric encryption, and outputs as a header the encryption of WIBE for each $ID_{S_i}$, ..., $ID_{S_w}$.

$$C = \langle [i_1, \ldots, i_w] | \text{WIBE.Encrypt}(\text{MPK}, ID_{S_i}, K), \ldots, \text{WIBE.Encrypt}(\text{MPK}, ID_{S_w}, K) \rangle, \ E_K(M) \rangle$$

$$C = \langle [i_1, \ldots, i_w] | \text{WIBE.Encrypt}(\text{MPK}, ID_{S_i}, K), \ldots, \text{WIBE.Encrypt}(\text{MPK}, ID_{S_w}, K) \rangle, \ E_K(M) \rangle$$

**Decrypt**($d_{ID,id}, C$): The user received the ciphertext $C$ as above. First, find $j$ such that $id \in S_i$ (in case $id \in R_{ID}$ the result is null). Second, use private key $d_{ID,id}$ to decrypt $\text{WIBE.Encrypt}(\text{MPK}, ID_{S_i}, K)$ to obtain $K$. Finally, compute $D_K(E_K(M))$ to recover the message $M$.

**Trace**($D_{MSK, ID}$): Our tracing algorithm is similar to the one in [NNL01]. It takes as input $msk, ID$, an illegal decryption box $D$, returns either a subset consisting at least one traitor or a new partition of $N_{ID} \backslash R_{ID}$ that renders the illegal decryption box useless. We note that this is not the strongest notion of traceability. In a black-box traitor tracing, one can identify at least one traitor. Our tracing algorithm relies on the tracing algorithms in NNL framework and target the same notion of traceability. However, unlike the tracing algorithms in NNL, our tracing algorithm can deal with PEvoA and Pirates 2.0.

**Security of WIBE-IDTR**

**Theorem 3.4.1** If the WIBE of depth $l$ is $(t', q_K', \epsilon')$ IND-CPA-secure, then the WIBE-IDTR is $(t^*, q_K^*, \epsilon^*)$ IND-CPA-secure with

$$t^* \leq t' / (r \log(\frac{N}{r})), \quad q^*_K \leq q'_K, \quad \epsilon^* \leq r \log(\frac{N}{r}) \times \epsilon'$$

where $N$ is the bound on the number of users and $r$ is the number of revoked users.

**Proof**: We organize our proof as a sequence of games. The first game **Game 0** defined will be the real identity- based trace and revoke game and the last one will be one in which the adversary has no advantage unconditionally. We will show that each game is indistinguishable from the next, under the assumption of the security of WIBE.

**Game 0**: This is the real attack game of an adversary $B$ against the proposed identity based trace and revoke system. After receiving the public key $\text{mpk}$, $B$ issues adaptively key derivation queries ($ID, id$). The challenger responds by running $\text{IDTR.Extract}(\text{MSK}, ID, id)$ algorithm to generate the private key $d_{ID,id}$ and passes $d_{ID,id}$ to $B$.

$B$ finally outputs two equal length plantexts $M_0, M_1 \in \mathcal{M}$ and a targeted set $ID^*$. The revoked set $R_{ID^*}$ consists the users’ identity $id$ such that $(ID^*, id)$ has been asked by the adversary $B$.

The challenger picks then a random bit $b \in \{0, 1\}$ and set $C = \text{IDTR.Encrypt}(\text{MPK}, N_{ID^*} \backslash R_{ID^*}, M_b)$. It sends $C$ as the challenge to $B$. 78
Upon receiving the challenge $C$, $B$ outputs a guess $b' \in \{0, 1\}$. $B$ wins the game if $b' = b$. More precisely, $B$ wins if its advantages, defined as below, of guessing the bit $b$ is non-negligible

$$\text{Adv}^{\text{idtr-wibe}}_{\text{ind}}(t) = |2 \times Pr[B(\text{IDTR.Encrypt}(\text{MPK}, N_{ID^*}/R_{ID^*}, M_b)) = b] - 1|$$

which can be equivalently written as:

$$\text{Adv}^{\text{idtr-wibe}}_{\text{ind}}(B) = \left| Pr[B(\text{IDTR.Encrypt}(\text{MPK}, N_{ID^*}/R_{ID^*}, M_1)) = 1] - Pr[B(\text{IDTR.Encrypt}(\text{MPK}, N_{ID^*}/R_{ID^*}, M_0)) = 1] \right|$$

In our construction, the encryption of trace and revoke system is performed as:

$$\text{IDTR.Encrypt}(\text{MPK}, N_{ID^*}/R_{ID^*}, M) = (\text{WIBE.Encrypt}(\text{MPK}, I_{SD_1}, M), \cdots, \text{WIBE.Encrypt}(\text{MPK}, I_{SD_w}, M))$$

where $N_{ID^*}/R_{ID^*}$ is partitioned to be $w$ subsets corresponding to nodes $I_{SD_1}, \cdots, I_{SD_w}$

In the following games, we will modify step by step the challenge given to the adversary. We define a modified encryption $\text{IDTR.Encrypt}^k$ as follow:

$$\text{IDTR.Encrypt}^k(\text{MPK}, N_{ID^*}/R_{ID^*}, M) = (\text{WIBE.Encrypt}(\text{MPK}, I_{SD_1}, M_0), \cdots, \text{WIBE.Encrypt}(\text{MPK}, I_{SD_k}, M_0), \text{WIBE.Encrypt}(\text{MPK}, I_{SD_{k+1}}, M), \cdots, \text{WIBE.Encrypt}(\text{MPK}, I_{SD_w}, M))$$

Note that

$$\text{IDTR.Encrypt}^0(.) = \text{IDTR.Encrypt}(.)$$

$$\text{IDTR.Encrypt}^k(\text{MPK}, N_{ID^*}/R_{ID^*}, M_0) = \text{IDTR.Encrypt}(\text{MPK}, N_{ID^*}/R_{ID^*}, M_0) \text{ for any } k$$

$$\text{IDTR.Encrypt}^w(\text{MPK}, N_{ID^*}/R_{ID^*}, M) = \text{IDTR.Encrypt}(\text{MPK}, N_{ID^*}/R_{ID^*}, M_0) \text{ for any } M$$

**Game_k for $k = 1, 2, \ldots, w$:** This is the same as in the game $k-1$ with an exception that the challenger use $\text{IDTR.Encrypt}^k(.)$ instead of $\text{IDTR.Encrypt}^{k-1}(.)$. We call $\text{Adv}^{\text{idtr-wibe}}_{\text{ind,k}}(B)$ the advantage of the adversary $B$ in Game $k$. We remark that, in the game $w$, the adversary $B$ has zero advantage because $B$ receives two ciphertext of the same message $M_0$. Therefore, the proof directly holds under the following lemma:

**Lemma 3.4.2**

$$\text{Adv}^{\text{idtr-wibe}}_{\text{ind,k}}(B) - \text{Adv}^{\text{idtr-wibe}}_{\text{ind,k-1}}(B) \leq \epsilon^*,$$

where $\epsilon^*$ is the bound on the advantages of the adversaries against $\text{WIBE}$.

**Proof:** We will construct an adversary $B'$ that breaks the IND-CPA security of the underlying $\text{WIBE}$ with an advantage of $\text{Adv}^{\text{idtr-wibe}}_{\text{ind,k}}(B) - \text{Adv}^{\text{idtr-wibe}}_{\text{ind,k-1}}(B)$.

**Setup:** The challenger of $B'$ runs setup algorithm of $\text{WIBE}$ to generate key pair $(\text{MPK}, \text{MSK})$. It sends $\text{MPK}$ to $B'$ and keeps $\text{MSK}$ private. $B'$ passes this $\text{MPK}$ to $B$. 

79
Phase 1: When $B$ asks key query for an user $id = id_1 \ldots id_l$ in a group $ID$, $B'$ sends the query $WID = (ID, id_1, \ldots, id_l)$ (a $(l+1)$-vector) to its challenger and gets a key $d_{WID}$. $B'$ passes this key $d_{WID}$ to $B$. It assures the correctness because in the construction $d_{ID,id}$ is defined to be $d_{WID}$ in the same way.

For each query on $(ID, id)$, $B'$ updates the revocation list for group $ID$ by adding $id$ to $R_{ID}$ (initially empty).

Challenge: The adversary $B'$ submits two equal length messages $M_0, M_1$ and an identity $ID^*$. The challenger picks a random bit $b \in \{0, 1\}$ and set $C = \text{Encrypt}(msk, ID^*, R_{ID^*}, M_b)$. The ciphertext $C$ is passed on to the adversary. $B'$ partitions $N_{ID^*} \setminus R_{ID^*}$ to $(S_{i_1}, S_2, \ldots, S_{i_w})$ as in the original Game$_0$. $B'$ submits $M_0, M_1$ and the identity $ID_{S_{i_k}}$ to its challenger and receives a challenge ciphertext $C_b = \text{WIBE.Encrypt}(\text{MSK}, ID_{S_{i_k}}, M_b)$. $B'$ encrypts $M_0$ for identities $ID_{S_{i_1}}, \ldots ID_{S_{i_{k-1}}}$ and encrypts $M_1$ for identities $ID_{S_{i_{k+1}}}, \ldots ID_{S_{i_w}}$. $B'$ finally gives $B$ the following challenge ciphertext:

$$(\text{WIBE.Encrypt}(\text{MPK}, ID_{S_{i_1}}, M_0), \ldots, \text{WIBE.Encrypt}(\text{MPK}, ID_{S_{i_{k-1}}}, M_0), C_b, \text{WIBE.Encrypt}(\text{MPK}, ID_{S_{i_{k+1}}}, M) \ldots, \text{WIBE.Encrypt}(\text{MPK}, ID_{S_{i_w}}, M))$$

Phase 2: $B'$ responses $B$’s key queries in a similar way to the Phase 1. As $B$ is not allowed to ask queries on $ID^*$, $B'$ will not make queries on the targeted identity.

Guess: When $B$ gives its guess, $B'$ outputs the same guess. We realizes that, when $b = 0$, the adversary $B$ exactly plays the Game$_{k-1}$ and when $b = 1$, the adversary $B$ exactly plays the Game$_k$. Therefore, the advantage of $B'$ is $|\text{Adv}_{idtr-wibe}^{idtr-wibe}(B) - \text{Adv}_{idtr-wibe}^{idtr-wibe}(B)|$.

3.4.4 Resistance to Pirate Evolution Attack - PEvoA

In our definition, we closely follow the formalization of [KP07] but we refine the user’s private key and evolution step: each user possesses a set of decryption keys, and at each iteration $i$ a pirate box $B_i$ must contain at least a decryption key which is used to decrypt the ciphertext encrypted at that iteration. This consideration might fail to cover attacks who can derive some partial decryption keys from a set of its key and then could use the partial key to break the security. In any case, our formalization covers the idea of the type of attacks described in [KP07]: each user in [NNL01] possesses a set of decryption keys (a set of long lived keys), and at each iteration the master pirate box $B$ spawns a new version of pirate box $B_i$, where $B_i$ contains at least a decryption key (a long lived key) at some node in the tree.

We also note that the previous schemes in the subset-cover framework are susceptible to pirate evolution attack because each user can always possess (or derive, in Asano et. al. schemes [Asa02, AK05]) keys corresponding to intermediate nodes in the tree. Therefore,
the traitors can easily produce many pirate boxes. This way of attacks are covered in our formalization.

We now show that our WIBE-IDTR scheme resists pirate evolution attack.

**Theorem 3.4.3 (Resistance of WIBE-IDTR to Pirate Evolution Attack)** If there exists a pirate evolution adversary \( B \) on WIBE-IDTR of depth \( L \), then there exists an adversary \( B' \) that breaks the security of HIBE.

**Proof:** The main idea how our scheme can resist to this type of attacks is come from the property of the underlying HIBE. In our scheme, each user has only a private key corresponding to its identity at depth \( L \). By the property of HIBE, no key at any intermediate node can be derived from the users’ key. Therefore, the traitors can only produce pirate box by putting in it their collected keys and the number of pirate box is thus the number of traitors.

Formally, we first prove the following simple lemma.

**Lemma 3.4.4** Considering a WIBE-IDTR scheme of depth \( L \), if there is an adversary \( B \), by corrupting \( t \) user private keys \( (d_{ID_1}, \ldots, d_{ID_t}) \) for its chosen identities \( (ID_1, \ldots, ID_t) \) of depth \( L \), is able to derives a new private key \( d_{ID^*} \) for an identity \( ID^* \) different than \( (ID_1, \ldots, ID_t) \), then one can construct an adversary \( B' \) that breaks the semantic security of HIBE.

**Proof:** The proof is simple, we run \( B' \) to simulate \( B \) and use the output of \( B \) to break the security of HIBE.

- In the setup phase, the challenger of \( B' \) runs setup algorithm of HIBE to create the public parameters and secret parameters. It gives the public parameters to the \( B' \), \( B' \) passes these public parameters to the \( B \). The public parameters in two systems are the same.

- Phase 1: \( B \) chooses arbitrarily \( t \) identities \( (ID_1, \ldots, ID_t) \) where each identity \( ID_i \) \( (i = 1, \ldots, t) \) is an identity of depth \( L \). \( B' \) queries its challenger key derivation oracle \( t \) times on such identities and returns the results to \( B \).

- The challenge phase: When \( B \) outputs the private key \( d_{ID^*} \) corresponding to identity \( ID^* \), \( B' \) submits two equal length messages \( M_0, M_1 \) and identity \( ID^* \) to its challenger. \( B' \) can use \( ID^* \) as a target identity because \( ID^* \) and its prefix never be queried on the phase 1. The challenger picks a random bit \( b \in \{0, 1\} \) and set \( C = \text{Encrypt}(ID^*, M_b) \). The ciphertext \( C \) is passed to \( B' \).

- Phase 2: \( B' \) uses \( d_{ID^*} \) to decrypt \( C \) and outputs \( b \).

From the above simulation, we induce that if there exists adversary \( B \) then there exists adversary \( B' \) with the same probability.

Assume now that the adversary \( B \) corruptions \( t \) traitor keys \( (d_{u_1}, \ldots, d_{u_t}) \). The above lemma shows that \( B \) cannot derive a new decryption key \( d_{w} \) not in \( (d_{u_1}, \ldots, d_{u_t}) \). Therefore, at each iteration \( i \) when \( B \) spawns a new version of pirate box decoder \( B_i \), \( B_i \) must contain at least a new key among the \( t \) traitor keys. This implies that the maximum number of pirate box decoders that \( B \) can spawn is \( t \).
3.4.5 Resistance to Pirates 2.0

We consider the Pirates 2.0 model for subset-cover framework. Recall that, in this framework, a traitor is a legitimate user \( u_i \) who possesses a set of decryption keys \( K_{u_i} = (K_{u_i,1}, \ldots, K_{u_i,n}) \). We suppose that traitors leak a subset of their decryption keys to the public domain in a discrete process. When pirates want to decrypt a ciphertext \( C \), they should collect relevant decryption keys from their public environment in order to produce a pirate decoder. We consider that a pirate decoder should contain at least a decryption key that can be based on it to decrypt \( C \). Our consideration, though not in the most general form, exactly reflects the attacks described in [BP09] against subset-cover schemes.

We also note that the previous schemes in the subset-cover framework are susceptible to Pirates 2.0 attack because each user can always possess (or derive as in Asano et. al's schemes [Asa02, AK05]) keys corresponding to intermediate nodes in the tree. Therefore, the traitors can easily contribute to the public domain a subset of their decryption keys leading to the production of a pirate decoder. These attacks are covered by our considered formalization.

Following the same argument as in the case of pirate evolution attacks, we can show that our WIBE-IDTR scheme resists Pirates 2.0.

**Theorem 3.4.5 (Resistance of WIBE-IDTR to Pirates 2.0)** For any Pirates 2.0 adversary \( B \) against WIBE-IDTR of depth \( L \), there exists an efficient algorithm to output at least a traitor in the collusion that produced \( B \).

**Proof:** In our scheme, each user has only a private key corresponding to its identity at depth \( L \). By Lemma 3.4.4, no key at any intermediate node can be derived from the users’ key. Consider now a traitor \( u_i \) contributing a subset of decryption keys \( S \) of \( K_{u_i} \) to the public domain. In this case, because \( K_{u_i} \) is its single decryption key, this immediately implies that \( S \) is the leave at user \( u_i \). By definition, the anonymity level of \( u_i \) is 1, meaning that \( u_i \) is traced to be a traitor.

3.4.6 Our Second Construction

A shortcoming in our first scheme is that the private key size is linearly in the depth of the tree. To overcome this issue, we propose a new primitive, called generalized WIBE (GWIBE for short) where the wildcard does not need to replace the whole identity at a given level but can be part of the identity at each level. Figure 3.7 illustrates our method. We group all levels below the root into a single level and put the wildcard as part of the identity. Therefore, we only need a 2-level GWIBE for encryption. We can thus regroup all levels below the root into a single level and put the wildcard as part of the identity. Therefore, we only need a 2-level GWIBE for encryption.
CHAPTER 3. RESISTANCE TO ADVANCED ATTACKS ON BROADCAST ENCRYPTION

Identity-Based Encryption With Generalized Wildcards - GWIBE

Identity-based encryption with generalized wildcards schemes are essentially generalizations of WIBE schemes. The difference is that, in WIBE, at each level, the wildcard is added to replace the whole identity at that level, while in GWIBE, the wildcard might be employed to replace a part of the identity.

More formally, at the time of encryption, the sender can decide to make the ciphertext decryptable by a whole range of users whose identities match a certain pattern. For an l-level GWIBE, such a pattern is described by a vector \( P = (P_1, \ldots, P_l) \). Each \( P_i, 1 \leq i \leq l \), is a bit string of length \( n \), and can contain a wildcard. The wildcard is determined by its starting position and its final position. For example, for an identity “univ*math” of size 16, the starting position of the wildcard is 5 and the final position is 12, the size of the wildcard is 8. When encrypting with the identity “univ*math”, any user who has a key corresponding to identities “univ-paris6.math” or “univ-paris9.math”, ... should be able to decrypt.

The starting position is in \( \{1, 2, \ldots, n, n+1\} \) with the convention that the value \( n+1 \) implies no wildcard in \( P_i \). If \( P_i \) contains a wildcard (that means that a starting position is between 1 and \( n \)) we call \( P_i \) a generalized wildcard. At this point, each \( P_i \) can be rewritten as \( P_i = P_{ip} || \ast || P_{is} \) where \( P_{ip}, P_{is} \) are the bit strings of length \( ip, is \); \( 0 \leq ip, is \leq n \). The size of the wildcard is \( n - ip - is \), its starting position is \( ip + 1 \) and its final position is \( n - is \). We say that an identity \( ID = (ID_1, \ldots, ID_{l'}) \) matches \( P \), denote \( ID \in_{\ast} P \), if and only if \( l' \leq l \) and \( \forall i = 1, \ldots, l': ID_i = P_i \) or \( ID_i = P_i \) and \( P_i \) have the form \( P_i = P_{ip} || \ast || P_{is} \) and \( ID_i = P_{ip} || ID_{i} || P_{is} \) where \( ID_{i} \) is some bit string of length \( n - ip - is \). From the above generalized notions of pattern and matching, the definition and the security model of GWIBE follows exactly the one of WIBE.

The generic construction of IDTR from 2-Level GWIBE is similar to the generic construction of IDTR from WIBE in Section 3.4.3. The only difference is that, instead of encrypting with WIBE, we will encrypt with 2-level GWIBE. The Theorem 3.4.1 can be adapted as follows.
Theorem 3.4.6 If the GWIBE of depth 2 is \((t', q_K', \epsilon')\) IND-CPA-secure, then the 2level-GWIBE-IDTR is \((t^*, q_K^*, \epsilon^*)\) IND-CPA-secure with

\[ t^* \leq t'/(r \log(\frac{N}{r})), \quad q_K^* \leq q_K', \quad \epsilon^* \leq r \log(\frac{N}{r}) \times \epsilon' \]

where \(N\) is the upper bound on the number of users in the system and \(r\) is the number of revoked users.

The main contributions in this section are about constructing a GWIBE scheme and then applying it to obtain an efficient trace and revoke system. We now detail these parts.

Concrete Construction of GWIBE Based on Waters’ HIBE (Wa-GWIBE)

We first explain some notation and the high level of the construction. A user’s identity is given by a vector \(ID = (ID_1, ID_2, \ldots, ID_l)\) where each \(ID_i\) is an \(n - \)bit string, applying a collision resistant hash function if necessary. When we write \(j \in ID_i\), we mean that the variable \(j\) iterates over all bit positions of string of \(ID_i\) where the \(j\)-th bit of \(ID_i\) is one. Using this notation, for \(i = 1, \ldots, l\), the Waters’ hash function \(F_i\) is defined as

\[ F_i(ID_i) := u_{i,0} \prod_{j \in ID_i} u_{i,j} \]

where the \(u_{i,j}\) are the elements in the master public key.

Now, for a generalized wildcard \(ID_{ip}|| * ||ID_{is}\), we define \(F_i(ID_{ip}|| * ||ID_{is}) := F_i(ID_{ip}||0\ldots0||ID_{is})\) with the number of added bit 0 is the size of the wildcard \(*\). The main remark is that if we have the values \(u_{i,j}\) for all \(j = ip + 1, \ldots, n - is\) and \(t \overset{\$}{\leftarrow} \), we can compute the values of \(F_i(ID_{ip}||ID^*||ID_{is})\) from \(F_i(ID_{ip}|| * ||ID_{is})\) for any bit string \(ID^*\) of length \(n - ip - is\).

Making use of this property, the main idea is that, if we postpone the computation of elements of wildcard to decryption time by including the separate elements \(u_{i,j}\) corresponding to wildcards in the ciphertext, each recipient can combine the factors corresponding to his own identity in the decryption.

We extend some notations from WIBE [ACD+06]. Let \(P = (P_1, \ldots, P_l)\) is a pattern where \(P_i\) is a string of length \(n\), define \(W(P)\) be the set containing all generalized wildcard indices in \(P\), i.e. the indices \(1 \leq i \leq l\) such that \(P_{ip}|| * ||P_{is}, 1 \leq ip \leq n\). Let \(\overline{W}(P)\) be the complementary set containing all non-generalized wildcard indices in \(P\). Clearly \(W(P) \cap \overline{W}(P) = \phi\) and \(W(P) \cup \overline{W}(P) = \{1, \ldots, l\}\).

Let \(\mathbb{G}, \mathbb{G}_T\) be multiplicative groups of prime order \(p\) with an admissible map \(\hat{e} : \mathbb{G} \times \mathbb{G} \leftarrow \mathbb{G}_T\).

Setup: The trusted authority chooses randomly generators \(g_1, g_2, u_{1,0}, \ldots, u_{L,n} \overset{\$}{\leftarrow} \mathbb{G}^*\), and a random value \(\alpha \overset{\$}{\leftarrow} \mathbb{Z}_p^*\), where \(L\) is the maximum hierarchy depth. Next, it computes \(h_1 \leftarrow g_1^\alpha\) and \(h_2 \leftarrow g_2^\alpha\). The master public key is \(\text{MPK} = (g_1, g_2, h_1, u_{1,0}, \ldots, u_{L,n})\), and the master secret key is \(\text{MSK} = h_2\).
For each level \( i = 1, \ldots, L \), a function \( F_i : \{0,1\}^n \rightarrow \mathbb{G}^* \) is defined as:

\[
F_i(ID_i) = u_{i,0} \prod_{j \in ID_i} u_{i,j}
\]

**Extract:** An identity of a user is given by a vector \( ID = (ID_1, \ldots, ID_L) \), where \( ID_i \) is a bit string of length \( n \) (applying a collision-resistant hash function if necessary). First, random values \( r_1, \ldots, r_L \) are chosen, then the private key \( d_{ID} \) is computed as

\[
(a_0, a_1, \ldots, a_l) = \left( h_2 \prod_{i=1}^l F_i(ID_i)^{r_i}, g_1^{r_1}, \ldots, g_1^{r_l} \right)
\]

A private key for identity \( ID = (ID_1, ID_2, \ldots, ID_L) \) can be computed by its parent whose identity is \( ID = (ID_1, ID_2, \ldots, ID_{l-1}) \) as follows. Let \( (a_0, a_1, \ldots, a_{l-1}) \) be the parent’s private key. It chooses \( r_l \) and outputs

\[
d_{ID} = (a_0 \times F_l(ID_l)^{r_l}, a_1, \ldots, a_{l-1}, g_1^{r_l})
\]

**Encrypt:** To encrypt a message \( m \) to all identities matching pattern \( P = (P_1, \ldots, P_l) \), the sender chooses \( t \) and outputs the ciphertext \( C = (P, C_1, C_2, C_3, C_4) \), where

\[
C_1 \leftarrow g_1^{t} \quad C_2 \leftarrow (C_{2,i} = F_i(P_i)^t)_{i \in W(P)} \quad C_3 \leftarrow m \cdot \hat{e}(h_1, g_2)^t
\]

\[
C_4 \leftarrow (C_{4,ip,is} = F_i(P_{ip}[0...0][P_{is}]^t), (C_{4,ij} = u_{i,j})_{j = ip+1, \ldots, n-is})_{i \in W(P)}
\]

Where \( P_{ip}[0...0][P_{is}] \) is a string of length \( n \), the number of bit 0 is \( n - ip - is \). That means that we replace the elements of wildcard in identity \( P_i \) by bit 0.

**Decrypt:** If the receiver is the root authority holding the master key \( MSK = h_2 \), then it can recover the message by computing \( C_3/\hat{e}(C_1, h_2) \). Any other receiver with identity \( ID = (ID_1, \ldots, ID_L) \) matching the pattern \( P \) to which the ciphertext was created (i.e., \( ID \in \mathbb{W} \)) can decrypt the ciphertext \( C = (P, C_1, C_2, C_3, C_4) \) by first computes \( C_2' = (C_{2,i})_{i = 1, \ldots, l} \) as

\[
C_{2,i} = F_i(ID_i)^t \left\{ \begin{array}{ll}
C_{2,i} & \text{if } i \in W(ID) \\
C_{4,ip,is} \times \prod_{j = ip+1, \ldots, n-is} C_{4,i,j} & \text{if } i \notin W(ID)
\end{array} \right.
\]

then computes

\[
\frac{\prod_{i=1}^l \hat{e}(a_i, C_{2,i})}{\hat{e}(C_1, a_0)} = m \cdot \hat{e}(h_1, g_2)^t \cdot \frac{\prod_{i=1}^l \hat{e}(g_i^{r_i}, F_i(ID_i)^t)}{\hat{e}(g_1^{r_1}, h_2) \cdot \prod_{i=1}^l \hat{e}(F_i(ID_i)^{r_i})}
\]

\[
= m \cdot \hat{e}(g_1^{r_1}, g_2)^t \cdot \frac{\prod_{i=1}^l \hat{e}(g_i^{r_i}, F_i(ID_i)^t)}{\hat{e}(g_1^{r_1}, g_2)^a \cdot \prod_{i=1}^l \hat{e}(F_i(ID_i)^{r_i})} = m
\]

85
Security Analysis The proof of the following theorem can be found in our paper [PT11].

**Theorem 3.4.7** If the Wa-HIBE of depth $L$ is $(t, q_K, \epsilon)$ IND-CPA-secure, then the Wa-GWIBE scheme of depth $L$ is $(t', q'_K, \epsilon')$ IND-CPA-secure

\[ t' \leq t - L \cdot n(1 + q_K) \times t_{exp}, \quad q'_K \leq q_K, \quad \epsilon' \leq \epsilon \times n^{2L} \]

where $t_{exp}$ is the time taken to perform an exponentiation in $G$.

The Wa-GWIBE used in our Trace and Revoke system. We first remark that the above reduction of a general construction of GWIBE is extremely costly. Fortunately, in our construction of a trace and revoke system, we only need to a 2-level Wa-GWIBE and we can get a much more efficient reduction. Let us briefly explain the presence of the factor $n^{2L}$ in the reduction of the above theorem. In fact, at each level the simulator has to guess the start position and the final position of wildcard (note that if the start position is $n + 1$ then it means there is no wildcard at this level). The probability of a good guess at a level is $1/n^2$. Because there are $L$ levels, therefore the probability that $A$ guesses $W(P^*)$ correctly is $1/n^{2L}$. We now see how we can reduce this reduction cost for the purpose of a trace and revoke system. Indeed, we only need to a 2-level Wa-GWIBE and moreover, the first layer (the group identity) does not contain wildcard. The used pattern $P$ used in encryption always has the form $P = (P_1, P_2||\ast)$. Therefore, in the proof of the above theorem, the simulator does not need to guess the position of the wildcard in the first layer, and at the second level, $B$ needs only to guess the starting position of the wildcard. Therefore, the factor $1/n^{2L}$ becomes $1/n$ in this case. This is acceptable because $n = \log(N)$ where $N$ is the number of users in the system.

We conclude that the $(t', q'_K, \epsilon')$ IND-CPA security of 2-level Wa-GWIBE can thus be reduced to a $(t, q_K, \epsilon)$ IND-CPA-security of a 2-level Wa-HIBE with:

\[ t' \leq t - \log(N)(1 + q_K) \times t_{exp}, \quad q'_K \leq q_K, \quad \epsilon' \leq \epsilon \times \log(N). \]  \hfill (3.1)

**Generic Construction of IDTR From 2-Level GWIBE (2level-GWIBE-IDTR)**

**Setup($\lambda$):** Take a security parameter $\lambda$ and the maximum number of users in each group $N$. Run the setup algorithm of GWIBE with the security parameter $\lambda$ and the hierarchical depth $l = 2$ which returns $(MPK, MSK)$. The setup then outputs $(MPK, MSK)$.

Similar to the scheme in 3.4.3, the setup also defines a method $(E_K, D_K)$ to encrypt the message itself.

**Extract(MSK, ID, id):** Run the key derivation of GWIBE for 2 level identity $WID = (ID, id)$ and get the decryption key $d_{WID}$. Output $d_{ID, id} = d_{WID}$.

**Encrypt(MPK, ID, R_{ID}, M):** A sender wants to send a message $M$ to a group $ID$ with the revocation list $R_{ID}$. The revocation works as in the complete subtree scheme. Considering a group $ID$ with its revocation list $R_{ID}$, the users in $N_{ID} \setminus R_{ID}$ are partitioned into disjoint subsets $S_{i_1}, \ldots, S_{i_w}$ which are all the subtrees of the original tree (rooted at $ID$) that hang off the Steiner tree defined by the set $R_{ID}$. 

86
Theorem 3.4.8 If the GWIBE of depth 2 is (\(t', q_K, \epsilon')\) IND-CPA-secure, then the 2level-GWIBE-IDTR is \((t^*, q_K, \epsilon^*)\) IND-CPA-secure with

\[
t^* \leq t' / (r \log(\frac{N}{r})), \quad q_K^* \leq q_K, \quad \epsilon^* \leq r \log(\frac{N}{r}) \times \epsilon'
\]

where \(N\) is the upper bound on the number of users in the system and \(r\) is the number of revoked users.

Proof: We deduce the proof of this theorem from the proof of Theorem 3.4.1.

Concrete Construction of IDTR from 2-Level Wa-GWIBE (2level-Wa-GWIBE-IDTR)

Applying the above construction of 2-level-Wa-GWIBE to the generic construction of IDTR from 2-level-GWIBE, we obtain a 2level-Wa-GWIBE-IDTR. We give the concrete construction in our paper [PT11].

Concerning the security of the scheme, from Theorem 3.4.7 and the equation 3.1 which precisely states the security of the 2-level Wa-GWIBE used in our construction of trace and revoke system, and from Theorem 3.4.8, we directly obtain the following result:

Theorem 3.4.9 If Waters’ HIBE of depth 2 is \((t, q_K, \epsilon)\) IND-CPA-secure, then the 2level-Wa-GWIBE-IDTR is \((t^*, q_K, \epsilon^*)\) IND-CPA-secure with

\[
t^* \leq (t - \log(N)(1 + q_K)) / (r \log(\frac{N}{r})), \quad q_K^* \leq q_K, \quad \epsilon^* \leq r \log(\frac{N}{r}) \times \epsilon \times \log(N)
\]

In addition, it is proved in [ACD+06] that: if there exists a \((t, q_K, \epsilon)\) adversary against Waters’ HIBE of depth \(L\), then there exists an algorithm solving the bilinear decisional Diffie-Hellman (BDDH) problem with advantage \(\epsilon_{\text{BDDH}} = O(\epsilon/\log^2(N)q_K^2)\). In our scheme, we only use 2-level Waters’ HIBE, and since \(n \leq \log(N)\), we have \(\epsilon_{\text{BDDH}} = O(\epsilon/\log^2(N)q_K^2)\).

In conclusion, the security of our scheme can be based on the hardness of the BDDH problem:
Theorem 3.4.10 If the BDDH is \((\epsilon_{\text{BDDH}})\) secure, then the 2level-Wa-GWIBE-IDTR is \((q_K^*, \epsilon^*)\) IND-CPA-secure with
\[
\epsilon^* = O\left( r \log\left( \frac{N}{r} \right) \times \epsilon_{\text{BDDH}} \times (q_K^*)^2 \times \log^3(N) \right).
\]

2level-GWIBE-IDTR resists to PEvoA and pirates 2.0 Under similar arguments for the WIBE-IDTR scheme, our construction of 2level-GWIBE-IDTR resists to PEvoA and pirates 2.0. Again, the main reason comes from the same proof that the adversary cannot derive a new decryption key at any intermediate node from a set of traitors’ secret keys, otherwise the security of the underlying scheme (here is 2level-GWIBE) could be broken, as shown in lemma 3.4.4.

3.5 Resisting Pirates 2.0 Attack in Bounded Leakage Model

In the above method (and all existed methods [DdP11, DdP13, ZZ11]), one only considers the model in which each time a traitor contributes his secret information to the public domain, he has to entirely contribute one of his sub-keys. In this contribution [PT13], we consider the general model attack of Pirates 2.0 in which each time a traitor can contribute an arbitrary number of bit secret informations of his secret key to the public domain as long as the total number of contributed bit informations of his secret key is less than a given threshold. We realize that this model closely relates to the subject of key-leakage resilient cryptography, we thus study a connection between the rich domain of key-leakage resilient cryptography and Pirates 2.0 and propose a method to fight against Pirates 2.0 in bounded leakage model. To this aim, we first formalize the key-leakage resilient security for a revoke scheme, which enhances its classical security model, we then propose a concrete construction of key-leakage resilient revoke scheme. Our construction is based on the identity-based encryption with wildcards (WIBE) [ACD+06, ABC+11] in the similar way to above contribution [PT11], it turns out that we need to construct a key-leakage resilient WIBE which is inspired from the work of [LRW11], and is achieved in successive steps:

- The security of a key-leakage resilient WIBE generalizes the full security of a WIBE by allowing the adversary to make additional leak queries. Our first step is then to construct an efficient fully secure WIBE. Fortunately, with the recent dual system encryption technique in [Wat09] and changing the distribution of exponent of \(G_{p_2}\) part in the semi-functional key, we can construct a variant of the Boneh-Boyen-Goh’s WIBE (BBG – WIBE) [ACD+06] scheme that is fully secure with a very efficient reduction that avoids a loss of an exponential factor in hierarchical depth as in the classical method of reducing the full security of WIBE to the full security of the underlying HIBE in [ACD+06].

- Inspired by the security proof technique of the key-leakage resilient HIBE in [LRW11], our second step is to transform this variant of fully secure BBG – WIBE to a secure key-leakage resilient WIBE.
CHAPTER 3. RESISTANCE TO ADVANCED ATTACKS ON BROADCAST ENCRYPTION

Fighting Pirates 2.0  We first define Pirates 2.0 attack in bounded leakage model, and show that all existed methods for fighting Pirates 2.0 [DdP11, DdP13, PT11, ZZ11] only consider a particular form of Pirates 2.0 in bounded leakage model. We then present a theoretical result in which any key-leakage resilient revoke scheme satisfying the following conditions will resist Pirates 2.0 in bounded leakage model:

- any user’s secret key is a high independent source, i.e., it has a high entropy even under the condition that all the keys of the others users are known.
- resilience to a sufficient high level of leakage at secret keys of users.

Intuitively, the first condition assures that the secret keys of users are sufficiently independent each from the others and the second condition implies that the users should contribute a high information about its key to produce an useful decoder. Combining the two conditions, the users have to contribute high information of their own independent sources and thus lose their anonymity.

Finally, we prove that our key-leakage resilient identity-based revoke scheme resists Pirates 2.0 in bounded leakage model.

3.5.1 Key-Leakage Resilient Revoke Scheme

The definition of a key-leakage resilient revoke scheme is the same as a classical revoke scheme. Formally, it consists of four polynomial-time algorithms (Setup, Extract, Encrypt, Decrypt):

Setup(\(\lambda, N_u\)): Takes as inputs the security parameter \(\lambda\) and the number of users \(N_u\). This algorithm generates a master public key \(MPK\) and a master secret key \(MSK\).

Extract(MSK, \(i\)): Takes as inputs an indices \(i\) of user and the master secret key \(MSK\), the key extraction algorithm generates a user secret key \(SK_i\).

Encrypt(MPK, \(R, M\)): The encryption algorithm which on inputs of the master public key \(MPK\), a revocation list \(R\) of revoked users in the scheme, and a message \(M\) outputs a ciphertext \(C\).

Decrypt(SK\(_i\), C): The decryption algorithm which on input of a user secret key \(SK_i\) and a ciphertext \(C\) outputs a plaintext message \(M\), or \(\perp\) to indicate a decryption error.

For correctness we require that Decrypt(SK\(_i\), Encrypt(MPK, \(R, M\))) = M with probability one for all \(i \in \mathbb{N} \setminus R\), \(M \in \{0, 1\}^*\), \((MPK, MSK) \leftarrow \text{Setup}(\lambda, N_u)\) and \(SK_i \leftarrow \text{Extract}(MSK, i)\).

Security Model

We now present the security model for a (\(\ell_{SK}\))-key-leakage resilient revoke scheme in bounded leakage model (each user leaks maximum \(\ell_{SK}\) bits on his secret key \(SK\)).
3.5. RESISTING PIRATES 2.0 ATTACK IN BOUNDED LEAKAGE MODEL

Setup: The challenger takes as inputs the security parameter $\lambda$, a maximum number of users $N_u$ and runs $\text{setup}(\lambda, N_u)$ algorithm. The master public key $\text{MPK}$ is passed to the adversary. Also, it sets the set of revoked users $\mathcal{R} = \emptyset$, $\mathcal{T} = \emptyset$, note that $\mathcal{R} \subseteq \mathcal{I}$, and $\mathcal{T} \subseteq \{\mathcal{I} \times \mathcal{SK} \times \mathcal{N}\}$ (users indices - secret key of users - leaked bits).

Phase 1: The adversary can adaptively request three types of query:

- **Create**($i$): The challenger makes a call to $\text{Extract}(\text{MSK}, i) \rightarrow \text{SK}_i$ and adds the tuple $(i, \text{SK}_i, 0)$ to the set $\mathcal{T}$ if the indices $i$ does not exists in $\mathcal{T}$.
- **Leak**($i, f$): The challenger first finds the tuple $(i, \text{SK}_i, L)$, then it checks if $L + |f(\text{SK}_i)| \leq \ell_{\text{SK}}$. If true, it responds with $f(\text{SK}_i)$ and updates the $L = L + |f(\text{SK}_i)|$. If the checks fails, it returns $\perp$ to the adversary.
- **Reveal**$(i)$: The challenger first finds the tuple $(i, \text{SK}_i, L)$, then responds with $\text{SK}_i$ and adds the indices $i$ to the set $\mathcal{R}$.

Challenge: The adversary submits two equal length messages $M_0, M_1$. The challenger picks a random bit $b \in \{0, 1\}$ and set $C = \text{Encrypt}(\text{MSK}, \mathcal{R}, M_b)$. The ciphertext $C$ is passed to the adversary.

Phase 2: This is identical to phase 1 except that the adversary is not allowed to ask **Reveal**$(i)$ query in which $i \notin \mathcal{R}$.

Guess: The adversary outputs a guess $b'$ and wins the game if $b' = b$.

**Definition 3.5.1** A revoke scheme is $(\ell_{\text{SK}})$-key-leakage resilient secure if all probabilistic polynomial-time adversaries (called PPT adversaries for short) have at most a negligible advantage in winning the above security game.

3.5.2 A Construction of Key-Leakage Resilient Revoke Scheme - KIDTR

In the IDTR scheme, we integrate a WIBE scheme into a CS scheme. KIDTR scheme is constructed in the same way as IDTR, except that in KIDTR scheme we use a key-leakage resilient WIBE scheme instead of WIBE. Therefore, the definition of KIDTR follow closely to the one in IDTR scheme.

Security Model

Setup: The challenger takes a security parameter $\lambda$, a maximum number of users in each group $N_u$ and runs $\text{setup}(\lambda, N_u)$ algorithm. The master public key $\text{MPK}$ is passed to the adversary. Also, it sets $\mathcal{R}_{ID} = \emptyset$, $\mathcal{T}_{ID} = \emptyset$, note that $\mathcal{R}_{ID} \subseteq \{\text{ID}, \text{id}\}$, and $\mathcal{T}_{ID} \subseteq \{(\text{ID}, \text{id}), \mathcal{SK}, \mathcal{N}\}$ (group’s identity and users’ identities - secret key of users - leaked bits) for all $\text{ID}$.

Phase 1: The adversary can adaptively make three types of query:
• \textbf{Create}(ID, id): The challenger makes a call to \textbf{Extract}(MSK, ID, id) \rightarrow d_{ID, id} and adds the tuple \((ID, id, d_{ID, id}, 0)\) to the set \(T_{ID}\) if this identity does not exist in \(T_{ID}\).

• \textbf{Leak}((ID, id), f) The challenger first finds the tuple \((ID, id, d_{ID, id}, L)\). It checks if \(L + |f(d_{ID, id})| \leq \ell_{SK}\). If this is true, it responds with \(f(d_{ID, id})\) and updates the \(L\) in the tuple with \(L + |f(d_{ID, id})|\). If the checks fails, it returns \(\bot\) to the adversary.

• \textbf{Reveal}(ID, id): The challenger first finds the tuple \((ID, id, d_{ID, id}, L)\). The challenger responds with \(d_{ID, id}\) and adds the identity \((ID, id)\) to the set \(R_{ID}\).

\textbf{Challenge}: The adversary submits two equal length messages \(M_0, M_1\) and an identity \(ID^*\). The challenger picks a random bit \(b \in \{0, 1\}\) and set \(C = \text{Encrypt}(\text{MSK}, ID^*, R_{ID^*}, M_b)\). The ciphertext \(C\) is passed to the adversary.

\textbf{Phase 2}: This is identical to phase 1 except that the allowed queries are \textbf{Create} queries, and only \textbf{Reveal}(ID, id) queries in which \(ID \neq ID^*\) or \(ID = ID^*\) and \(id \in R_{ID^*}\).

\textbf{Guess}: The adversary outputs a guess \(b'\) of \(b\).

\textbf{Definition 3.5.2} A KIDTR scheme is \((\ell_{SK})\)-key-leakage secure if all PPT adversaries have at most a negligible advantage in the above security game.

The rest of section is now devoted to construct a key-leakage resilient revoke scheme. The construction is achieved via the following steps:

1. we first propose a variant of BBG – WIBE scheme which is proven fully secure by using the dual system encryption technique.

2. we then construct a key-leakage resilient BBG – WIBE scheme.

3. we finally apply the generic transformation from a WIBE to an identity based trace and revoke scheme to construct a key-leakage resilient identity-based revoke scheme.

\textbf{BBG – WIBE in Composite Order Groups}

In [LW10], Lewko and Waters apply the dual system encryption technique to prove the full security of the BBG – HIBE scheme. This technique first splits the security game into \(q + 5\) games where \(q\) is the maximum number of queries that adversary makes. The first game is the real BBG – HIBE security game and the final game gives no advantage for the adversary. Second, based on the three complexity assumptions 1, 2, 3 in [LW10], step by step they prove that these games are indistinguishable, this automatically avoids a loss of an exponential factor in hierarchical depth as in the classical method. This is achieved via the main concept of the nominal semi-functionality, in which a semi-functional key is nominal with a semi-functional ciphertext if the semi-functional key can decrypt the semi-functional ciphertext. If the challenger only can create a nominal semi-functional...
ciphertext with respect to the semi-functional challenge key, then he cannot test by himself whether the challenge key is semi-functional or not because the decryption always successes.

We follow their approach by applying the dual system encryption technique to construct a fully secure variant of the BBG – WIBE scheme. The problem here is that the transformation from the BBG – HIBE to the BBG – WIBE needs to introduce additional components ($C_{3,i}$) in the ciphertext, and these components demolish the nominal property. The reason is the challenger can create a nominal semi-functional ciphertext with respect to the semi-functional challenge key, then use ($C_{3,i}$) and the components ($E_{i}$) in the challenge key to test by himself whether the challenge key is semi-functional or not. In order to retain the nominality, we should manage to impose the distribution of exponents of $G_{2}$ part in $C_{3,i}$ and in $E_{i}$ in the semi-functional key and the corresponding semi-functional ciphertext in a compatible way such that they are always nominal with each other.

We provide the details about our construction of BBG – WIBE scheme in composite order groups and the proof of its full security in our paper [PT13].

KWIBE: Key-Leakage Resilient WIBE

In the construction of key-leakage resilient HIBE in [LRW11], the user’s secret key is constructed from elements in subgroups $G_{1}$ and $G_{3}$. This leads to secret keys that are relatively low independent sources because they are only in subgroups $G_{1}$ and $G_{3}$. In order to enhance the independent sources of each user’s secret key, in our construction of KWIBE, the secret keys are in the semi-functional form and each user’s secret key is now a high independent source since the main part of the secret key is in the whole group $G = G_{1} \times G_{2} \times G_{3}$. Fortunately, this slightly change doesn’t affect the functionality and the security of the scheme.

Construction from BBG – WIBE The main point in proving the key-leakage resilience of HIBE in [LRW11] is to show that the adversary cannot distinguish between two games KeyLeak$_{0}$ and KeyLeak$_{1}$ which are briefly described as follows: in the game KeyLeak$_{b}$ game (for both $b = 0$ and $b = 1$), the adversary can choose to receive a normal key or a semi-functional key from each leak and reveal query for all keys except one key - called the challenge key. Concerning the challenge key, it is set to be a normal key in the game KeyLeak$_{0}$ and a semi-functional key in the game KeyLeak$_{1}$. We can realize that, in this technique of proving the security, there is no significant difference between a HIBE attack and a WIBE attack. Indeed, the main difference between HIBE and WIBE is that an adversary against WIBE can ask more leak queries (for keys that match the challenge pattern) than an adversary against HIBE (who can only ask for keys which are prefix of the challenge identity). However, because the difference between two games KeyLeak$_{0}$ and KeyLeak$_{1}$ is only related to the challenge key which has the same form in both HIBE and WIBE, the proof in HIBE is well adapted to WIBE.

In order to make BBG – WIBE resilient to key-leakage, in the following construction, we first impose the distribution of exponents of $G_{2}$ part in $C_{3,i}$ and in $E_{i}$ in a compatible
way such that they are nominal with each other, then we manage to choose compatibly some constants (as \( r_1, r_2, z_k, z_c \)) to keep the following properties:

- if \( \overrightarrow{\Gamma} \) is orthogonal to \( \overrightarrow{\delta} \) then the challenge key is nominally semi-functional and well-distributed.
- if \( \overrightarrow{\Gamma} \) is not orthogonal to \( \overrightarrow{\delta} \), then the challenge key is truly semi-functional and well-distributed.

The construction is detailed as follows.

**Setup** \((\lambda^1) \rightarrow (\text{MPK}, \text{MSK})\) The setup algorithm chooses a bilinear group \( G = G_1 \times G_2 \times G_3 \) of order \( N = p_1p_2p_3 \) (each subgroup \( G_i \) is of order \( p_i \)). We will assume that users are associated with vectors of identities whose components are elements of \( \mathbb{Z}_N \). If the maximum depth of the WIBE is \( D \), the setup algorithm chooses a generator \( g_1 \overset{\$}{\leftarrow} G_1 \), a generator \( g_2 \overset{\$}{\leftarrow} G_2 \), and a generator \( g_3 \overset{\$}{\leftarrow} G_3 \). It picks \( b, a_1, \ldots, a_D \overset{\$}{\leftarrow} \mathbb{Z}_N^{D+1} \) and sets \( h = g_b, u_1 = g_1^{a_1}, \ldots, u_D = g_1^{a_D} \). It also picks \( n \) random exponents \( \langle \alpha, x_1, x_2, \ldots, x_n \rangle \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+1} \). The secret key is \( \text{MSK} = (\alpha, a_1, \ldots, a_D) \), and the public parameters are:

\[
\text{MPK} = (N, g_1, g_3, h, u_1, \ldots, u_D, e(g_1, g_1)^\alpha, g_1^{x_1}, g_1^{x_2}, \ldots, g_1^{x_n})
\]

**ExtractSF** \((\text{MSK}, (ID_1, ID_2, \ldots, ID_j), g_2, \text{MPK})\) The key generation algorithm picks \( n + 1 \) random exponents \( \langle r, t_1, t_2, \ldots, t_n \rangle \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+1}, \overrightarrow{\gamma} \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+2} \) and \( z_1, \rho_{n+3}, \ldots, \rho_{n+2D-j} \overset{\$}{\leftarrow} \mathbb{Z}_N \), and \( \overrightarrow{\gamma} = (\gamma_1, \ldots, \gamma_{n+2}) \) in which \( (\gamma_1, \ldots, \gamma_n, \gamma_{n+2}) \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+1} \). \( \gamma_{n+1} = \gamma_{n+2}(z_k - \sum_{i=1}^j a_i ID_i) \). It outputs the secret key \( SK = (K', E_{j+1}, \ldots, E_D) \):

\[
\begin{align*}
&= \left\langle \left( g_1^{\alpha}, g_1^{x_1}, \ldots, g_1^{x_n}, \prod_{i=1}^D u_i^{ID_i} \right)^{-r} \prod_{i=1}^n g_1^{-x_it_i} g_1^{\gamma_i} \right\rangle \cdot g_3^\overrightarrow{\gamma} \cdot g_2^r \cdot u_{j+1}^{\rho_{n+3}}, g_2^{\gamma_{n+2}^2}, \ldots, u_D g_3^{\rho_{n+2D-j}^2}
\end{align*}
\]

Note that, to run the **ExtractSF** algorithm one doesn’t need to have \( g_2 \), he only need to have \( X_2 \in G_2 \) or \( X_2, X_3 \in G_3 \) in which \( X_2, X_3 \in G_3 \).

**Delegate** \(((ID_1, ID_2, \ldots, ID_j), \text{SK}', ID_{j+1})\) Given a secret key \( \text{SK}' = (K', E_{j+1}, \ldots, E_D) \) for identity \((ID_1, ID_2, \ldots, ID_j)\), this algorithm outputs a key for \((ID_1, ID_2, \ldots, ID_{j+1})\). It works as follow:

It picks \( n + 1 \) random exponents \( \langle r', y_1, y_2, \ldots, y_n \rangle \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+1}, \overrightarrow{\gamma'} \overset{\$}{\leftarrow} \mathbb{Z}_N^{n+2} \), and \( \rho_{n+3}, \ldots, \rho_{n+2D-j} \overset{\$}{\leftarrow} \mathbb{Z}_N \). It outputs the secret key \( SK = (K', E_{j+2}, \ldots, E_D) \):

\[
\begin{align*}
&= (K' \cdot \left\langle \left( g_1^{y_1}, g_1^{y_2}, \ldots, g_1^{y_n}, h^{-r'}(E_{j+1})^{-ID_{j+1}} \left( \prod_{i=1}^{j+1} u_i^{ID_i} \right)^{-r'} \prod_{i=1}^n g_1^{-x_iy_i} g_1^{\gamma_i} \right) \cdot g_3^\overrightarrow{\gamma'}, E_{j+2}^{r' \rho_{n+3}}, \ldots, E_D u_D g_3^{r' \rho_{n+2D-j}} \right\rangle)
\end{align*}
\]
3.5. RESISTING PIRATES 2.0 ATTACK IN BOUNDED LEAKAGE MODEL

**Encrypt**\((M, (P_1, P_2, \ldots, P_j))\) The encryption algorithm chooses \(s \triangleleft Z_N\) and outputs the ciphertext:

\[
CT = (C_0, C_1, C_2) = \\
(M \cdot e(g_1, g_1)^{\alpha_s}, \left((g_1^{x_1})^s, \ldots, (g_1^{x_n})^s, g_1^s, (h \cdot \prod_{i \in \mathbb{W}(P)} u_i^{P_i})^s\right), (C_{2,i} = u_i^s)_{i \in \mathbb{W}(P)})
\]

**Decrypt**\((CT, SK)\) Any other receiver with identity \(ID = (ID_1, ID_2, \ldots, ID_j)\) matching the pattern \(P\) to which the ciphertext was created can decrypt the ciphertext \(CT = (C_0, C_1, C_2)\) as follows

First, he recovers the message by computing

\[
\overrightarrow{C_1} = \left((g_1^{x_1})^s, \ldots, (g_1^{x_n})^s, g_1^s, (h \cdot \prod_{i \in \mathbb{W}(P)} u_i^{P_i})^s \cdot \prod_{i \in \mathbb{W}(P)} (u_i^s)^{ID_i}\right)
\]

Finally, compute

\[
e_{n+2}(\overrightarrow{K}, \overrightarrow{C_1}) = e(g_1, g_1)^{\alpha_s} \cdot e(g_1, u_1^{ID_1} \cdots u_j^{ID_j} h)^{-rs} \cdot e(g_1, u_1^{ID_1} \cdots u_j^{ID_j} h)^{rs} \cdot \prod_{i=1}^{n} e(g_1, g_1)^{x_i t_i s} \cdot \prod_{i=1}^{n} e(g_1, g_1)^{x_i t_i s} = e(g_1, g_1)^{\alpha s}
\]

Notice that the \(G_2\) and \(G_3\) parts do not contribute because they are orthogonal to the ciphertext under \(e\).

**Security of Key-Leakage Resilient BBG – WIBE** Formally, the security model of a \(\ell_{SK}\)-key-leakage resilient WIBE, we call **Leak – WIBE** security game, is defined as follows:

We let \(I^*\) denote the set of all possible identity vectors, \(R\) denote the set of all revealed identities

**Setup**: The challenger makes a call to **Setup**\((1^\lambda)\) and gets the master secret key \(MSK\) and the public parameters \(MPK\). It gives \(MPK\) to the attacker. Also, it sets \(R = \emptyset\) and \(\mathcal{T} = \emptyset\), note that \(\mathcal{R} \subseteq I^*, \mathcal{T} \subseteq \{I^*, SK, N\}\) (identity vectors - secret keys - leaked bits).

**Phase 1**: The adversary can adaptively make three types of query:

- **Create**\(\left(\overrightarrow{T}\right)\): The challenger makes a call to **ExtractSF** to generate \(SK_I\) and adds the tuple \((\overrightarrow{T}, SK_I, 0)\) to the set \(\mathcal{T}\) if this identity does not exist.
- **Leak**\((\overrightarrow{T}, f)\): The challenger first finds the tuple \((\overrightarrow{T}, SK_I, L)\), then it checks if \(L+ | f(SK_I) | \leq \ell_{SK}\). If true, it responds with \(f(SK_I)\) and updates the \(L = L+ | f(SK_I) |\). If the checks fails, it returns \(\perp\) to the adversary.
- **Reveal**\(\left(\overrightarrow{T}\right)\): The challenger first finds the tuple \((\overrightarrow{T}, SK_I, L)\), then responds with \(SK_I\) and adds the identity vector \(\overrightarrow{T}\) to the set \(R\).
CHAPTER 3. RESISTANCE TO ADVANCED ATTACKS ON BROADCAST ENCRYPTION

Challenge: The adversary submits a challenge pattern $\overrightarrow{P}^{*}$ with the restriction that no identity vector in $\mathcal{R}$ matches $\overrightarrow{P}^{*}$. It also submits two messages $M_0, M_1$ of equal size. The challenger flips a uniform coin $c \leftarrow \{0, 1\}$ and encrypts $M_c$ under $\overrightarrow{P}^{*}$ with a call to $\text{Encrypt}(M_c, \overrightarrow{P}^{*})$. It sends the resulting ciphertext $CT^{*}$ to the adversary.

Phase 2: This is the same as Phase 1, except the only allowed queries are $\text{Create}$ queries for all identity vector, and $\text{Reveal}$ queries for secret keys with identity vectors which do not matches $\overrightarrow{P}^{*}$.

Guess: The adversary outputs a bit $c' \leftarrow \{0, 1\}$. We say it succeeds if $c' = c$.

Definition 3.5.3 A $\text{KWIBE}$ scheme is $(\ell_{SK})$-key-leakage secure if all PPT adversaries have at most a negligible advantage in the above security game.

Theorem 3.5.4 (Security of Key-Leakage Resilient BBG − WIBE) Under assumptions 1, 2, 3 in [LW10] and for $\ell_{SK} = (n - 1 - 2c) \log(p_2)$, where $c > 0$ is any fixed positive constant, our key-leakage resilient $\text{BBG} − \text{WIBE}$ scheme is $(\ell_{SK})$ - key-leakage secure.

The condition for $c$ is $p_2^{-c}$ is negligible. The length of secret key $sk$ at level $i$ is $(n + 2 + D - i)(\log(p_1) + \log(p_2) + \log(p_3))$ where $D$ is the depth of $\text{WIBE}$. As we can see, the leakage fraction of secret key at leaf node is the biggest. The proof of this theorem can be found in our paper [PT13].

Generic Construction of KIDTR

The construction of KIDTR closely follows to the construction of $\text{WIBE-IDTR}$, except that we use the new primitive $\text{KWIBE}$ instead of $\text{WIBE}$ for encryption. We integrate $\text{KWIBE}$ into the complete subtree method: each group $ID \in \{0, 1\}^r$ represents a binary tree and each user $id \in \{0, 1\}^l \ (id = id_1 id_2 \cdots id_l, \ id_i \in \{0, 1\} )$ in a group $ID$ is assigned to be a leaf of the binary tree rooted at $ID$. For encryption, we will use a $\text{KWIBE}$ of depth $l + 1$, each user is associated with a vector $(ID, id_1, \cdots, id_l)$.

Security of KIDTR

Theorem 3.5.5 (Security of KIDTR) If the $\text{KWIBE}$ is $(\ell_{SK})$ - key-leakage secure then our KIDTR is also $(\ell_{SK})$ - key-leakage secure.

The sketch of the proof: Our proof follows closely to the proof of theorem 2 in [PT11]. We also organize our proof as a sequence of games. The first game Game 0 defined will be the real KIDTR game and the last one will be one in which the adversary has no advantage unconditionally. We then show that each game is indistinguishable from the next, under the assumptions of the security of $\text{KWIBE}$.

95
3.5.3 KIDTR is Immune to Pirates 2.0 in Bounded Leakage Model

Pirates 2.0 in Bounded Leakage Model

The basic idea behind Pirates 2.0 attack is that traitors are free to contribute some piece of secret key as long as several users of the system could have contributed exactly the same information following the same (public) strategy: this way, they are able to remain somewhat anonymous. The leakage information is formalized via extraction function which is any efficiently computable function $f$ on the space of the secret keys and a traitor $u$ is said to be masked by a user $u'$ for an extraction function $f$ if $f(sk_u) = f(sk_{u'})$. The anonymity level is meant to measure exactly how anonymous they remain. This is defined in [BP09] as follows.

**Definition 3.5.6 (Anonymity Level)** The level of anonymity of a traitor $u$ after a contribution $\cup_1^{\leq t} f_i(sk_u)$ is defined as the number $\alpha$ of users masking $u'$ for each of the $t$ extraction functions $f_i$ simultaneously:

$$\alpha = \#\{u' \mid \forall i, f_i(sk_u) = f_i(sk_{u'})\}.$$  

**Definition 3.5.7 (Pirates 2.0 in Bounded Leakage Model)** We say that a Pirates 2.0 attack is in bounded leakage model if for every traitor $u$ with his secret key $sk_u$, at each time $i$ ($i = 1, \ldots, t$), is free to choose any strategy $f_i$ to contribute the bits information of $sk_u$ to the public domain as long as

$$\sum_{i=1}^{t} | f_i(sk_u) | \leq \ell_{SK}$$

where $\ell_{SK}$ is the threshold.

Comparison to Other Methods

Until now, there have been many methods aiming to fight against Pirates 2.0 [DdP11, DdP13, PT11, ZZ11] but all of them only consider a particular form of leakage of secret key in bounded leakage model. In fact, it is assumed in these methods that by using a public extraction function the dishonest users leak the entire information of some sub-keys which could be used in the encryption procedure. Concretely, in [DdP13] they describe the extraction function as a projection function, and the secret key of user $SK$ is a vector $(SK_1, \ldots, SK_l)$ of elements, where each $SK_i, i = 1, \ldots, l, k$ contains $k$ bits information. The output of extraction function, at each time, is an $i$–th element of the vector. This require that a traitor, at each time, must contribute at least $k$ bits information of his secret key where $k$ must be bigger than the security parameter of the scheme. This kind of attack is thus a particular case of Pirates 2.0 in bounded leakage model in which the strategy of the traitor is limited: at each time each traitor has to choose a whole sub-key to contribute.

We consider the general form of Pirates 2.0 attack in bounded leakage model, by considering any strategy of the adversary. This generalizes thus all the previous consideration of Pirates 2.0. However, there is still a gap between the Pirates 2.0 attack in bounded
leakage model and the general form of the Pirates 2.0 attack where the pirate can combine the information of the bits of the secret key and then contribute a particular form of information. This kind of attack could be captured by considering a general form of leakage for revoke schemes and this seems a very challenging problem.

Pirates 2.0 in Bounded Leakage Model Viewed from the Information Theory

We aim to re-explain the way Pirates 2.0 in bounded leakage model works under the information theory. This is also the basic starting point so that we can establish a sufficient condition for a scheme to resist Pirates 2.0 in bounded leakage model in the next subsection. In a revoke scheme, when a user joins the system, its key is generated and has some entropy. However, as keys of users could be correlated, the user can contribute some correlated information without the risk being identified. The user really lose its anonymity when he contributes its independent secret information that the other users don’t have. More formally, these are entropy conditioned on the information about the other users’ keys. Let us first recall some classical definitions about entropy.

**Definition 3.5.8** Let $X$ be a random variable. The min-entropy of $X$ is

$$H_\infty(X) = \min_x - \log(\Pr[X = x]) = - \log(\max_x \Pr[X = x])$$

We say that $X$ is a $k$-source if $H_\infty(X) \geq k$.

The high min-entropy is used rather than the Shannon entropy in cryptography for describing good distributions for the keys. In fact, the conventional notion in cryptography is the intuitive notion of “guess ability” and a distribution $X$ has min-entropy of $k$ bits if even an unbounded adversary cannot guess a sample from $X$ with probability greater than $2^{-k}$.

However, in context of Pirates 2.0 in bounded leakage model, a high min-entropy is not enough because the keys could be correlated. We should thus need to define how many information of the key a user has that is independent to the keys of the others users. This is quantified via the conditional min-entropy.

**Definition 3.5.9** Let $X, E$ be a joint distribution. Then we define the min-entropy of $X$ conditioned on $E$—denoted $H_\infty(X|E)$—as

$$H_\infty(X|E) = - \log \max_e [\max_x \Pr[(X|E = e)]]$$

We say that $X$ is a $k$-independent source of $E$ if $H_\infty(X|E) \geq k$.

We note that Dodis et. al. [DRS04] defines the conditional min-entropy as average entropy $\log E[\max_x \Pr [(X|E = e)]]$. In our setting, we follow a conservative approach, taken in [RW05], and manage to deal with the above stronger notion. In fact, we will see later in our construction that the secret keys of users are sufficiently independent each from the others, the consideration of the conditional min-entropy can be justified. We first define the independence between the secret keys in a revoke system as follows.
Figure 3.8: An example of a complete subtree scheme where the center covers all non-revoked users with the nodes $S_1, \ldots, S_6$. A user is a leaf on the binary tree where each node is assigned to a long-lived randomly chosen key. Each user possesses all the long-lived keys of the nodes on the path from the user’s leaf to the root.

**Definition 3.5.10 (Independent Source)** In a revoke system of $N_u$ users, let $X_i$ be the distribution outputted by the key generation for a user $i$ and let $E = (X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_{N_u}, \text{pub})$ where $\text{pub}$ denotes the distribution of the public parameters in the system. Then we say that the key of user $i$ is a $k$-independent source if $H_\infty(X_i|E) \geq k$.

The key of user $i$ is a $k$-independent source if it has $k$-bit entropy independently from the keys of the others users and from all the public information of the systems.

We now review the Pirates 2.0 in bounded leakage model in the context of Complete Subtree resumed in Figure 3.5.3. For a $D$-level tree, each user’s key is a $(D \times \lambda)$-source but only a $\lambda$-independent source because each user only has an independent sub-key at the leaf. Therefore, even if a user contributes $((D - 1) \times \lambda)$ entropy of its key, the remained information could still be a $\lambda$-independent source. Without leaking any independent entropy, the user could remain anonymous at a level $\alpha > 1$ (because at least two different users can have the same contributive information). In the example in Figure 3.5.3, the user $U$ is assigned 5 sub-keys corresponding to the nodes from the root to the leaf. The user $U$ can contribute a key $S_4$ and specifies the target set at $S_4$ that covers 4 users of the sub-tree rooted at $S_4$. A pirate decoder with only one key at $S_4$ can decrypt the ciphertext for the chosen target set $S_4$ with non-negligible probability while preserving an anonymity level $\alpha = 4$ for the contributor and therefore, the scheme is vulnerable against the Pirates 2.0 in bounded leakage model.

**Key-Leakage Resilience vs. Pirates 2.0 in Bounded Leakage Model**

We are now ready to prove a sufficient condition so that a key-leakage resilient revoke scheme is immune to Pirates 2.0 attack in bounded leakage model. We first use the following lemma

**Lemma 3.5.11** For any function $f, g$, and any random variable $X, Y$, if $H_\infty(X|Y) \geq k$ and $H_\infty(X|f(X), Y) \leq k - \alpha$ then

$$\Pr_{x \in X, y \in Y}[f(x) = g(y)] \leq \frac{1}{2^\alpha}.$$
CHAPTER 3. RESISTANCE TO ADVANCED ATTACKS ON BROADCAST ENCRYPTION

Proof: Assume there exists \( g \) such that \( \Pr_{x \in X, y \in Y}[f(x) = g(y)] > \frac{1}{2^n} \), then there exists \( z = g(y) \) such that: \( \Pr_{x \in X}[f(x) = z] > \frac{1}{2^n} \). On the other hand,

\[
H_\infty(X_i|Y) \geq k \iff \max_x \max_y \Pr[X = x|Y = y] \leq \frac{1}{2^k}
\]

We now compute:

\[
H_\infty(X_i|f(X), Y) = -\log(\max_{z,y} \Pr[X = x|f(X) = z, Y = y])
\]

\[
= -\log(\max_{z,y} \frac{\Pr[(X = x|Y = y) \wedge f(X) = z]}{\Pr[f(X) = z]}) \geq -\log(\max_{z,y} \frac{\Pr[(X = x|Y = y)]}{\Pr[f(X) = z]})
\]

\[
> -\log(\frac{1}{2^n}) = k - \alpha
\]

This is contradict with the assumption.

The following theorem gives a condition on the independence of the user’s key under which we can relate the leakage resilience to the Pirates 2.0.

Theorem 3.5.12 Let \( \Pi \) be a \( (\ell_{SK}) \)-key-leakage resilient revoke system of \( N_u \) users in which each user’s key has length of \( m \) bit and is a \( m' \)-independent source. If \( \alpha = \frac{N_u}{2^{\ell_{SK} + m' - m}} \leq 1 \), then \( \Pi \) is immune to any Pirates 2.0 attack in bounded leakage model.

Proof:

Proposition 3.5.13 In a Pirates 2.0 attack in bounded leakage model, if a user leaks \( k \) bits of his secret key to the public domain then his anonymity level is at most \( \frac{N_u}{2k + m' - m} \).

Proof: Intuitively, as the key of the user \( u \) is a high independent source even when the others users contribute their whole secret keys, if \( u \) leaks too much information on its key then it will also leak many independent information and loses its anonymity.

Formally, following the definition 3.5.6 of anonymity level in pirates 2.0, assume that a user \( u \) contributes \( k \) bits information \( L_u \) of his secret key \( sk_u \) to the public domain, we need to compute the probability for an user \( u' \) to contribute exactly the same information as the user \( u \), at each period of time \( i \).

- At time 0: \( u \) contributes nothing to the public domain. Let \( E_i = (\cup_{w \neq u} sk_w, \text{pub}_i) \) where \( \text{pub}_i \) denotes the public information at the time \( i \) which contains the publics parameters of the system plus contributed information of the users after the time \( i - 1 \). Because each user’s key is a \( m' \)-independent source: \( H_\infty(sk_u|E_0) \geq m' \).

- At time \( i \): \( u \) contributes his secret informations \( L_u^i = f_i(sk_u, \text{pub}_i) \) to the public domain by leaking \( k_i \) bits of his secret keys. If we denote \( k_i^{in} \) the number of independent bits that the user \( u \) losses in time \( i \), i.e., \( k_i^{in} = H_\infty(sk_u|E_i) - H_\infty(sk_u|E_{i-1}) \). From the lemma 3.5.11, the probability that \( u' \) could contribute exactly the same information \( L_u^i \) is at most \( \frac{1}{2^{k_i^{in}}} \). Note that \( E_0 \) and thus \( E_i \) already contain \( \cup_{w \neq u} sk_w \), i.e., all the contributed information of the other users are already contained in \( E_i \) (for all \( i \)), the \( k_i^{in} \) independent bits are among \( k_i \) bit that the user \( u \) leaks at the time \( i \).
At the end, after the time \( t \), the user \( u \) contributes to the public domain by totally leaking \( k = k_1 + \cdots + k_t \) bits of its secret information. By the above computation, the probability that an user \( u' \) can contribute exactly the same total information like \( u \) is at most \( \prod_{j=1}^{t} \frac{1}{2^{k_j}} \), and

\[
\sum_{j=1}^{t} k_j^\text{in} = H_{\infty}(sk_u|E_0) - H_{\infty}(sk_u|E_t)
\]

Because the bit length of the secret key \( sk_u \) is \( m \) and the user \( u \) leaks \( k \) bits, we deduce that \( H_{\infty}(sk_u|E_t) \leq m - k \) and therefore \( \sum_{j=1}^{t} k_j^\text{in} \geq m' - (m - k) = k + m' - m \) which implies that the probability that an user \( u' \) can contribute exactly the same information like \( u \) as required in Pirates 2.0 is at most \( \frac{1}{2^{k + m' - m}} \) and the anonymity level of \( u \) cannot be assured to be higher than \( \frac{N_2}{2^{k+m'-m}} \).

**Proposition 3.5.14** Let \( \Pi \) be a \((\ell_{SK})\)-key-leakage resilient revoke scheme. If each user leaks no more than \( \ell_{SK} \) bits of his secret key to the public domain, then one cannot produce a Pirates 2.0 decoder in bounded leakage model.

**Proof:** We suppose by contradiction that there is a Pirates 2.0 \( A \) in bounded leakage model against \( \Pi \) in which each user leaks no more than \( \ell_{SK} \) bits of his secret key to the public domain, then we build an algorithm \( B \) that breaks the security of \( \Pi \) in the context of key leakage resilience.

Algorithm \( B \) simulates \( A \) and makes use of the outputs of \( A \) to break the security of \( \Pi \). It works as follows:

- At time 0: users contribute nothing to the public domain.
- At time 1: suppose that a user \( u \) decides to contribute \( L_1^u = f_1(sk_u) \) bits to the public domain by using a strategy \( f_1 \) where \( f_1 \) is a polynomial-time computable function, \( B \) requests the leak query \((u,g_1 := f_1)\) to his challenger and forwards the result to \( A \).
- At any time \( i \): suppose that a user \( u \) decides to contribute \( L_i^u = f_i(sk_u,I) \) bits to the public domain, where \( I \) is the public collected information after the time \( i - 1 \). At this stage, \( B \) defines a polynomial-time computable function \( g_i,I(sk_u) := f_i(sk_u,I) \), then requests the leak query \((u,g_i,I)\) to his challenger and forwards the result to \( A \).
- When \( A \) outputs a pirate decoder and a target \( S \) so that the pirate decoder can decrypt ciphertexts for \( S \) with a non-negligible probability, \( B \) simply outputs \( S^* = S \) and two different messages \( M_0, M_1 \) to his challenger. By using this pirate decoder, \( B \) can decrypt the challenge ciphertext with a non-negligible probability and thus break the security of the scheme.

We note that, since each user contributes maximum \( \ell_{SK} \) bits to the public domain, \( B \) only need to ask in total at most \( \ell_{SK} \) bits to his challenger. By definition, \( \Pi \) is then not \( \ell_{SK}\)-key leakage resilient.

The theorem immediately follows from the above two propositions.
**Proposition 3.5.15** In KIDTR scheme, if we call $p_1, p_2, p_3$ are primes of $\lambda_1, \lambda_2, \lambda_3$ bits, then each user’s secret key with length $m = (n+2)(\lambda_1 + \lambda_2 + \lambda_3)$ is $m'$-independent source where $m' = ((n+1)(\lambda_1 + \lambda_2 + \lambda_3) + \lambda_2 + \lambda_3)$.

**Proof:** In our KIDTR scheme, we make use of a KWIBE scheme in which each user’s secret key is at leaf node 3.5.2, therefore an user’s secret key is of the following form:

$$SK = K_1 = \left( g_1^{t_1}, g_1^{t_2}, \ldots, g_1^n, g_1^0 \left( h \cdot \prod_{i=1}^j u_i^{ID_i} \right) \prod_{i=1}^n g_1^{-x_i t_i}, g_1^r \right)$$

where $r, t_1, t_2, \ldots, t_n, z_k \xleftarrow{} Z_N, \xleftarrow{} Z_{N+2}^n$, and $\gamma = (\gamma_1, \ldots, \gamma_{n+2})$ in which $(\gamma_1, \ldots, \gamma_n, \gamma_{n+2}) \xleftarrow{} Z_{N+1}^n$, $\gamma_{n+1} = \gamma_{n+2}(z_k - \sum_{i=1}^j a_i ID_i)$.

We realize that in each secret key, the elements corresponding to the indices 1, $\ldots$, $n, n+2$ are randomly generated in the whole group $G = G_1 \times G_2 \times G_3$, the element corresponding to the indice $n+1$ is not independent in $G_1$ but randomly generated in $G_2 \times G_3$. Therefore, it’s easy to see that each user’s secret key is of $(n+2)(\lambda_1 + \lambda_2 + \lambda_3)$ bit length and is a $((n+1)(\lambda_1 + \lambda_2 + \lambda_3) + \lambda_2 + \lambda_3)$-independent source.

**Theorem 3.5.16** The KIDTR scheme is immune to Pirates 2.0 attack in bounded leakage model for any choice of parameters $n, c, \lambda_1, \lambda_2$ such that $2^{(n-1-2c)\lambda_2-\lambda_1} > N_u$, where $N_u$ is the number of subscribed users in the systems.

**Proof:** From the theorems 3.5.4 and theorem 3.5.5, we decude that the KIDTR scheme is $\ell_{SK}$-leakage resilient with $\ell_{SK} = (n - 1 - 2c)\lambda_2$ for any fixed positive constant $c > 0$ (such that $p_2^{-c}$ is negligible). From the theorem 3.5.12, one cannot mount a Pirates 2.0 attack with an anonymity level larger than $\alpha = \frac{N_u}{2^{\ell_{SK}+m'-m}} = \frac{N_u}{2^{(n-1-2c)\lambda_2-\lambda_1}} < 1$.

We note that there is no need to choose particular parameters for our system. For example, simply with $c = 1, n = 5$ and $\lambda_1 = \lambda_2 = 512$ ($p_2^{-c} = 2^{-512}$ is negligible) and suppose that there are $N_u = 2^{40}$ subscribed users, our system is immune to Pirates 2.0 in bounded leakage model because $2^{(n-1-2c)\lambda_2-\lambda_1} = 2^{512} > N_u$ and the user’s secret key contains only 7 elements in $G$. 

101
3.5. RESISTING PIRATES 2.0 ATTACK IN BOUNDED LEAKAGE MODEL
Chapter 4

Multi-Channel Broadcast Encryption

Contents

4.1 Introduction ................................................................. 104
4.2 Definition and Security Notions of Multi-Channel Broadcast Encryption ................................................. 106
  4.2.1 Syntax ................................................................. 106
  4.2.2 Security Model ...................................................... 106
4.3 Constructions ................................................................. 108
  4.3.1 BGW Scheme ......................................................... 108
  4.3.2 Intuition ............................................................... 109
  4.3.3 Multi-Channel Broadcast Encryption I – MCBE$_1$ ......................................................... 109
  4.3.4 Multi-Channel Broadcast Encryption II – MCBE$_2$ ....................................................... 113

Broadcast encryption aims at sending a content to a large arbitrary group of users at once. Currently, the most efficient schemes provide constant-size headers, that encapsulate ephemeral session keys under which the payload is encrypted. However, in practice, and namely for pay-TV, providers have to send various contents to different groups of users. Headers are thus specific to each group, one for each channel: as a consequence, the global overhead is linear in the number of channels. Furthermore, when one wants to zap to and watch another channel, one has to get the new header and decrypt it to learn the new session key: either the headers are sent quite frequently or one has to store all the headers, even if one watches one channel only. Otherwise, the zapping time becomes unacceptably long.

In this contribution [PPT13] we deals with encapsulation of several ephemeral keys, for various groups and thus various channels, in one header only, and we call this new line of research is general broadcast encryption and the new construction for it is Multi-Channel Broadcast Encryption – MCBE: one can hope for a much shorter global overhead and a much shorter zapping time since the decoder already has the information to decrypt any available channel at once. Our candidates are private variants of the BGW scheme, with a constant-size global header, independently of the number of channels.
4.1 Introduction

Broadcast encryption has been widely and deeply studied as it is a core primitive for many concrete applications. In the following, we focus on the pay-TV scenario, in which users own decoders to decode only the channels they subscribed to. In this context, the broadcaster sends several channels at the same time, to different groups of users or target sets.

Unfortunately, previous broadcast encryption models only dealt with one single content and one single target set at a time. This was a first reasonable goal to get such an efficient broadcast encryption scheme, but not quite relevant to practice. In fact, TV systems contain many channels, with different sets of privileged users. One could argue that this scenario is covered by the usual systems, applying independent broadcast encryption schemes for each channel. However, even with a constant-size ciphertext (header) broadcast encryption, this results in a very inefficient scheme: the bandwidth, or header size, linearly grows in the number of channels, which could be very large. Of course, one header is enough to decrypt one channel, but in case of zapping from one channel to another channel, one has to start from scratch, and namely to wait for the reception of the new appropriate header, which can take some time, unless the decoder stores all the headers all the time.

These bandwidth and zapping-time problems lead to new efficiency criteria, with a common solution: a broadcast encryption with a short global header for multiple channels. The problem of optimizing the bandwidth already appeared in the context of classical (one-channel) broadcast encryption: a broadcast encryption can trivially be constructed from any encryption scheme, by encrypting the session key under each user’s key. But this induces a cost that is linear in the number of users. It took more than a decade from the introduction of the primitive [FN94] to come up with an optimal solution: without considering the description of the target set, the header is of constant size [BGW05]. This BGW solution [BGW05] is particularly interesting even if it is still not practical, due to the high decryption complexity: the latter is indeed linear in the size of the target set.

Our new primitive MCBE, for Multi-Channel Broadcast Encryption, addresses the bandwidth and zapping-time problems. In the following, we show that it is possible to solve these problems in an optimal way: a constant-size global header, independently of the number of channels (and of the users too). Unfortunately, this is still an asymptotic result, with room for improvement in practice. Actually, our solutions suffer from the same weakness as the above BGW scheme: the decryption has to take into account all the public keys of the users involved in all the target sets. It is thus not quite efficient. However, it seems that this problem is unavoidable when one compacts the information for all the targeted users into one constant-size ciphertext. We also notice that there is a simple and similar trade-off between the ciphertext size and the decryption time as in [BGW05] by partitioning the set of channels into different subsets and then encrypting to each of these subsets. The union of the target sets in a ciphertext is smaller, but there are more ciphertexts. Our objective is therefore to show that the bandwidth and zapping-time problems in the multi-channel setting can be improved from trivial techniques, as BGW did in the one-channel setting.

Finally, we emphasize that the solution requires some new techniques that we will
develop in the section 4.1. In particular, we have to deal with the problem of encapsulating different and independently-looking session keys for the different channels into one constant-size element only. We will then prove the security in the new multi-channel setting.

Our results. We first propose a formalization of the problem, with the so-called Multi-Channel Broadcast Encryption – MCBE. Because of some constraints between the various target sets, we introduce the dummy-helper technique that helps to prove the security. We eventually propose two constructions, derived from the BGW scheme. They are private broadcast encryption schemes, with the following properties:

- The first construction is, asymptotically, very competitive with the BGW scheme. In fact, it achieves the constant-size header, independently of the number of channels, while the private decryption key size remains linear in the number of the channels that a user has subscribed to. In addition, it is fully-collusion resistant against basic selective adversaries, i.e. adversaries who can only ask corruption queries to get the decryption keys of users in the selective security model (the challenge target set is announced before having seen the global parameters). This is also the security level that the original BGW scheme achieves and our security proof holds under the standard assumption \( n - BDHE \), as in the original BGW scheme [BGW05].

- The second construction improves on the previous one, to resist to strong selective adversaries who have the power of basic selective adversaries plus unlimited access to encryption and decryption queries, while keeping the parameter sizes and computational assumptions unchanged. To this aim, we introduce the dummy-helper technique and make use of a random oracle [BR93]. Our scheme is more efficient than the CCA version of the BGW scheme [BGW05] but our dummy-helper technique actually requires the random oracle model.

Dummy-helper technique. In the multi-channel setting, since the session keys of all channels are compacted in only one ciphertext, even if they have to look independent for adversaries, there exists an implicit relation between them, which could be known by the simulator without the whole knowledge of the master key. The dummy-helper technique consists in adding a new channel for one additional dummy user. We then get the following interesting properties:

1. For the security analysis: it gives our simulator the possibility to decrypt this channel and get the corresponding session key. This is then sufficient for the simulator to derive the other session keys and successfully answer any decryption query, if the simulator knows the above implicit relation between the encapsulated keys;

2. In practice: by eventually publishing the decryption key of the dummy user, it introduces a channel that can be decoded by all the users registered in the system. It can then be used to send them the program or ads.

We implement this dummy-helper technique in the random oracle model. It is worth noting that, though working in a more complex setting of multi-channel broadcast encryption, the security is achieved under the standard assumption \( n - BDHE \) as in the BGW scheme.
4.2 Definition and Security Notions of Multi-Channel Broadcast Encryption

4.2.1 Syntax

In this section we describe the model for a multi-channel broadcast encryption system. Formally, such a system consists of four probabilistic algorithms:

\textbf{Setup}(\lambda):\text{ Takes as input the security parameter } \lambda, \text{ it generates the global parameters } \text{param of the system, including } n \text{ the maximal number of users (receivers are implicitly represented by integers in } \{1, \ldots, n\} \text{), and returns a master key } \text{MSK} \text{ and an encryption key } \text{EK}. \text{ If the scheme allows public encryption, } \text{EK} \text{ is public, otherwise } \text{EK} \text{ is kept private, and can be seen as part of } \text{MSK}.

\textbf{Extract}(i, \text{MSK}):\text{ Takes as input the user's index } i, \text{ together with the master key, and outputs the user's private key } d_i.

\textbf{Encrypt}(S_1, S_2, \ldots, S_m, \text{EK}):\text{ Takes as input } m \text{ subsets (or target sets) } S_1, S_2, \ldots, S_m \text{ where, for } i = 1, \ldots, m, S_i \subseteq \{1, \ldots, n\}, \text{ and the encryption key } \text{EK}. \text{ It outputs } (\text{Hdr}, K_1, K_2, \ldots, K_m) \text{ where } \text{Hdr} \text{ encapsulates the ephemeral keys } (K_i)_{i=1,\ldots,m} \in \mathcal{K}. \text{ The key } K_i \text{ will be associated to the subset } S_i. \text{ We will refer to } \text{Hdr} \text{ as the broadcast ciphertext, or } \text{header}, \text{ whereas this header together with the description of all the target sets is called the } \text{full header}.

\textbf{Decrypt}(S_1, S_2, \ldots, S_m, \text{Hdr}, j, d_j, i):\text{ Takes as input a full header } (S_1, S_2, \ldots, S_m, \text{Hdr}), \text{ a user } j \in \{1, \ldots, n\} \text{ and its private key } d_j, \text{ together with a subgroup index } i \in \{1, \ldots, m\}. \text{ If } j \in S_i, \text{ then the algorithm outputs the ephemeral key } K_i \in \mathcal{K}.

For correctness, we require that for all } S_i \subseteq \{1, \ldots, n\}, j \in S_i, \text{ if } (\text{EK}, \text{MSK}) \leftarrow \text{Setup}(\lambda), d_j \leftarrow \text{Extract}(j, \text{MSK}) \text{ and } (\text{Hdr}, K_1, \ldots, K_m) \leftarrow \text{Encrypt}(S_1, S_2, \ldots, S_m, \text{EK}), \text{ one then should get } K_i = \text{Decrypt}(S_1, S_2, \ldots, S_m, \text{Hdr}, j, d_j, i).

In practice, the goal of such ephemeral keys is to encrypt the payload, which consists of } m \text{ messages } M_1, \ldots, M_m \text{ to be broadcast to the sets } S_1, \ldots, S_m \text{ respectively. They will thus be encrypted under the symmetric keys } K_1, \ldots, K_m \text{ into the ciphertexts } C_{M_1}, \ldots, C_{M_m} \text{ respectively. The overall data the broadcaster sends consists of } (S_1, S_2, \ldots, S_m, \text{Hdr}, C_{M_1}, C_{M_2}, \ldots, C_{M_m}) \text{ where } (S_1, S_2, \ldots, S_m, \text{Hdr}) \text{ is the } \text{full header} \text{ and } (C_{M_1}, C_{M_2}, \ldots, C_{M_m}) \text{ is often called the } \text{encrypted payload}.

4.2.2 Security Model

We define the security of a multi-channel broadcast encryption system by the following game between an attacker } A \text{ and a challenger, in the Real-or-Random setting:

\textbf{Setup}. The challenger runs the } \text{Setup} \text{ algorithm to generate the global parameters } \text{param of the system, and returns a master key } \text{MSK} \text{ and an encryption key } \text{EK}. \text{ If the
scheme is asymmetric, $E_K$ is given to $A$, otherwise it is part of the MSK, and thus kept secret. Corruption and decryption lists $\Lambda_C, \Lambda_D$ are set to empty lists.

**Query phase 1.** The adversary $A$ adaptively asks queries:

1. Corruption query for the $i$-th user: the challenger runs $\text{Extract}(i, \text{MSK})$ and forwards the resulting private key to the adversary. The user $i$ is appended to the corruption list $\Lambda_C$;

2. Decryption query on the full header $(S_1, \ldots, S_m, Hdr)$ with $u \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$. The challenger answers with $\text{Decrypt}(S_1, \ldots, S_m, Hdr, u, d_u, j)$. The pair $(Hdr, S_j)$ is appended to the decryption list $\Lambda_D$;

3. Encryption query (if $E_K$ is private) for the target sets $(S_1, S_2, \ldots, S_m)$. The challenger answers with $\text{Encrypt}(S_1, S_2, \ldots, S_m, E_K)$.

**Challenge.** The adversary $A$ outputs $t$ target sets $S_1^*, \ldots, S_t^* \subseteq \{1, \ldots, n\}$ and an index $j$, which specifies the attacked target set $S_j^*$.

The challenger runs $\text{Encrypt}(S_1^*, S_2^*, \ldots, S_t^*, E_K)$ and gets $(\text{Hdr}^*, K_1^*, K_2^*, \ldots, K_t^*)$. Next, the challenger picks a random $b \leftarrow \{0, 1\}$. If $b = 1$, it picks a random $K_j^* \leftarrow \mathcal{K}$. It outputs $(\text{Hdr}^*, K_1^*, \ldots, K_t^*)$ to $A$.

Note that if $b = 0$, $K_j^*$ is the real key, encapsulated in $\text{Hdr}^*$, and if $b = 1$, $K_j^*$ is random, independent of the header.

**Query phase 2.** The adversary $A$ continues to adaptively ask queries as in the first phase.

**Guess.** The adversary $A$ eventually outputs its guess $b' \in \{0, 1\}$ for $b$.

We say the adversary wins the game if $b' = b$, but only if $S_j^* \cap \Lambda_C = \emptyset$ and $(\text{Hdr}^*, S_j^*) \notin \Lambda_D$. We then denote by $\text{Succ}^{\text{ind}}(A) = \Pr[b' = b]$ the probability that $A$ wins the game, and its advantage is

$$\text{Adv}^{\text{ind}}(A) = 2 \times \text{Succ}^{\text{ind}}(A) - 1 = \Pr[1 \leftarrow A|b = 1] - \Pr[1 \leftarrow A|b = 0].$$

**Definition 4.2.1 (Full Security)** A multi-channel broadcast encryption scheme is said $(t, \varepsilon, q_C, q_D, q_E)$-secure if for any $t$-time algorithm $A$ that makes at most $q_C$ corruption queries, $q_D$ decryption queries, and $q_E$ encryption queries, one has $\text{Adv}^{\text{ind}}(A) \leq \varepsilon$. We denote by $\text{Adv}^{\text{ind}}(t, q_C, q_D, q_E)$ the advantage of the best $t$-time adversary.

There are two classical restricted scenarios: a selective attacker provides the target sets $S_1^*, S_2^*, \ldots, S_t^* \subseteq \{1, \ldots, n\}$ and index $j$, which specifies the attacked target set $S_j^*$, at the beginning of the security game, and one can also restrict the adversary not to ask some queries.

**Definition 4.2.2 (Basic Selective Security)** A multi-channel broadcast encryption scheme is said to be $(t, \varepsilon, q_C)$-selectively secure if it is $(t, \varepsilon, q_C, 0, 0)$-secure against a selective adversary. We denote by $\text{Adv}^{\text{b-ind}}(t, q_C)$ the advantage of the best $t$-time basic selective adversary.
Note that in the public broadcast setting (where encryption is public), this just excludes decryption queries: we allow CPA adversaries.

**Definition 4.2.3 (Strong Selective Security)** A multi-channel broadcast encryption scheme is said to be \((t, \varepsilon, q_C, q_D, q_E)\)-selectively secure if it is \((t, \varepsilon, q_C, q_D, q_E)\)-secure against a selective adversary. We denote by \(\text{Adv}^{\text{sec}}(t, q_C, q_D, q_E)\) the advantages of the best \(t\)-time strong selective adversaries.

This definition is much stronger since it not only allows decryption queries in the public setting, but also encryption queries in the private setting.

### 4.3 Constructions

#### 4.3.1 BGW Scheme

As a warm up, we first recall the BGW scheme [BGW05], on which our constructions will rely.

**Setup**\((\lambda)\): Let \(G\) be a bilinear group of prime order \(p\). The algorithm first picks a random generator \(g \in G\) and a random scalar \(\alpha \in \mathbb{Z}_p\). It computes \(g_i = g^{\alpha i} \in G\) for \(i = 1, 2, \ldots, n, n + 2, \ldots, 2n\). Next, it picks a random scalar \(\gamma \in \mathbb{Z}_p\) and sets \(v = g^\gamma \in G\). The public key is \(E_K = (g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, v)\), whereas the private decryption key of user \(i \in \{1, \ldots, n\}\) is \(d_i = v^{\alpha_i}\). These decryption keys are sent by the **Extract** algorithm.

**Encrypt**\((S, E_K)\): Pick a random scalar \(r \in \mathbb{Z}_p\), and set \(K = e(g_{n+1}, g)^r\), where \(e(g_{n+1}, g)\) can be computed as \(e(g_n, g_1)\) from \(E_K\). Next, set: \(\text{Hdr} = (g^r, (v \cdot \prod_{j \in S} g_{n+1-j})^r)\), and output \(\text{Hdr}, K\).

**Decrypt**\((S, \text{Hdr}, i, d_i, E_K)\): Parse \(\text{Hdr} = (C_1, C_2)\), output \(K = e(g_i, C_2)/e(d_i \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, C_1)\).

Trivially, when one wants to broadcast \(m\) different messages to \(m\) different sets \(S_1, S_2, \ldots, S_m\), one can combine \(m\) independent BGW schemes:

**Setup**\((\lambda)\): As in the BGW scheme.

**Encrypt**\((S_1, S_2, \ldots, S_m, E_K)\): Pick random scalars \(r_1, \ldots, r_m \in \mathbb{Z}_p\), and set \(K_1 = e(g_{n+1}, g)^{r_1}, \ldots, K_m = e(g_{n+1}, g)^{r_m}\)

\[
\text{Hdr} = \left(\left(g^{r_1}, (v \cdot \prod_{j \in S_1} g_{n+1-j})^{r_1}\right), \ldots, \left(g^{r_m}, (v \cdot \prod_{j \in S_m} g_{n+1-j})^{r_m}\right)\right)
\]

**Decrypt**\((S_1, \ldots, S_m, \text{Hdr}, i, (E_K, d_i), j)\): Extract \(C_1 = g^{r_j}, C_2 = (v \cdot \prod_{j \in S_j} g_{n+1-j})^{r_j}\) from \(\text{Hdr}\) and decrypt as in BGW.
4.3.2 Intuition

One can note that, in the above “trivial” construction, the number of elements in the header is $2^m$, and we want to reduce it. A first attempt is by reusing the same random scalar in all the ciphertexts, which leads to a header of size $m + 1$:

$$\text{Hdr} = \left(g^r, v \cdot \prod_{j \in S_1} g_{n+1-j}^r, \ldots, (v \cdot \prod_{j \in S_m} g_{n+1-j})^r\right).$$

However, this reuse of random coins suffers from a simple attack: the same random coins result in the same session keys for all channels and a subscriber of a channel can decrypt all channels, since the session key is $e(g_{n+1}, g)^r$. Different $r$’s are thus required in each session keys, but not necessarily totally independent. Our idea is to add an element $X_i \in G$ corresponding to users $i = 1, \ldots, n$, and to adapt the session key and Hdr using scalars $x_i$, where $X_i = g^{x_i}$, for $i = 1, \ldots, n$,

$$K_1 = e(g_{n+1}, g)^{r + \sum_{j \in S_1} x_j}, \ldots, K_m = e(g_{n+1}, g)^{r + \sum_{j \in S_m} x_j},$$

$$\text{Hdr} = \left(g^r, (v \cdot \prod_{j \in S_1} g_{n+1-j})^{r + \sum_{j \in S_1} x_j}, \ldots, (v \cdot \prod_{j \in S_m} g_{n+1-j})^{r + \sum_{j \in S_m} x_j}\right).$$

The above step shorts the header to $m + 1$ elements, with no more easy attack. But our goal is to have a constant number of elements:

$$\text{Hdr} = \left(g^r, (v \cdot \prod_{j \in S_1} g_{n+1-j})^{r + \sum_{j \in S_1} x_j} \times \cdots \times (v \cdot \prod_{j \in S_m} g_{n+1-j})^{r + \sum_{j \in S_m} x_j}\right)$$

where we essentially multiply all the ciphertexts together. And, magically, it works because a user in a set $S_i$ can cancel out all the terms $(v \cdot \prod_{j \in S_k} g_{n+1-j})^{r + \sum_{j \in S_k} x_j}$ for $k \neq i$ in this product and transform it into his corresponding ciphertext in $S_i$.

Of course, security has to be proven, this is the goal of the next section to prove the basic selective security. Limitation not to ask decryption nor encryption queries is quite strong, and is the main drawback of the first scheme MCBE$_1$. And thus, we provide a second construction MCBE$_2$ that covers strong selective adversaries. For that, we replace $\prod_{j \in S_k} X_j$ by a value outputted by a random oracle on the set $S_k$ and the value $g^r$ at the time of encryption. It will prevent malleability. The dummy-helper technique will make the rest.

4.3.3 Multi-Channel Broadcast Encryption I – MCBE$_1$

Description

Let us now describe formally our first construction MCBE$_1$. We will then prove its basic selective security.

Setup$(\lambda)$: The algorithm takes as input the security parameter $\lambda$, it generates the global parameters $\text{param}$ of the system as follows: Let $G$ be a bilinear group of prime
order $p$. The algorithm first picks a random generator $g \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p$. It computes $g_i = g^{\alpha^i} \in \mathbb{G}$ for $i = 1, 2, \ldots, n, n + 2, \ldots, 2n$. Next, it picks a random $\gamma \in \mathbb{Z}_p$ and sets $v = g^\gamma \in \mathbb{G}$. It also picks additional random scalars $x_1, x_2, \ldots, x_n \in \mathbb{Z}_p$ and sets $X_1 = g^{x_1}, X_2 = g^{x_2}, \ldots, X_n = g^{x_n}$. The master secret key is $\text{MSK} = (g, v, \alpha, \gamma, x_1, x_2, \ldots, x_n)$, while the encryption key (that is private to the broadcaster) is $\text{EK} = (g, v, g_{n+1}, x_1, x_2, \ldots, x_n)$. The public global parameters are $(g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, X_1, X_2, \ldots, X_n)$, whereas the private decryption key is $d_i = v^{\alpha^i}$, for $i \in \{1, \ldots, n\}$. These decryption keys are sent by the Extract algorithm.

We note that if a user registers to $t$ different channels, he will possess $t$ different private decryption keys: $n$ will be the product of the number of users and the number of channels.

\section{Evidence Computation}

\begin{itemize}
\item \textbf{Encrypt}($S_1, S_2, \ldots, S_m, \text{EK}$): Pick a random scalar $r \leftarrow \mathbb{Z}_p$, set $K_k = e(g_{n+1}, g)^{r+\sum_{j \in S_k} x_j}$ for $k = 1, \ldots, m$. Next, set
\end{itemize}

$$
\text{Hdr} = \left( g^r, \prod_{k=1}^{m} \left( v \cdot \prod_{j \in S_k} g_{n+1-j} \right)^{r+\sum_{j \in S_k} x_j} \right).
$$

The broadcaster knows $g_{n+1}, x_1, \ldots, x_n$ from $\text{EK}$. It eventually outputs $(\text{Hdr}, K_1, K_2, \ldots, K_m)$. 
Decrypt($S_1, \ldots, S_m, \text{Hdr}, i, d_i, k$): Parse $\text{Hdr} = (C_1, C_2)$. If $i \in S_k$ then one computes

$$K_k = \frac{e(g_i, C_2)}{\prod_{\ell \neq k}^{\ell=m} e(d_i \cdot \prod_{j \in S_k, j \neq i} g_{n+1-j+i}, C_1 \cdot \prod_{j \in S_k} X_j)} \cdot \frac{1}{e(g_i, C_2)} \cdot e(d_i \cdot \prod_{j \in S_k, j \neq i} g_{n+1-j+i}, g^{r+\sum_{j \in S_k} x_j})$$

$$= \frac{e(g^\alpha, \prod_{\ell=1}^{\ell=m} (v \cdot \prod_{j \in S_k, j \neq i} g_{n+1-j})^{r+\sum_{j \in S_k} x_j})}{\prod_{\ell=1}^{\ell=m} e(v^{\alpha^i} \cdot (\prod_{j \in S_k, j \neq i} g_{n+1-j})^{\alpha^i} g^{r+\sum_{j \in S_k} x_j})} \cdot \prod_{\ell=1}^{\ell=m} e((v \cdot \prod_{j \in S_k, j \neq i} g_{n+1-j})^{\alpha^i} g^{r+\sum_{j \in S_k} x_j})$$

$$= \frac{e((v \cdot \prod_{j \in S_k, j \neq i} g_{n+1-j})^{\alpha^i} g^{r+\sum_{j \in S_k} x_j})}{e((v \cdot \prod_{j \in S_k, j \neq i} g_{n+1-j})^{\alpha^i} g^{r+\sum_{j \in S_k} x_j})} \cdot \prod_{\ell=1}^{\ell=m} e((v \cdot \prod_{j \in S_k, j \neq i} g_{n+1-j})^{\alpha^i} g^{r+\sum_{j \in S_k} x_j})$$

$$= e(g_{n+1-i}^{\alpha^i} g^{r+\sum_{j \in S_k} x_j}) = e(g_{n+1}; g^{r+\sum_{j \in S_k} x_j})$$

$$= e(g_{n+1}, g^{r+\sum_{j \in S_k} x_j})$$

We used the relations $d_i = v^{\alpha^i}; g_{n+1-j+i} = g^{\alpha^i}_{n+1-j}$, and $g^{\alpha^i}_{n+1-i} = g_{n+1}$. 

Remark 4.3.1 In MCBE, the encryption key $EK$ contains $g_{n+1}, x_1, x_2, \ldots, x_n$ and thus cannot be public: this is a private variant of BGW scheme. Actually, $g_{n+1}$ is not really required, but the $x_i$s would be enough to break the semantic security, and thus cannot be public either. However, the broadcaster does not need to know $\alpha, \gamma$ to encrypt, and without them it cannot generate decryption keys for users. We can separate the role of group manager (who generates the decryption keys) and broadcaster (who encrypts and broadcasts the content).

Security Result

We now prove the semantic security of the first scheme.
The MCBE system achieves the basic selective security under the DBDHE assumption in $G$. More precisely, if there are $n$ users,

$$\text{Adv}^{b-\text{ind}}(t, q_C) \leq 2 \times \text{Adv}^{\text{dbdhe}}(t', n),$$

for $t' \leq t + (mn + nq_C)T_e$ where $T_e$ is the time complexity for computing an exponentiation and $m$ is the maximum number of channels in the system.

**Proof:** Let us assume there exists an adversary $A$ which breaks the semantic security of our first scheme, we build an algorithm $B$ that has the same advantage in deciding the DBDHE problem in $G$. This algorithm $B$ proceeds as follows:

1. **Init.** Algorithm $B$ first takes as input a DBDHE instance $(g, G, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, T)$, where $T$ is either $e(g_{n+1}, G)$ or a random element of $G$. It implicitly defines $o$: $g_i = g^{\alpha_i}$. $B$ then runs $A$, and since we are in the selective model, it receives $m$ sets $S_1, \ldots, S_m$ and an index $k$ that $A$ wishes to be challenged on.

2. **Setup.** $B$ now generates the public global parameters and private keys $d_i$, for $i \notin S_k$: it first chooses a random scalar $r \in \mathbb{Z}_p$ and sets $h = g^r$, and $h_i = g_i^r$, for $i = 1, \ldots, n$. One chooses a random index $\eta$ in $S_k$, and for $i \in \{1, \ldots, n\}\{\eta\}$, one chooses a random scalar $x_i \in \mathbb{Z}_p$, and computes $X_i = g^{x_i}$. One eventually sets $X_\eta \overset{\text{def}}{=} G/\prod_{i \in S_k \setminus \{\eta\}} X_i = g^{\alpha_\eta}$: All the scalars $x_i$ are known, excepted $x_\eta$. $B$ gives $A$ the public global parameters:

$$(g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, X_1, X_2, \ldots, X_n)$$

$B$ has to compute all the private decryption keys $d_i$ except for $i \in S_k$: It chooses a random $u \in \mathbb{Z}_p$ and sets

$$v \overset{\text{def}}{=} g^u \cdot \left(\prod_{j \in S_k} g_{n+1-j}\right)^{-1}$$

$$d_i \overset{\text{def}}{=} \frac{g_i^u}{\prod_{j \in S_k} g_{n+1-j+i}} = g^{u \cdot \alpha_i} \cdot \left(\prod_{j \in S_k} g_{n+1-j}\right)^{-\alpha_i} = v^{\alpha_i}$$

On can remark that $B$ can compute, without explicitly knowing $\alpha$, $\prod_{j \in S_k} g_{n+1-j+i}$ for any $i \notin S_k$, and cannot when $i \in S_k$. Moreover, since $d_i = v^{\alpha_i}$, it satisfies the specifications of the schemes.

3. **Challenge.** To generate the challenge for $A$, $B$ first computes $\text{Hdr} = (C_1, C_2)$ by setting
\[ C_1 = h, \text{ and} \]
\[ C_2 = (h^u \cdot G^u) \cdot \prod_{\ell=1, \ell \neq k}^{\ell=m} \left( h^u \cdot \left( \prod_{j \in S_k} h_{n+1-j} \right) \right) \cdot \left( v \cdot \prod_{j \in S_t} g_{n+1-j} \right) \sum_{j \in S_t} x_j \]
\[ = \left( \prod_{j \in S_k} g_{n+1-j} \right) \sum_{j \in S_t} x_j \]
\[ = \left( v \prod_{j \in S_k} g_{n+1-j} \right) \sum_{j \in S_t} x_j \]
\[ = \prod_{\ell=1}^{\ell=m} \left( v \prod_{j \in S_k} g_{n+1-j} \right) \sum_{j \in S_t} x_j \]

We used the following notations and relations \( h = g^r \) and \( g_{n+1-j}^r = h_{n+1-j} \).

Note that \( \mathcal{B} \) knows all the values \( x_i \), excepted \( x_{i,k,t} \), that appears in \( h^u \cdot G^u = (v \cdot \prod_{j \in S_k} g_{n+1-j})^{r+\sum_{j \in S_t} x_j} \). To generate session keys, \( \mathcal{B} \) first computes, for all \( i \neq k \), \( \tilde{K}_i = e(g_{n+1}, g)^{\sum_{j \in S_t} x_j} \cdot e(g_n, h_1) \), and sets \( K_k = T \cdot e(g_n, h_1) \). It outputs \( (\text{Hdr}, \tilde{K}_1, \ldots, \tilde{K}_m) \) as the challenge to \( \mathcal{A} \).

Note that, for \( i \neq k \), \( \tilde{K}_i = e(g_{n+1}, g)^{\sum_{j \in S_t} x_j} \), and, if \( T \) is the correct value, \( K_k = e(g_{n+1}, g) \cdot e(g_n, h_1) = e(g_{n+1}, g_{n+1-j}) \cdot e(g_{n+1}, g^r) = e(g_{n+1}, g)^{r+\sum_{j \in S_t} x_j} \). If \( T \) is random, the latter is also random.

**Guess.** \( \mathcal{A} \) outputs its guess \( b' \) for \( b \). If \( b' = b \) the algorithm \( \mathcal{B} \) outputs 0 (indicating that \( T = e(g_{n+1}, G) \)). Otherwise, it outputs 1 (indicating that \( T \) is random in \( \mathbb{G}_1 \)). From the above remark, if \( T \) is the correct value, \( \Pr[\mathcal{B} = 1] = \Pr[b' = b] = (\text{Adv}^{\text{ind}}(\mathcal{A}) + 1)/2 \). However, if \( T \) is a random value, \( \Pr[\mathcal{B} = 1] = 1/2: \text{Adv}^{\text{dbdhe}}(\mathcal{B}) = \text{Adv}^{\text{ind}}(\mathcal{A})/2 \).

### 4.3.4 Multi-Channel Broadcast Encryption II – MCBE₂

We now improve the previous scheme to allow encryption and decryption queries. To this aim, we will need a random oracle.
4.3. CONSTRUCTIONS

Dummy-Helper Technique

First, in order to achieve semantic security, we still have to embed the critical element from the \( n - \text{BDHE} \) instance in the challenge header related to the specific target set \( S_k \). In the previous scheme, it was implicitly embedded in the \( X_q \), or at least in one of them. But then, if this element is involved in a decryption query, the simulator cannot answer, hence the limitation for the adversary not to ask decryption queries. For the same reason, it was not possible to simulate encryption queries with this critical value.

Using a random oracle, it is possible to embed this element at the challenge time only, and then, instead of a deterministic \( \sum_{i \in S_i} x_i \) one can use a random \( y_j \) implicitly defined by \( Y_j \) given by a random oracle. With the knowledge of the discrete logarithm \( y_j \) (excepted in the challenge ciphertext), the simulator is able to answer all encryption queries, but this is still not enough to answer decryption queries: the simulator has no idea about the random scalar \( r \) involved in the ciphertext, whereas it has to compute \( e(g_{n+1}, g)^r \). But this can be done by adding a dummy set for which the session key can be computed by the simulator. In this case, we apply the dummy-helper technique to prove the security.

Description

**Setup**(\( \lambda \)): it takes as input the security parameter \( \lambda \), and generates the global parameters \( \text{param} \) of the system as follows: Let \( \mathbb{G} \) be a bilinear group of prime order \( p \); pick a random generator \( g \in \mathbb{G} \) and a random scalar \( \alpha \in \mathbb{Z}_p \); compute \( g_i = g^{\alpha^i} \in \mathbb{G} \) for \( i = 1, 2, \ldots, 2n \); pick a random scalar \( \gamma \in \mathbb{Z}_p \) and set \( v = g^\gamma \in \mathbb{G} \) and \( d_n = v^{\alpha^n} \). The algorithm also uses a random oracle \( \mathcal{H} \) onto \( \mathbb{G} \).

The master key is \( \text{MSK} = (g, v, \alpha, \gamma) \), the private encryption key is \( \text{EK} = \text{MSK} \) and the public global parameters are \( (g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, d_n) \), whereas the private decryption key is \( d_i = v^{\alpha^i} \), for \( i \in \{1, \ldots, n\} \). These decryption keys are sent by the **Extract** algorithm.

**Encrypt**\((S_1, \ldots, S_m, \text{EK})\): Pick a random scalar \( r \in \mathbb{Z}_p \); set \( S_{m+1} = \{n\} \), for each set \( S_i \), for \( i = 1, \ldots, m + 1 \) compute \( Y_i = \mathcal{H}(i, g^r) \) (\( Y_i = g^{y_i} \), for some unknown scalar \( y_i \)), and for \( i = 1, \ldots, m + 1 \)

\[
K_i = e(g_{n+1}, Y_i) \cdot e(g_{n+1}, g)^r = e(g_{n+1}, g)^{r+y_i}
\]

Eventually compute \( \text{Hdr} = (C_1, C_2, C_3) \) as follows:

\[
C_1 = g^r
\]

\[
C_2 = \prod_{i=1}^{i=m+1} \left( Y_i^{\gamma + \sum_{j \in S_i} \alpha^{n+1-j}} \cdot \left( v \cdot \prod_{j \in S_i} g_{n+1-j} \right)^i \right) = \prod_{i=1}^{i=m+1} \left( v \cdot \prod_{j \in S_i} g_{n+1-j} \right)^{r+y_i}
\]

\[
C_3 = \mathcal{H}(C_1, C_2)^r
\]

Note that the broadcaster knows both \( \alpha \) and \( \gamma \) to compute \( C_2 \). It outputs \( (\text{Hdr}, K_1, \ldots, K_{m+1}) \).

**Decrypt**\((S_1, \ldots, S_m, \text{Hdr}, i, d_i, k)\): Set \( S_{m+1} = \{n\} \), parse \( \text{Hdr} = (C_1, C_2, C_3) \). If \( i \in S_k \) then one first checks whether \( e(C_1, \mathcal{H}(C_1, C_2)) = e(g, C_3) \), computes \( Y_i = \mathcal{H}(i, g^r) \),
CHAPTER 4. MULTI-CHANNEL BROADCAST ENCRYPTION

Game 1: In this game, we know all the secret keys, but answer the CDH given a hdr assumption in Theorem 4.3.3. Security We organize our proof in three games:

Proof: For $i = 1, \ldots, m+1$, and computes (as in the previous scheme, where the $Y_j$’s replace some products of the $X_j$’s)

$$K_k = e(g_1, C_2) \cdot \frac{1}{e(d_i \cdot \prod_{j \leq i} g_{n+1-j+i}, C_1 \cdot Y_k)} \cdot \prod_{\ell=1}^{\ell=m+1} e(d_i \cdot \prod_{j \in S_{\ell}} g_{n+1-j+i}, C_1 \cdot Y_\ell)$$

$$= e(g_{n+1}; g)^{r\cdot y_k}$$

Note that $d_i = v^{\alpha_i}, g_{n+1-j+i} = g_{n+1-j}^{\alpha_i}$, and $g_{n+1-j}^{\alpha_i} = g_{n+1}$.

Theorem 4.3.3 The MCBE system achieves the strong selective security under the DBDHE assumption in $G$. More precisely, if there are $n$ users,

$$\text{Adv}^{\text{ind}}(t, q_C, q_D, q_E) \leq 2 \times \text{Adv}^{\text{dbdhe}}(t', n) + 2 \times \text{Succ}^{\text{cdh}}(t'') + 2/p,$$

for $t' \leq t + (mq_C + nmq_D + nmq_E)T_e + (mq_D + mq_E)T_p + mq_DT_{lu}$ and $t'' \leq t + (q_C + q_D + nmq_E)T_e + (q_D + mq_E)T_p + q_DT_{lu}$, where $T_e, T_p$ are the time complexity for computing an exponentiation, a pairings, $T_{lu}$ is the time complexity for a look up in a list, and $m$ is the maximum number of channels in the system.

Proof: We organize our proof in three games:

1. **Game 0**: The real strong selective security game between an adversary and a challenger.

2. **Game 1**: This is similar to Game 0 with a following exception: if we denote $\text{Hdr} = (C_1, C_2, C_3)$ the challenge header, then any decryption query on a different header $\text{Hdr}' = (C_1', C_2', C_3')$, but with the same $C_1$, we answer $\perp$ (i.e. invalid ciphertext). We can show that this exception happens with negligible probability under the CDH assumption.

3. **Game 2**: We can now safely answer all decryption queries $\text{Hdr}' = (C_1', C_2', C_3')$ by $\perp$ and the others using either a valid decryption key or $d_n$. Using the programmability of the random oracle, and thus the knowledge of the $y_i$, one can easily simulate the encryption queries. Eventually, the semantic security then relies on the DBDHE assumption.

**Game 1**: In this game, we know all the secret keys, but answer $\perp$ to a decryption query $\text{Hdr}' = (C_1', C_2', C_3')$, with the same first $C_1$ as in the challenge header. Our algorithm $B$ is given a CDH instance $g, A = g^{\gamma}, B$, and should answer $C = B^{\gamma}$. It runs the adversary $A$:

- since we consider selective attacks only, the target sets are known from the beginning, and $B$ can thus first generate the challenge header using $r^\alpha$ as random scalar, without knowing it: $C_1 = A$. Since $B$ knows $\text{MSK}$, and namely $\alpha$ and $\gamma$, it can compute the appropriate $C_2$: $v^{\gamma} = A^\gamma$ and $g_1^{\gamma} = A^{\alpha_i}$. It then programs $H(C_1, C_2) = g_u$ for a random scalar $u$ and sets $C_3 = A^u$. The triple $(C_1, C_2, C_3)$ is a perfect header;
4.3. CONSTRUCTIONS

- answers all the hash queries $\mathcal{H}(A,X)$, for any $X$, by $B'$ for some randomly chosen scalar $t$;
- answers all the other queries with MSK.

Let us now assume that $A$ asks for a valid decryption query $(S'_1, \ldots, S'_{m'+1}, k', \text{Hdr}')$ in which $C'_1 = A$. Since $C'_3 = \mathcal{H}(C_1, C'_2)^{r'} = B'^{-t}$ for a known value $t$, one can extract $C = B^r = (C'_3)^{1/t}$, which breaks the CDH assumption. $\text{Succ}^\text{ind}(A) - \text{Succ}_1(A) \leq \text{Succ}^\text{cdh}(B)$.

**Game 2:** We now assume there exists a selective adversary $A$ that breaks the semantic security of our scheme while decryption queries with the same $C_1$ as in the challenge are answered by $\perp$. We build an algorithm $B$ that has twice the advantage in deciding the DBDHE in $\mathbb{G}$. As said above, the programmability of the random oracle will help simulating the encryption queries, and the dummy set will help answering the decryption queries. In game 2.1, the algorithm $B$ is defined as follows:

**Init.** Algorithm $B$ first takes as input a DBDHE instance $(g, G, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, T)$ where $T = e(g_{n+1}, G)$. It implicitly defines $\alpha$: $g_i = g^{\alpha_i}$. $B$ then runs $A$ to receive $m^*$ sets $S'_1, \ldots, S'_{m^*}$ and an index $k^*$ that $A$ wishes to be challenged on. Note that $n \notin S'_k$, because the decryption key $d_n$ is public. $B$ makes use of a random oracle $\mathcal{H}$ which output is a random element in $\mathbb{G}$, and a hash List is initially set empty list, to store all the query-answer, with additional information, when possible. Namely, for a query $q$, with answer $Y = g^y$, the tuple $(q, Y, y)$ is stored. Sometimes, $y$ will not be known, and thus replaced by $\perp$.

**Setup.** $B$ needs to generate the public global parameters and decryption keys $d_i, i \notin S'_k$: it chooses a random $u \in \mathbb{Z}_p$ and sets $v \overset{\text{def}}{=} g^u/\prod_{j \in S'_k} g_{n+1-j}$. It then computes

$$d_i \overset{\text{def}}{=} g_i^u/\prod_{j \in S'_k} g_{n+1-j+i} = g^{u-\alpha_i} \cdot \left(\prod_{j \in S'_k} g_{n+1-j}\right)^{-\alpha_i} = v^{\alpha_i}.$$

Eventually, $B$ gives $A$ the public global parameters $(g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, d_n)$.

**Phase 1.** Since we now allow encryption and decryption queries, let show how they can be answered. We first start by the hash queries:

1. There are two kinds of useful hash queries, $(j, u) \in \mathbb{Z}_p \times \mathbb{G}$ or $(u_1, u_2) \in \mathbb{G}^2$. But for any query $q$, if it has already been asked, the same answer is sent back. Otherwise, $B$ chooses a random scalar $y \overset{\$}{\leftarrow} \mathbb{Z}_p$ and sets $\mathcal{H}(q) = g^y$. It appends the appropriate tuple $(q, g^y, y)$ to the hash List.

2. For an encryption query $(S_1, S_2, \ldots, S_m)$, $B$ makes the ciphertext as follows: it first chooses a random scalar $r \in \mathbb{Z}_p$ and sets $S_{m+1} = \{n\}$, and $Y_i = \mathcal{H}(i, g^r) = g^{p_i}$ for $i = 1, \ldots, m + 1$: $y_i$ is obtained from the hash List. To generate $\text{Hdr} = (C_1, C_2, C_3)$, $B$ sets $C_1 = g^r$, and computes

$$C_2 = \prod_{i=1}^{m+1} \left(v \cdot \prod_{j \in S_i} g_{n+1-j}\right)^{r+y_i}, C_3 = \mathcal{H}(C_1, C_2)^r$$
and \( K_i = e(g_n, g_1)^{r+y_i}, \) for \( i = 1, \ldots, m + 1. \)

3. For a decryption query \((S_1, \ldots, S_{m+1}, \text{Hdr}, i, k)\) in the name of user \( i \in S_k, \mathcal{B}\) decrypts as follows: it first checks whether \( S_k \subseteq S_k^* \) or not. In the negative case, it finds \( j \in S_k \setminus S_k^* \), and using \( d_j \) it can decrypt as the decryption oracle would do; in the positive case

- \( \mathcal{B} \) uses \( d_n \) to decrypt, using the decryption oracle, and obtain \( K_{m+1} = e(g_{n+1}, g)^{r+y_{m+1}}; \)
- \( \mathcal{B} \) extracts, from the hash List for \( \mathcal{H}(m+1, C_1) \), the value \( y_{m+1} \), and computes
  \[
  L = \frac{K_{m+1}}{e(g_n, g_1)^{y_{m+1}}} = e(g_n, g_1)^r
  \]
- \( \mathcal{B} \) extracts, from the hash List for \( \mathcal{H}(k, C_1) \), the value \( y_k \), and computes the session key
  \[
  K_k = L \times e(g_n, g_1)^{y_k} = e(g_n, g_1)^{r+y_k}
  \]

**Challenge.** The challenge has to be generated on the target sets \( S_1^*, \ldots, S_{m^*}^* \), with the index \( k^* \) for the indistinguishability of the key:

- \( \mathcal{B} \) first chooses a random scalar \( r^* \in \mathbb{Z}_p \) and sets
  \( h = g^{r^*}, \) and \( h_i = g_i^{r^*} \) for \( i = 1, \ldots, n; \)
- it chooses a random scalar \( z^* \in \mathbb{Z}_p \) and sets
  \( \mathcal{H}(k^*, h) = Y_{k^*}^* = G/g^{z^*}, \) which is the value \( Y_{k^*}^* = g^{y_{k^*}} \) for an unknown \( y_{k^*} \). The tuple \((k^*, h, Y_{k^*}^*, \perp)\) is appended to the hash List;
- \( \mathcal{B} \) asks for the other values \( Y_i^* = \mathcal{H}(i, h) = g^{y_i}, \) for \( i = 1, \ldots, k^* - 1, k^* + 1, \ldots, m^* + 1 \)

Note that \( S_{m^*+1} = \{n\} \), then \( \mathcal{B} \) generates \( \text{Hdr}^* = (C_1^*, C_2^*, C_3^*) \) by setting \( C_1^* = h \) and \( C_3^* = \mathcal{H}(C_1^*, C_2^*)^{r^*} \), where (as in the previous proof)

\[
C_2^* = \left( h^u \cdot (Y_{k^*}^*)^{y_i} \right) \prod_{\ell=m^*+1}^{=m^*} \left( h^{y_i} \cdot \frac{\prod_{j \in S_i^*} h_{n+1-j}^{y_i}}{\prod_{j \in S_{k^*}^*} h_{n+1-j}} \right) \left( v \prod_{j \in S_{k^*}^*} g_{n+1-j} \right)^{y_i} = \prod_{\ell=1}^{=m^*} \left( v \prod_{j \in S_{k^*}^*} g_{n+1-j} \right)^{r^*+y_i}
\]

To generate the session keys, \( \mathcal{B} \) first computes

\[
K_i^* = e(g_n, g_1)^{y_i} \cdot e(g_n, h_1) = e(g_{n+1}, g)^{r^*+y_i}, \quad i \neq k^*
\]

It then sets

\[
K_{k^*}^* = \frac{T \cdot e(g_n, h_1)}{e(g_{n+1}, g^{r^*})}
\]

It gives \((\text{Hdr}^*, K_1^*, \ldots, K_{m^*+1}^*)\) as the challenge to \( \mathcal{A} \).

Note that when \( T = e(g_{n+1}, G), \) with \( G = Y_{k^*}^* g^{z^*}, \)

\[
K_{k^*}^* = \frac{(g_{n+1}, Y_{k^*}^* g^{z^*}) \cdot e(g_n, h_1)}{e(g_{n+1}, g^{r^*})} = e(g_{n+1}, g)^{y_{k^*}} \cdot e(g_{n+1}, g)^{r^*} = e(g_{n+1}, g)^{r^*+y_{k^*}}.
\]
Phase 2. $B$ responds as in the first phase. Note that, if $A$ asks a decryption query with $C_1 = C'_1$, $B$ simply answers ⊥.

In this game 2.1, the advantage of $A$ is unchanged, except in case of problem during the programmation of $H$, which is required once only, and the query has already been asked with probability $1/p$: $\text{Succ}_1(A) - \text{Succ}_{2,1}(A) \leq 1/p$. In a game 2.2, we replace $T$ by a random element in $G$: $\text{Succ}_{2,2}(A) = 1/2$, whereas $\text{Succ}_{2,1}(A) - \text{Succ}_{2,2}(A) \leq \text{Adv}^{\text{dbdhe}}(B)$.

As a consequence,

$$\text{Succ}^{\text{s-ind}}(A) \leq \text{Succ}^{\text{cdh}}(B_1) + \text{Adv}^{\text{dbdhe}}(B_2) + 1/p + 1/2,$$

where $B_i$ denotes the simulator $B$ in Game $i$. 

■
Chapter 5

Conclusion

In this thesis, we give three contributions to the domain of broadcast encryption. In the first contribution, we focus on the non-black-box tracing and the single-key black box tracing models and proposed an optimal and practical scheme in these models. As far as we know, this is the first practical fully collusion resistant traitor tracing scheme. However the most important open problem in traitor tracing remains the construction of a practical fully collusion resistant traitor tracing scheme in the general black box tracing model. The schemes in [BP08, BN08] have constant ciphertext size but when considering the full collusion, the secret key size of user is $O(N^2)$ which is impractical. The most relevant schemes in [BSW06] and in [BW06] still have large ciphertext size of $O(\sqrt{N})$ and require the use of bilinear maps in groups of composite order. One of the promising direction is to consider a model between the single-key black box tracing and the general black box tracing model in which one can still achieve a practical scheme.

In the second contribution, we consider two types of recent attacks on BE schemes, pirate evolution attack and Pirates 2.0, to which we propose two solutions to cope with these attacks. Our first solution is that we integrate a WIBE scheme into the CS scheme to form the first identity-based trace and revoke scheme (IDTR). Besides resisting well these two types of attacks, the efficiency of the IDTR scheme can be comparable to the original scheme CS, particularly it achieves the constant size secret key. In the second solution, we show that all existed solutions fighting against Pirates 2.0 attack only consider a particular form of Pirates 2.0 attack in the bounded leakage model, we thus investigate the Pirates 2.0 attacks in the bounded leakage model to find out a connection between Pirates 2.0 attack and key-leakage resilient cryptography. To this aim, we first formally define a Pirates 2.0 attack in the bounded leakage model, and then propose a key-leakage resilient scheme which resists Pirates 2.0 attack in the bounded leakage model.

For the last contribution, we initiate the new research line on multi-channel broadcast encryption. Our objective is to optimize the ciphertext size while maintaining the polynomial-time complexity of all the algorithms. We propose two efficient schemes with constant-size ciphertexts and leave some challenging open problems:

- As already mentioned, our schemes share the same weakness as with BGW scheme: the decryption takes into account of all the corresponding public keys of the users in all the target sets. It is thus not quite efficient. A trade-off between the ciphertext
size and the decryption time can be done by partitioning the sets for each channel into subsets and then encrypting to each of these subsets. A better solution than this trade-off would definitely be very interesting.

• While privacy concerns imply independent keys for all the channels a user subscribed to, this however also leads to large decryption keys for users (linear in the number of channels). One could prefer to have shorter or even constant-size keys, sacrificing on privacy. This problem is quite related to the above one.

• Our first scheme achieves the basic selective security level in the standard model while our second scheme achieves the strong selective security level, which resists to both CPA and CCA, but in the random oracle model. Ruling out the random oracle seems quite challenging because of the implicit relations between session keys.
Bibliography


BIBLIOGRAPHY


[TSN06] Dongyu Tonien and Reihaneh Safavi-Naini. An efficient single-key pirates tracing scheme using cover-free families. In Jianying Zhou, Moti Yung, and
BIBLIOGRAPHY


Résumé

Dans cette thèse, nous étudions le domaine de la diffusion de données chiffrées avec traçage de traîtres et nous avons trois contributions. Tout d’abord, dans le contexte de traçage de traîtres, nous rappelons les trois modèles dans la littérature: modèle de traçage sans boîte-noire, modèle de traçage avec boîte-noire à une clé, et modèle de traçage avec boîte-noire général. Le dernier modèle est évidemment le plus important car il couvre toutes les stratégies d’adversaires, mais les deux premiers modèles sont également utiles car il couvre de nombreux scénarios pratiques. Nous proposons un schéma optimal dans le modèle de traçage sans boîte-noire et dans le modèle de traçage avec boîte-noire à une clé.

Ensuite, nous étudions les deux nouveaux types d’attaques avancées (à savoir les pirates évolutifs et les Pirates 2.0) qui ont été proposées pour souligner certaines faiblesses des schémas du paradigme de subset-cover, ou plus généralement des schémas combinatoires. Comme les schémas du paradigme de subset-cover ont été largement mis en œuvre dans la pratique, il est nécessaire de trouver des contre-mesures pour résister à ces deux types d’attaque. Dans notre travail, nous construisons deux types de schémas qui sont relativement efficaces et qui résistent bien deux types d’attaque.

Enfin, nous étudions un modèle généralisé de la diffusion de données chiffrées qui peut être appliquée dans la pratique comme dans les systèmes de télévision à péage. Dans ce contexte, le centre peut chiffrer plusieurs messages à plusieurs ensembles des utilisateurs légitimes “en même temps”. Nous appelons cette nouvelle primitive la diffusion de données chiffrées multi-chaîne. Nous arrivons à construire un schéma efficace avec les chiffres de taille constante.

Mot-Clés: diffusion de données chiffrées, traçage de traîtres, sécurité prouvée.

Abstract

In this thesis, we work on the domain of broadcast encryption and tracing traitors. Our contributions can be divided into three parts. We first recall the three tracing models: non-black-box tracing model, single-key black box tracing model, and general black box tracing model. While the last model is the strongest model, the two former models also cover many practical scenarios. We propose an optimal public key traitor tracing scheme in the two first models.

We then consider two new advanced attacks (pirate evolution attack and Pirates 2.0) which were proposed to point out some weaknesses of the schemes in the subset-cover framework, or more generally of combinatorial schemes. Since these schemes have been widely implemented in practice, it is necessary to find some counter-measures to these two types of attacks. In the second contribution, we build two schemes which are relatively efficient and which resist well these two types of attacks.

In the last contribution, we study a generalized model for broadcast encryption which we call multi-channel broadcast encryption. In this context, the broadcaster can encrypt several messages to several target sets “at the same time”. This covers many scenarios in practice such as in pay-TV systems in which providers have to send various contents to different groups of users. We propose an efficient scheme with constant size ciphertext.

Keywords: broadcast encryption, traitor tracing, provable security.