Thèse de doctorat
de l'Université de Limoges

École doctorale 610 - Spécialité Mathématiques

Constructions de Schémas Cryptographiques Multi-utilisateurs

Préparée à
Université de Limoges et
Vietnam National University

Soutenue le 26/3/2021 par

Xuan Thanh DO

Encadrée par

Duong Hieu PHAN
Université de Limoges et
Telecom Paris, Institut
Polytechnique de Paris

Minh Ha LE
Vietnam National University et
Vietnam Institute for Advanced
Study in Mathematics

Rapporteurs

Carlos Aguilar Melchor
ISAE SUPAERO, Université de
Toulouse

Céline Chevalier
Université Panthéon-Assas
Paris 2

Examinateurs

David Pointcheval
CNRS, École Normale Supérieure

Olivier Blazy
Université de Limoges

Constructions de Schémas Cryptographiques Multi-utilisateurs

Xuan Thanh Do

Supervised by Duong Hieu Phan and Minh Ha Le

Abstract

This thesis considers a number of challenging aspects in multi-user cryptosystems such as traitor tracing, broadcast encryption, trace & revoke, and functional encryption. While a broadcast encryption scheme ensures the confidentiality of digital content against unauthorized users in the system, traitor tracing is an important tool to prevent authorized users from sharing decryption keys outside because if so, the tracer, with the help of the traitor tracing algorithm, will identify which user has disclosed information.

In the first part, we focus on privacy in broadcast encryption schemes. We propose an anonymous broadcast encryption scheme in the bounded model (AnoBEB) whose security is based on the $k-$LWE assumption, which is a variant of the learning with errors (LWE) assumption. Our construction enjoys optimal efficiency (as efficient as LWE encryption) in the case where the number of users is bounded.

In the second part, we integrate the proposed AnoBEB system with a robust identifiable parent property code (IPP) into a traceable scheme. Moreover, we achieve a very strong functionality scheme, also covering revocation and thus yielding the first trace & revoke scheme from a traceability code. Our construction becomes the most efficient trace&revoke scheme for standard black-box tracing in the bounded collusion model.

The third part deals with traitor tracing algorithms for functional encryption. We introduce a new primitive, which is called *traceable functional encryption*. We then formalize the notion of security and provide a concrete construction for *traceable inner product functional encryption* (traceable IPFE). The proposed construction relies on pairings. It enjoys a high efficiency and achieves black-box confirmation.

Finally, we recall the notion of *revocable functional encryption*. We provide several pairing-based constructions for inner product functional encryption with short ciphertexts or decryption keys. We will then extend this notion to the *fine-grained revocable functional encryption* and propose a candidate construction for fine-grained revocable inner product functional encryption.

Résumé

Cette thèse considère des aspects principaux dans les cryptosystèmes multi-utilisateurs tels que la diffusion de données chiffrées, la révocation, le traçage des traîtres et le chiffrement fonctionnel. Alors qu'un schéma de diffusion de données chiffrées garantit la confidentialité du contenu numérique contre les utilisateurs non autorisés du système, le traçage des traîtres est un outil important pour empêcher les utilisateurs autorisés de partager les clés de déchiffrement à l'extérieur.

Dans la première partie, nous revisitons la *privacy* dans les schémas de diffusion de données chiffrées. Nous proposons un schéma anonyme (AnoBEB) dont la sécurité est basée sur l'hypothèse $k-$LWE, qui est une variante de l'hypothèse d'apprentissage avec erreurs (LWE). Notre construction bénéficie d'une efficacité optimale (aussi efficace que le chiffrement LWE) dans le cas où le nombre d'utilisateurs est borné.

Dans la deuxième partie, nous intégrons le système AnoBEB proposé avec un code traçable IPP robuste dans un schéma de traçage de traîtres. De plus, nous obtenons également une propriété de révocation et produisons ainsi le premier schéma de *trace&revoke* à partir d'un code traçable. Notre construction devient le schéma de *trace&revoke* le plus efficace pour le traçage en boîte noire dans le modèle de collusion bornée.

La troisième partie traite des algorithmes de traçage des traîtres pour le chiffrement fonctionnel. Nous introduisons dans un premier temps une nouvelle primitive, appelée *traceable functional encryption* (TFE). Nous formalisons ensuite la notion de sécurité et fournissons une construction concrète du TFE dans le cas du produit scalaire (*traceable* IPFE). La construction proposée repose sur des couplages sur des courbes elliptiques, est très efficace et obtient le niveau de traçabilité dit de *black-box confirmation*.

Enfin, nous rappelons la notion de *revocable functional encryption*. Nous fournissons plusieurs constructions basées sur les couplages pour le chiffrement fonctionnel dans le cas du produit scalaire avec des textes chiffrés courts ou des clés de déchiffrement courts. Nous étendrons ensuite cette notion au *fine-grained revocable functional encryption* et proposerons une construction candidate pour *fine-grained revocable inner product functional encryption*.

Acknowledgements

First and foremost, I would like to express my immense gratitude to my advisor, Duong Hieu Phan. In my view, he is a passionate, enthusiastic, energetic person in research work and a crypto expert. These are the main reasons why I am determined to pursue a doctoral degree in Cryptography. He is also an active and strong advocate to promote the development of Vietnamese cryptography. Asiacrypt 2016 international conference that he organized in Hanoi to help students, the Vietnamese crypto community have a chance to hear the best results and meet the world's leading experts. He has either directly taught or invited experts to teach crypto courses to Vietnamese students. A lot of talented students are helped by him, went to France to study. I was fortunate to be brought to Limoges and directly supervised by him. I truly thank him for his efforts. Although he is very busy with his scientific activities, he devoted as much time as possible to my study and research theme: patiently teaches me the basics of cryptography, proofs, and techniques until I get it, allowed me to attend conferences and courses to gain more new knowledge, and provided insightful discussions about the research theme. He always encourages me to ask questions and is ready to answer any questions I have about crypto. I really enjoyed the topic he gave me as I found that it is important and really meaningful. When I got stuck, he helped me get through. He often advised me to have good living habits in daily life: not to stay up late and to do sport to stay healthy. I thank him for all the wonderful things he did for me.

I also sincerely thank my co-advisor, Minh Ha Le, for his kindness and support. He is always willing to help when I needed.

I am very grateful to the reviewers, Carlos Aguilar Melchor and Céline Chevalier, who agreed to review my manuscript. I am also very grateful to David Pointcheval and Oliver Blazy for accepting as examiners in my thesis committee.

I am glad to have had the opportunity to work with David Pointcheval and Moti Yung. I would like to say thank you to them. I really learned a lot from the discussions with them.

I would like to thank the administrative and technical staff at XLIM: Nicolas Annie, Sabrina Brugier, Débora Thomas and Henri Massias, as well as Thuy Phung, Hong Duc Chu at VNU.

Thank you to all members of the Cryptis team and the Falculty of Mathematics, Mechanics and Informatics (VNU), especially the lecturers who directly taught me during my PhD studies.

I am also grateful to my Vietnamese friends: Huy Vu, Truong Mac, Chi Do, Minh Nhon, Nang Thieu, Ngoc Nguyen, Hoang Duc, Thoi, Nga, Lan. We used to play

football, ping pong, drink beer, cook together, and play rummy, board games for the holidays. I would like to thank Jérémy, my office mate, and was always excited to tell me about his papers and explain the critical points of these papers, Chloé, Charline, who were enthusiastically helping me when I first arrived in Limoges, picked me up at Limoges Station, showed me around and introduced some of the places in the city center, and sometimes we walked the park and had dinner together, Hamza, who is very close to Vietnamese students. We used to play ping pong together every weekend. With all of you, I had the best time in Limoges.

Last but not least, my deepest love goes to my family, who have always supported and understood me. They have always accompanied me throughout my studies.

Contents

1 Introduction

Modern Cryptography

Nowadays, with the widespread popularity of the internet, step by step, social activities are now done online on the internet. As a result, the need to secure cyberspace information is legitimate and growing. This demand exists in security, national defense, and state management systems. Not only that, but this need also appears in many socio-economic activities such as finance, banking, commerce, etc., and even in people's daily activities (electronic mail, payment, credit, etc.). Due to its vital significance, the field of information security lures the attention of many experts in Cryptography and Mathematics all over the world. The primary purpose of cryptography is to protect the security of communication over public and insecure channels. The content of the conversation has to be encrypted so that adversaries can learn no any information. Besides ensuring the confidentiality of the conversation's content, cryptography is also used for many other purposes, such as authentication, integrity, non-repudiation.

Before the 1970s, all cryptographic communications are based on symmetric cryptography techniques. In a symmetric cryptosystem, each pair of parties have to agree on a shared secret key which is used for both encryption and decryption. During communication, the sender uses the encryption algorithm $\mathsf{Enc}(k, m)$, where m is the message to be encrypted, and k is the secret key, to obtain a ciphertext c corresponding to plaintext m. This encrypted message is transmitted to a recipient. The recipient recovers the message using the decryption algorithm $\mathsf{Dec}(k, c)$.

In contrast to the symmetric cryptography, asymmetric cryptosystems (also known as public-key encryption) refer to the use of different keys for encryption and decryption. The decryption key is known only to a decryptor (recipient); it is called the private key. The key that is published and thus, known to any party, is called the public key. Encryption algorithm $\mathsf{Enc}(\mathsf{pk}, m)$ takes as input a message m and the public key pk, it then encrypts the message m along with the public key pk to generate a ciphertext c. The ciphertext c is then transmitted over an insecure channel. A recipient receives the ciphertext c and applies decryption algorithm $\mathsf{Dec}(\mathsf{sk}, c)$, where sk is the private key, to recover the message m.

The participants do not need anymore to agree on a shared secret key in advance. They could communicate freely over untrusted networks by communicating only the public key. This is indeed a major breakthrough and has revolutionized the field of cryptography. Nowadays, it becomes the backbone of most large-scale encryption

schemes, for instance on the Internet.

With pioneering primitives in asymmetric cryptosystems, namely key exchange in 1976 by Diffie and Hellman, public-key encryption in 1978 by Rivest, Shamir, and Adleman, the scope of applications of cryptology is significantly expanded, and a new era of modern cryptography has been open.

Public-key encryption (PKE) has always been a pillar in modern cryptography and gradually became one of the most widely used and studied cryptographic primitives. Traditional public key encryption schemes are used to provide confidentiality for one-to-one communication over a public broadcast network. Practical applications of PKE can be listed email (PGP), secure web browsing (SSH, SSL, TLS) and it occurs in almost all areas of life: e-banking, e-commerce, e-insurance,...

The last decade has witnessed the emergence of cloud computing, where millions of users worldwide with portable devices, laptops, smart devices, etc. can access and store personal data such as photos, videos, invoices, personal medical records, etc. on the Internet. Users can access and work from anywhere, anytime. Clearly, it has drastically changed the ways of communication, work, and entertainment of people worldwide. A wide range of applications, utilities, and platforms have been designed, built, and launched to meet the communication and work needs. It has gone beyond its primary purpose as a means of storing data: when a user wants to access and work with data, he will have to download it to his device and then process it. Consequently, the computation burden must be placed on personal devices where the computing capacity is very limited. In such early days of cloud computing, users simply stored personal data in a clear form. It is potentially a loss of personal data if the cloud servers are attacked and lose control. As we have analyzed, establishing trust in service providers in a computer network environment is quite risky. We cannot be sure what they will do with our data. Therefore, we should encrypt our data before posting it online. The birth of cloud computing with the primary task is to provide computing services and return results to individual devices, end-users. Nevertheless, the data stored in the cloud is in not clear form, so how can the servers compute with this data? Is there any mechanism that allows servers to compute over encrypted data without knowing anything about the underlying content?

To ensure the data privacy of users and the security of the computing systems on the cyberspace of those platforms, traditional PKE is clearly not enough to satisfy the increasing needs of users. Cryptographic primitives need to be extended and considered in the multi-user setting. There have been many important studies and implementations that have been conducted by the community of cryptographers aiming to improve and upgrade PKE. The question is whether there is any method (supporting the multi-user setting) to ensure the confidentiality of personal data where heavy computation is outsourced to servers with strong computing power in the sense that it leverages the power of cloud computing. Three options can satisfy the above question: Multi-party Computation (MPC), Fully Homomorphic Encryption (FHE) [Gen09] and Functional Encryption (FE) [BSW11].

1. MPC protocols allow parties to agree and work together to compute an arbitrary common function so that each party knows only the value of the joint function and nothing more. MPC is very useful in the design of decentralized protocols. The general result [Yao82] shows that we can have MPC solution for any function. That means MPC has supported fine-grained computation on data. However, the main limitation of such a general result is the cost of communication, in particular, the interaction between parties is extremely high and impractical.

2. FHE and multi-key FHE ([Gen09], [LTV12], [MW16]) are powerful tools and extremely useful. Gentry's groundbreaking work allowed the computation of an arbitrary function on encrypted data in the single-user setting, while multi-key FHE allows any computation on encrypted data in the multi-user setting. One notable feature of Gentry's FHE systems and variants (compared to MPC) is that the schemes are non-interactive, so they are effective at low communications. However, FHE constructions have not yet achieved practical effectiveness.

3. Functional encryption schemes capture the benefits of PKE schemes (low cost of interaction between users) and MPC protocols (fine-grained access control to the data). FE notion is introduced by Boneh, Sahai, and Waters. It is a generalization of some notion such as identity-based encryption (IBE), attribute-based encryption (ABE), predicate encryption (PE) and broadcast encryption (BE). Informally, a functional encryption scheme for a circuit family $\mathcal{C}$ associates secret keys SK_f with every circuit $f \in \mathcal{C}$, and ciphertexts CT with every input x. Any user who has the secret key SK_f and the ciphertext CT should be able to learn $f(x)$, and nothing more. There has been constructions of FE schemes for arbitrary general functions. However, these constructions are based on non-standard cryptographic assumptions like indistinguishability obfuscation ($i\mathcal{O}$) or multi-linear maps. Although there is no FE construction for every function based on well-known assumptions, there have been many effective FE constructions for linear and quadratic functions as well as extended versions of these functions in multi-input and multi-client settings.

While FE and ABE are of much interest in theory, it is still hard to believe that there will be a practical solution in their general form. Therefore, it is also important to study more basic primitives that are relevant in the real-world. In this thesis, we focus on broadcast encryption and traitor tracing in multi-user encryption and in inner-product functional encryption, which has many practical applications such as pay-TV and satellite transmission.

While broadcast encryption can be seen as a generalization of PKE to the one-to-many setting, traitor tracing is concerned with a genuinely new problem. We first look at the setting where a center encrypts a message to all users. We want to protect against users sharing their secret decryption keys with people outside the group. Users who act in this way are called traitors. Since there is no way to prevent users from sharing their secret keys or algorithms containing these keys, the idea is that identifying a traitor from his decryption key will prevent users from giving their keys away. We consider a practical scenario (pay-TV, for instance) in which a content provider wants to privately broadcast to a set of paying clients. In this scenario, each user receives a decryption box, called a decoder, containing its subscription information that helps to decrypt the broadcasted ciphertext. If an attacker can either corrupt some of the already paying clients or buy several subscriptions, then some secret information can be extracted by the adversary. The adversary then uses the information to produce a pirate device that allows anyone who owns the device to recover illegally any private content in the broadcast system. Such pirate devices caused serious economic damage to digital content delivery services. The confidentiality feature of encryption schemes does not provide any guarantees against such attackers, and this is obviously a new problem that we need to consider when designing multi-user schemes. Such schemes are generally called Traitor tracing (TT

in short). We will consider TT in the standard multi-user encryption setting and extend it to the case of FE.

1.1 Broadcast Encryption

Broadcast encryption is a generalization of one-to-one encryption to the one-to-many setting, where a sender can choose at the time of encryption the target set of users that can recover a broadcasted message. Traditional public-key encryption clearly exposes a limitation (being very inefficient) in encrypting a message to many users. When the target set of users does not change much, a possible solution is that we use a common key for all users in the target set. If users' target set often changes, which is the case in practice, efficient encryption of a message for many users may become a much more challenging task.

1.1.1 Notion

Broadcast encryption is a cryptographic primitive designed to efficiently distribute an encrypted content via a public channel to a designated set of users so that only privileged users can decrypt while the other users cannot learn anything about the content. The first constructions of broadcast encryption were proposed by Berkovits [Ber91], and most notably by Fiat-Naor [FN94] who also advocated that an efficient scheme should be more efficient than just repeating a single ciphertext per user.

In the scheme in [FN94], the ciphertext size is sublinear in the number of users; $O(t \log^2 t \log N)$ and it is secure against a collusion of t users. In real-life applications, broadcast encryption schemes were largely employed in pay TV, satellite subscription services, DVD content, multicast communication,.. Due to their usefulness, finding efficient constructions of broadcast encryption schemes has received considerable attention from cryptographers ([NNL01], [BGW05], [DPP07]).

We consider the situation in which there is a group of users whose decryption keys are obsolete or intendly leaked outsite for profit purposes. The objective of the content provider is to revoke all of those malicious users from the scheme. Roughly speaking, the content provider has to prevent them from decrypting the content even if they collude together. Revocation functionality of the distributor is inherently provided in some broadcast encryption systems. These form a special kind of broadcast encryption schemes (a variant of BE schemes) in which a content provider broadcasts encrypted messages that are generated in such a way so that all but a small subset of recipients (the "revoked" users or non-paying subscribers) can recover the message. They are called revocation schemes. In case the number of users in the system is given and fixed, we consider broadcast encryption and revocation schemes in which we do not differentiate between them. In case the set of users in a system can change and update, there is a difference between BE and revocation schemes. The Broadcast encryption scheme in this case is called dynamic BE scheme [DPP07], [KHAM08], [PPS12]. For a dynamic BE scheme, users who join the system after the broadcasted ciphertext is sent are in the revoked set, while for a revocation scheme new users are in the privileged set.

1.1.2 State of The Art in Broadcast Encryption

There are two approaches to construct broadcast encryption schemes: an algebraic approach and a combinatoric approach. The algebraic constructions often exploit properties of algebraic structures (pairings, lattices, multi-linear maps, for instance) directly. In this category, the most prominent result in broadcast encryption area was the outcome in the paper of Boneh, Gentry and Waters [BGW05]. By using a symmetric pairing in a prime-order group, Boneh, Gentry, and Waters (BGW) [BGW05] introduced the first fully collusion secure broadcast encryption scheme with constant-size ciphertext and decryption keys of users (independent of the number of users in the system) and the public key size in the system is linear in the number of users. A BE scheme is said to be collusion resistant if no coalition of users outside of target set should be able to learn the original message from the broadcasted ciphertext. The BGW scheme is proven secure in the standard model under the N-BDHE assumption (q-type assumption). One of shortcomings of Boneh et al.'s scheme [BGW05] is that it is proven selectively (static) secure where the adversary is required to choose the corrupted users before the setup. It constrasts with the notion of adaptive security. A typical scheme with adaptive security is of Gentry and Waters [GW09]. However, the main drawbacks of the scheme is that it has decryption keys with size linear in N and it is proven secure in the random oracle model.

Since the first fully collusion secure broadcast encryption scheme with constant-size ciphertext was described by Boneh, Gentry, and Waters [BGW05], many other fully collusion secure broadcast encryption scheme based on bilinear maps were proposed [Del07], [GW09], [PPSS12], [Wee16].

Gay, Kowalczyk, and Wee [GKW18a] proposed the first broadcast encryption scheme with constant decryption key and ciphertext size that achieve adaptive security under a static assumption (k-Lin) in prime-order bilinear groups. However, their construction have a limitation in which the public key is quadratic instead of linear in the total number of users in the system.

Lewko, Sahai, and Waters [LSW10] proposed an identity based revocation encryption scheme with no bound on the number of users that can be revoked by applying the 2-equation technique. Their constructions are based on symmetric bilinear maps in prime order groups and their revocation scheme has ciphertext size in $O(r)$, private key size in $O(1)$, and public key size in $O(1)$.

A family of fully collusion secure broadcast encryption schemes relied on multi-linear maps has been proposed [BW13], [GGH13]. These schemes achieve optimal constants for ciphertext size and decryption key size, but the size of public key is large and the encryption algorithms are private. In Crypto 2014, Boneh, Waters, and Zhandry [BWZ14] introduced public key broadcast encryption schemes that make use of the multi-linear map candidates. Their schemes achieve ciphertext size, decryption key size in $O(1)$ and public key size in $O(\log N)$. Lastly, the scheme of Boneh and Zhandry [BZ14] achieves constant size keys, ciphertext size and short public key. However, their scheme relies on strong assumptions namely, indistinguishability obfuscation.

At Eurocrypt 2020 [AY20], Agrawal and Yamada considered BE schemes as ciphertext policy attribute based encryption ($\mathsf{CP-ABE}$) schemes for policies that are circuits in class NC1. In a $\mathsf{CP-ABE}$ scheme, a message m will be encrypted along with a policy f, and secret keys are generated for users with public attributes x. Decryption will recover the original message m as long as the attribute x satisfies the

policy f, namely $f(x) = 1$. In the $\mathsf{CP-ABE}$ context, BE will be restated as follows: the encrypt algorithm takes as input a checking membership circuit f_S for a set of target S and a message m, it will output a ciphertext C such that only users with a secret key that encodes the attribute $i \in S$ can decrypt the message. To construct a $\mathsf{CP-ABE}$ where the sizes of the decryption key, public key, and ciphertext are independent of the number of users, they employ the decomposability property in the encryption algorithm of Boneh et al. scheme [BGG+14]. We note that Boneh's scheme is a key policy attribute based encryption ($\mathsf{KP-ABE}$), which is a dual notion of $\mathsf{CP-ABE}$ where the roles of the private key and the policy are interchanged, and the construction of $\mathsf{KP-ABE}$ for all circuit (including $\mathsf{NC1}$) has relied on an assumption on a lattice problem (LWE). Applying Boneh's construction directly does not give a collusion resistant scheme. Agrawal and Yamada lifted components of the ciphertext and secret keys into the exponent form of a group element rather than scalars to overcome this. They thus make use of a pairing to obtain a scheme with collusion resistance. The parameters of their scheme are the following: the sizes of a public key and a decryption key only depend on the depth of policy f_S and are independent of the size of that policy; the size of a ciphertext only depends on the length of the input (the length of the attributes) and also does not depend on the size of the circuit f_S. It is known that for a set of target $S \subset [N]$ the depth of policy check membership f_S is $O(\log N)$ and the input length is also $O(\log N)$. After converting $\mathsf{KP-ABE}$ into $\mathsf{CP-ABE}$, they achieved a BE scheme with optimal parameters and the security of their scheme is proved in the generic group model.

The combinatoric broadcast encryption schemes are constructions using tree structures or fingerprinting codes. Pioneering results in tree-based broadcast encryption constructions we can mention are of Fiat and Naor [FN94]; Naor, Naor, and Lotspiech [NNL01] (NNL scheme) with the Subset Cover Framework. The scheme of Fiat and Naor was designed to be secure against a collusion of t users. In this case, if there is an attacker who can compromise the private keys of more than t users, then he may break the security of the BE scheme.

In order to construct a fully collusion resistant BE scheme in the sense that the scheme is secure without a bound on the number of colluded users, Naor, Naor, and Lotspiech [NNL01] proposed a general paradigm which is called the subset cover (SC) framework, and they also proposed symmetric key revocation schemes such that a content distributor can transmit a broadcasted ciphertext to all users but r revoked users. They provided two instantiations of the SC framework which are the complete subtree (CS) and the subset difference (SD) schemes. The CS scheme has a ciphertext size in $O(r \log N/r)$ and a decryption key size in $O(\log N)$, and the SD scheme has a ciphertext size in $O(r)$ and a decryption key size in $O(\log^2 N)$ where N is the number of users in the system and r is the number of revoked users. Halevy and Shamir [HS02] proposed the layered subset difference (LSD) scheme that has a ciphertext size in $O(r)$ and a decryption key size in $O(\log^{1.5} N)$. Dodis and Fazio [DF02] proposed a public key version of the subset-cover framework using an identity based encryption hierarchical IBE (HIBE) scheme for the CS structure and HIBE of depth $\log N$ for the SD structure. Their scheme retains the same efficiency, using (H)IBE keys instead of symmetric keys.

1.2 Traitor Tracing

1.2.1 Notion

As we all know, nowadays, digital content piracy is becoming more and more serious, and this can cause significant damage to companies and economies of countries. A possible solution that can contribute to mitigating the issue is to use cryptographic tools. One such tool used to deal with this issue is traitor tracing, a fundamental primitive in cryptography. Traitor tracing ensures that anyone who intentionally violates digital copyright content can be detected and punished. A rough description of a traitor tracing (TT) system is as follows. We consider a TT scheme of N recipients (users), where each user holds a secret decryption key and a digital content distributor. The distributor uses a public key to encrypt a digital content (message). The corresponding ciphertext is then put on a public channel (insecure channel) and transmitted to all recipients. The legitimate recipients who own a valid decryption key can recover the digital content.

Suppose a coalition of recipients (traitors) pooling together their secret decryption keys, and they jointly produce an illegal decoder device (pirate device). Whenever a pirate decoder is caught, a traitor tracing scheme provides a tracing algorithm that can identify at least one of the recipients in the coalition.

The primary purpose of traitor tracing systems is to help content distributors identify traitors (pirates) who violate copyright restrictions. The very first traitor tracing scheme was proposed by Chor, Fiat, and Naor [CFN94]. A traitor tracing system basically consists of five algorithms Setup, Extract, Encrypt, Decrypt, and Tracing. The Setup algorithm generates a public key PK, a master secret key MSK, and a tracing key TK. The Extract algorithm uses the master secret key MSK to produce decryption keys $\mathsf{sk}_1, ..., \mathsf{sk}_N$, which are then delivered to legal users. The Encrypt algorithm takes as input the public key PK and a message m, it outputs a ciphertext. The message can be recovered successfully by using the Decrypt algorithm with any legal decryption key sk_i. The illegal users who do not own any legal decryption key learn nothing about the content. The Tracing algorithm takes the tracing key TK as input, interacts with a pirate decoder. It outputs at least an index of user $i \in \{1, \ldots, N\}$, which is associated with the decryption key sk_i that was used to build the pirate decoder.

Suppose that the pirate decoder only has a maximum of t decryption keys (the number of traitors has been assumed to be bounded by a threshold t). A traitor tracing scheme is called t-collusion resistant if the tracing algorithm still works correctly in this case. When the parameter t is an arbitrary polynomial, and the number of traitors is no longer required to be smaller than a certain threshold (anyone in the system can be a traitor), the scheme is called fully collusion resistant.

A traitor tracing scheme is called secret key if Tracing algorithm of the Tracer makes use of the master secret key MSK to identify traitors (tracing key is master secret key). In the opposite direction, if the Tracer only uses public information to identify traitors, then we will call it a public-key traitor tracing scheme. The public traceability property of not using any secret information to identify traitor tracing is very interesting (any party can run the Tracing algorithm), and is one of a desirable property of the traitor tracing schemes. Boneh and Franklin [BF99] introduced the first efficient public-key traitor tracing scheme that supports bounded collusions in the sense that given a positive integer t, for any collusion of traitors with size at most t, at least a traitor is accused correctly by a tracer who runs Tracing algorithm.

The scheme is efficient because the parameters: the public key and private key for each user are independent of the number of users, and the ciphertext size is only linear in t.

There are two approaches to deal with the traitor tracing problem: the algebraic approach (pairing-based and lattice-based schemes, for example) and the combinatoric approach. The algebraic schemes give us traitor tracing schemes supporting both bounded collusion [BF99], [CPP05], [LPSS14], [ABP+17] and unbounded collusion [BSW06], [BW06], [BZ14], [GKW18b] while the combinatoric approach based schemes mainly produce bounded collusion schemes [CFN94], [BS95], [BN08], [BP08],....

Traitor tracing models The crucial part of a traitor tracing scheme is the traitor tracing procedure. Typically, the traitor tracing procedure works in one of the two following models: non-black-box and black-box.

1. In a non-black-box tracing model, the tracer is supposed to be able to use the reverse engineering technique to open the pirate decoder and get the stored keys inside the pirate decoder. In this case, the tracing algorithm should find at least one traitor given the set of pirate keys. We note that the decryption keys are put in the pirate decoders may not be the same as the colluder's original decryption key. To make the traitor tracing process more difficult, the colluders intentionally deviate decryption keys such that the decrypting process of the pirate decoder still recovers the original message correctly. They do that by mixing decryption keys (for example, they take a linear combination of decryption keys randomly) and then put them into the pirate decoder.

2. In the black-box tracing model, the situation is more complicated because the reserve engineering technique may not work in this case. Then the tracer cannot open the decoder box and access the stored keys. In this setting, the tracer treats the pirate decoder as a black-box oracle. However, in this model, the tracer can do interactions with the pirate decoder by making query ciphertexts and observing the responses. That is, it can query encrypted messages to the pirate decoder and see the output of the pirate decoder. To deal with black-box pirate decoders, we can apply the linear tracing technique proposed by Boneh and Franklin. In this technique, the tracer prepares ciphertexts. Each ciphertext has many components. The pirate decoder is tested by the tracer who sends ciphertexts. In the next steps, the tracer replaces step by step each component with a random element. When we change a ciphertext component into a random component, this only affects at most one user. Thus, if the tracer detects pirate decoder decrypts differently from one step to the next, we can catch a traitor.

3. A weaker form of black-box traitor tracing is called confirmation black-box traitor tracing. In this tracing model, the purpose of this algorithm is to verify a suspected set of identities. That is, the tracer has a list of suspected identities, and he wants to check his suspicion. A tracing traitor algorithm is a black-box confirmation if it satisfies two properties:

   (a) Confirmation: If a suspected set of users actually contains the entire set of traitors, then the output of the Tracing algorithm always returns at least an identity i as guilty. More concretely, we assume that $\mathcal{K}_{\mathcal{D}}$ is a set of secret keys used to build the pirate decoder $\mathcal{D}$ and $\mathcal{K}_{\mathsf{suspect}}$ is a set

of secret keys of suspected users. With the condition $\mathcal{K}_{\mathcal{D}} \subseteq \mathcal{K}_{\mathsf{suspect}}$, the Tracing algorithm returns at least an identity i as guilty such that the secret key corresponds to the identity i in $\mathcal{K}_{\mathsf{suspect}}$ as guilty.

(b) Soundness: The honest users will never be accused if the Tracing algorithm outputs an identity as guilty; it is impossible for traitors to deceive the Tracing algorithm into blaming innocent users. Said differently, if the Tracing algorithm outputs an identity i as guilty then the secret key of i also belongs to $\mathcal{K}_{\mathcal{D}}$.

We can use black-box confirmation for black-box traitor tracing by testing all possible subsets. However this manner will give a very inefficient traitor tracing algorithm with exponential running time.

In some traitor tracing schemes, Pirate decoder's operating model is assumed to fall into one of the following categories:

Available decoders: The pirate decoder does not maintain states between decryptions. The available decoder always decrypts any ciphertext of the tracer and does not employ any reaction mechanism.

Resettable decoders: In this setting, after decrypting each ciphertext, the pirate decoder will be reset to the initial state by the tracer. It means that the pirate decoder is not allowed to maintain state. It is prevented from storing information about previous queries of tracer, and it answers each query independently.

Abrupt decoders: While available decoder always decrypts any ciphertext of the tracer and does not employ any reaction mechanism, the abrupt decoder has a reaction mechanism in the sense that it can switch to self-protection mode (finish mode) to against the tracing procedure if it detects something out of the ordinary for example it is being traced by the tracer by turning off the pirate device (refusing to decrypt further ciphertexts), making it useless, deleting colluder keys inside the pirate device, etc. Once the finish mode is enabled, the tracer is not allowed to submit any further ciphertexts anymore.

Stateful decoders: In contrast to available and resettable decoders, the stateful decoder is stronger because they have a memory to keep previously queried ciphertexts. If they detect problematic ciphertext, then they will use active reaction mechanisms or output a random message.

Requirement on the pirate decoder The requirement for pirate decoder may fall into one of three types as follows. We arrange in ascending order of strength of the requirement for pirate decoders. The first one is the weakest pirate decoder, and the last one is the strongest.

1. In a black-box traitor tracing scheme, the interaction between a tracer and a pirate decoder in which the tracer has full accesses black-box tracing in the sense that the tracer receives the actual message returned by the pirate decoder. In brief, the requirement for the pirate decoder is whenever it is fed ciphertexts that are queried by the tracer, it should return a full message.

2. Another situation of black box traitor tracing was considered in minimal access black-box tracing [BF99]. For any query to the pirate decoder, the tracer does not obtain the plaintext. However, it merely determines whether the pirate-decoder can decrypt the ciphertext and "play" it (e.g., the case of a pirate cable-box incorporating a TV-set).

3. We consider the same setting for the pirate as in [GKW18b]: of course, we do not require that the pirate decoder $\mathcal{D}$ outputs the entire message (or an indicator bit as in minimal access model) nor decrypts with high probability every ciphertext which is taken from random messages. Instead, it is enough that the pirate decoder can distinguish the encryption of two messages m_0, m_1, which are chosen by itself (see [GKW18b]). This very strong notion of Pirate Distinguisher has been introduced in [GKW18b]. It requires the pirate distinguisher to be able to distinguish the encryption of two different messages m_0, m_1. As shown in [GKW18b], this notion is stronger than the classical Pirate Decoder, which is able to correctly decrypt random messages with non-negligible probability.

1.2.2 State of The Art in Traitor Tracing

We can say that the linear tracing technique is one of the critical tools to deal with the problem of traitor tracing, and to understand the state of the art of this domain, we quickly recall this technique.

Boneh, Sahai, and Waters (BSW) [BSW06] formalized the linear tracing technique via a new primitive called Private Linear Broadcast Encryption (PLBE). A PLBE is defined as a broadcast encryption scheme such that the broadcaster can only broadcast to linear sets S_i which are target sets of the form $S_i = \{i, i+1, \ldots, N\}$ for some $i \in \{1, 2, \ldots, N+1\}$. Thus, a PLBE enables the content distributor to generate ciphertexts that can only be decrypted properly under keys $\mathsf{sk}_i, \mathsf{sk}_{i+1}, \ldots, \mathsf{sk}_N$. If the index $i = 1$, it will broadcast to everyone in the sense that a message will be encrypted with $i = 1$, and anyone can decrypt the ciphertext. A crucial requirement of a secure PLBE scheme is that a ciphertext should reveal no non-trivial information about the recipient set. This means that a broadcast to users $\{i, i+1, \ldots, N\}$ should hide non-trivial information about the index i. There is no adversary who can distinguish an encryption using the index i from an encryption using the index $i+1$ without the secret key sk_i.

By constructing a PLBE scheme using the algebraic approach, Boneh, Sahai, and Waters [BSW06] introduced the first traitor tracing scheme supporting the full collusion of traitors and achieving sub-linear sizes: $O(\sqrt{N})$ in ciphertexts, $O(\sqrt{N})$ in public keys and constant size in secret keys $O(1)$, where N is the total number of users in the system. Their method is based on bilinear maps in groups of composite order. They organize users' identities in the scheme as elements of a matrix and then apply the linear tracing technique (construct PLBE scheme) for 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 still large (always in $O(\sqrt{N})$), whatever the maximum size of collusions is. It means that even if there are few traitors, the ciphertext size is still in $O(\sqrt{N})$. Another limitation of BSW scheme is that their tracing algorithm needs using the master secret key MSK. Therefore their scheme is secret tracing. After that, Boneh and Waters [BW06] overcame this limitation to achieve a public-key traitor tracing scheme where the tracing procedure does not need using the information of the master secret key.

In independent works, Freeman [Fre10] and Garg et al. [GKSW10] further improved the Boneh-Waters scheme [BW06] by proposing traitor tracing schemes where the security of their schemes relies on hardness assumptions in prime order bilinear groups. Because we know that hardness assumptions in composite order bilinear groups are limited by known attacks on factoring their moduli, and there are sub-exponential attacks against factoring problem, we should choose large composite order groups. Moreover, operations in these larger composite order groups are much slower than operation in prime order groups. The ciphertext, secret key, and public key in their schemes are still $O(\sqrt{N})$.

Several years later, Boneh and Zhandry [BZ14] utilized indistinguishability obfuscation to achieve a scheme along with the ideal parameters where ciphertexts grow polynomially in $\log(n)$. However, we do not know how to construct an indistinguishability obfuscation from standard assumptions.

Recently, Goyal, Koppula, and Waters (GKW) [GKW18b] constructed the first secure traitor tracing scheme from the polynomial hardness assumption of Learning with Errors with sub-exponential modulus-to-noise ratio. The GKW scheme achieves ciphertext, public key, and secret key sizes only in $poly(\log N)$, where N is the number of users in the system. Moreover, GKW is secure against very strong pirate decoders, which is called pirate distinguisher. That is, the pirate decoder is required that it can distinguish encryptions of two messages, which are chosen by itself instead of decrypting ciphertexts and responding answers to the tracer as in full access and minimal access pirate decoders.

The traitor tracing scheme GKW [GKW18b] is constructed by using a 1-query secure PLBE scheme. BSW [BSW06] argued that secure PLBE schemes are sufficient for constructing traitor tracing schemes and the authors give a generic transformation to convert a PLBE scheme into a traitor tracing scheme. In other words, any secure PLBE scheme implies a secure traitor tracing scheme. GKW [GKW18b] showed that the argument was not correct by providing a counter-example to show that there exists at least one secure PLBE scheme, but there is an attacker who can construct a non-traceable pirate decoder to defeat the tracing algorithm. In [GKW18b], the PLBE scheme of Boneh, Sahai, and Waters is called 0-query secure. Goyal, Koppula, and Waters [GKW18b] argued that 1-query secure PLBE schemes, where the adversary is allowed to ask one more ciphertext query, are sufficient for constructing traitor tracing schemes.

In order to achieve a 1-query secure PLBE scheme whose ciphertext size is independent of the number of users under standard assumption, Goyal, Koppula, and Waters combined key-policy an attribute-based encryption (KP − ABE) scheme and a Mixed Functional Encryption scheme. We have known that there are KP − ABE schemes for circuits that are secure under the LWE assumption [GVW13], [BGG+14]. Mixed Functional Encryption (MFE) is a function hiding functional encryption scheme and it is also built from the LWE assumption.

Nishimaki et al. [NWZ16] introduced a very interesting concept that allows at least a colluder (dishonest user) to be traced while preserving anonymity of honest people in the system. Instead of storing all the user's information, the system manager embeds all that information into the decryption key. Once the colluders joined together to build a pirate decoder, the tracing algorithm will interact with the pirate decoder to output a traitor's identity without reaching honest users. In this manner, honest users will remain anonymous to the tracing algorithm (all their personal information is secured). Only dishonest people who have built a malicious decoder

will face the possibility of being traced. The idea of embedding user information in the decryption key can be realized in the two-party computation (2PC) context as follows. The authority, taking as input the master secret key MSK, and a user join in a protocol to generate a decryption key for user i. At the end of the calculation, user i will learn the decryption key. All information of i has been embedded, and the authority learns nothing about the decryption key. It is different from previous systems where all the user's information is stored in a database server.

We know that the ciphertext size of PLBE schemes scales linearly in the bit length of identities. If the space of identities is exponential then the ciphertext size will scale linearly with the size of identity space. It leads to a traitor tracing with large ciphertexts and the running time is very slow in this case. To reduce the ciphertext size, we can trivially use a cryptographic hash function. As a result, the tracer will output a hash value instead of an identity and he need to use a table to look up the identity corresponding to the hash value. This violates the user's privacy in the system. Nishimaki et al. [NWZ16] provided an identity-based traitor tracing scheme with short ciphertexts which scales in $\mathsf{polylog}(N)$ and preserves user privacy. The technique used in their paper is known as "jump-finding technique". Some recent results in this line are [KW20], [GKW19].

There are similarities between watermarking public key encryption and traitor tracing. A collusion resistant watermarking scheme for a public-key encryption scheme would imply a collusion resistant traitor tracing scheme. The recent construction of watermarking for public-key primitives [GKM+19] does imply a traitor tracing scheme for general identities (with public tracing), but only provides bounded collusion resistance (in fact, in this setting, their construction precisely coincides with the bounded collusion resistant traitor tracing construction from [NWZ16]).

We already know that fingerprinting is a commonly used method in protecting copyright of software products and electronic documents. This approach's idea is that each product will be duplicated into multiple copies. Each of these will be marked with a distinctive string. That is, the copy owner information is encoded into codewords and embedded in the product so that the content provider can easily trace back the copy's owner when needed.

An important attack against identification schemes using the fingerprinting method: users will join coalition together and compare data to find codewords that have been embedded in each copy, and could then be making a pirated version that cannot be traced. The fingerprinting method shares some similarities with code-based traitor tracing schemes. The codes embedded in the traitor tracing schemes are collusion secure codes. Codewords of the code need to satisfy the marking assumption. This assumption states that any word generated by an arbitrary t-collusion must be identical at all fixed positions (undetectable positions) in the sense that it does not allow to modify values at agreement positions of the t codewords. A deterministic form of collusion secure codes is IPP code (identifiable parent property). A prominent feature of schemes based on IPP codes is that the tracing algorithm always accurately accuses at least one traitor, and no one is falsely accused. If the condition of catching exactly one traitor is relaxed (the allowed tracing algorithm can output an incorrect result with negligible probability), then we will have much more efficient schemes. These are traitor tracing schemes that rely on a randomized version of collusion secure codes. Typical constructions for randomized collusion secure codes are those of Boneh-Shaw, which are proposed in [BS95] and of Tardos in [Tar03]. In Boneh-Shaw codes, given an ϵ, the length of the codewords is $O\left(N^3 \log(N/\epsilon)\right)$ for fully-collusion

resistant cases and $O(t^4 \log(N/\epsilon))$ for t-collusion resistant cases (codes resisting collusions of at most t traitors). In Tardos codes, the length of codewords is optimal: the length of the codewords is $O(N^2 \log(N/\epsilon))$ for fully-collusion resistant cases and $O(t^2 \log(N/\epsilon))$ for t-collusion resistant cases.

Kiayias and Yung [KY02] discovered the beautiful property of traitor tracing schemes using collusion secure codes that these schemes often achieve black-box tracing and transmission rate (the ratio between the ciphertexts and the plaintexts) as constant. We note that minimizing the transmission rate is a practical requirement and that traitor tracing schemes based on algebraic approaches often fail to achieve this. Kiayias and Yung introduced the first black-box traitor tracing scheme with a constant transmission rate based on collusion secure codes. They follow the code-based design paradigm: firstly, they construct a traceable PKE scheme for two users and then extend it to multiple users with a collusion-secure code. Following this approach, Phan, Safavi-Naini, and Tonien [PST06] constructed a black-box publicly traceable scheme based on IPP codes from a PKE scheme and a q-ary IPP code. Fazio, Nicolosi, and Phan [FNP07] introduced the first black-box traceable scheme with transmission rate 1 (optimal transmission rate) based on collusion-secure codes, and the scheme also applies an all-or-nothing transform (AONT) to force decoders to decrypt all ciphertexts. Apart from rate transmission, we also need to consider other parameters such as the size of secret keys, public keys, and ciphertexts. These parameters depend on the length of codes that the traitor tracing schemes are based on. Boneh-Naor [BN08] and Billet-Phan [BP08] provided schemes that obtained optimally (constant) ciphertext lengths, and thus, these are the state of the art in this line.

Trace & Revoke systems

A Trace & Revoke system can be seen as an extension of a standard traitor tracing scheme in the sense that there is an additional secret key-revocation method so that the owner of digital content can use it to disable the decryption capabilities of malicious keys. Tracing and revoking users in a multi-receiver encryption setting is an interesting and quite hard problem. At the same time, achieving these two functionalities is very challenging as broadcast encryption and traitor tracing have been viewed as two orthogonal problems. In practice, revocation should be considered together with tracing. There are very few efficient constructions of the Trace & Revoke system. There are two main approaches to tackle this problem:

1. If we restrict to a bounded model of collusions (this model is quite close to practical scenarios), there are a few efficient constructions such as [NP01], [NPP13], [NWZ16], [ABP+17].

2. When we consider the full collusion setting (all users can become traitors), there exists asymptotically efficient schemes which are built from assumptions such as indistinguishability obfuscation [NWZ16] and positional witness encryption [GVW19]. However, these are not standard assumptions. Trace & Revoke schemes constructed from standard assumption such as [BW06], [GKSW10], [GQWW19], [Zha20] seem more practical even though they are less efficient asymptotically.

1.3 Motivation for Our Works and Contributions

Anonymous Broadcast Encryption for Bounded Universe. Broadcast Encryption is a fundamental primitive supporting sending a secure message to any chosen target set of N users. Integrating privacy into BE is an important problem. While many efficient constructions for BE are known, understanding the efficiency possible for an "Anonymous Broadcast Encryption" (AnoBE), i.e., one which can hide the target set itself, is quite open. The best solutions by Barth, Boneh, and Waters [BBW06] and Libert, Paterson, and Quaglia [LPQ12] are built on public-key encryption (PKE), and their ciphertext sizes are, in fact, N times that of the underlying PKE (rate=N). Kiayias and Samari [KS12], in turn, showed a lower bound showing that such rate is the best possible if N is an independent unbounded parameter. However, when considering certain user set sizes bounded by a system parameter (e.g., the security parameter), the problem remains interesting. We consider the problem of comparing AnoBE with PKE under the same assumption. We call such schemes *Anonymous Broadcast Encryption for Bounded Universe* – AnoBEB.

Our first contribution is to construct an AnoBEB construction for up to k users from the LWE assumption, where k is bounded by the scheme security parameter. The scheme does not grow with the parameter. Actually, our scheme is as efficient as the underlying LWE public-key encryption; namely, the rate is, in fact, 1 and thus optimal.

Trace & Revoke System from AnoBEB and Robust IPP. We move on to employ the new AnoBEB in other more general broadcasting framework and, as a second contribution, we introduce a new approach to construct an efficient "Trace & Revoke scheme". Recall that, as it was put forth by Kiayias and Yung [KY02], combinatorial traitor tracing schemes can be constructed by combining a system for small universe, integrated via an outer traceability code (collusion-secure code or identifying parent property (IPP) code). There were many efficient traitor tracing schemes from traceability codes, but no known scheme supports revocation at the same time. Our new approach integrates our AnoBEB system with a robust IPP code, introduced by Barg and Kabatiansky [BK13]. This shows an interesting use for robust IPP in cryptography. The robust IPP codes were only implicitly shown by an existence proof. In order to make our technique concrete, we propose two explicit instantiations of robust IPP codes. Our final construction gives the most efficient Trace & Revoke scheme in the bounded collusion model.

The results in the above parts were published in ACNS '20 [DPY20].

Traceable Inner Product Functional Encryption. The third contribution of this thesis is related to constructing a traitor tracing scheme for functional encryption schemes. Functional Encryption (FE) has been widely studied in the last decade, as it provides a handy tool for restricted access to sensitive data: from a ciphertext, it allows specific users to learn a function of the underlying plaintext. In practice, many users may be interested in the same function on the data, say the mean value of the inputs, for example. The conventional definition of FE associates each function to a secret *decryption functional key*, and therefore all the users get the same secret key for the same function. This induces a critical problem: if one of these users (called a *traitor*) leaks or sells the decryption functional key to be included in a *pirate* decryption tool, then there is no way to trace back its identity. Our objective

is to solve this issue by introducing a new primitive, called *Traceable Functional Encryption*: the functional decryption key will not only be specific to a function but to a user too, in such a way that if some users collude to produce a pirate decoder, that successfully evaluates a function on the plaintext, from the ciphertext only, one can trace back at least one of them.

We propose a concrete solution for Inner Product Functional Encryption (IPFE). We first remark that the ElGamal-based IPFE from Abdalla et al. [ABDP15] shares many similarities with the Boneh-Franklin traitor tracing [BF99]. We can then combine these two schemes in a very efficient way, with the help of pairings, to obtain a Traceable IPFE with black-box confirmation.

The result in this part was published in CT-RSA '20 [DPP20].

Revocable Inner Product Functional Encryption. Because revocation is a crucial issue in multi-user encryption, naturally, we consider the revocation for functional encryption. The notion of *revocable functional encryption for all circuits* was first introduced by Nishimaki et al. at Eurocrypt '16 [NWZ16]. They gave a construction with optimal parameters (constant size in ciphertexts, private keys, and public parameters). However, their construction relies on a non-standard assumption (indistinguishability obfuscation).

In this part of the thesis, we are interested in constructing efficiently revocable functional encryption schemes for inner product under a standard assumption. A revocable inner product functional encryption in which there is a user who takes as input a list of revoked identities $\mathcal{R}$ and then encrypts $\mathcal{R}$ along with a message $\vec{y}$ to produce a ciphertext such that any non-revoked user can learn $\langle \vec{x}, \vec{y} \rangle$ by using a decryption key for a functionality $\vec{x}$ and nothing more. The fourth contribution of the thesis is the following:

- We give two pairing-based constructions of revocable functional encryption for inner product with short private keys. The constructions are efficient with the size of ciphertext is only $O(r)$ (r is the number of revoked users) and private key size is constant (only 3 group elements). We use the technique that combines the inner product functional encryption of Abdalla et al. at PKC '15 [ABDP15] with the two-equation technique by Lewko, Sahai, and Waters at SP '11 [LSW10].

- Using the n-equation technique of Attrapadung et al. at PKC '10 [AL10], we provide two schemes of revocable functional encryption for inner product with constant size ciphertext, independent of the number of revokers.

Our results can be viewed as an independent work of the recently published results of Abdalla et al. [ACGU20]. They also gave several pairing-based constructions of attribute-based inner product functional encryption. Their schemes are obtained from combining the inner product functional encryption scheme of Agrawal et al. [ALS16] with any attributed-based encryption schemes relying on the dual-system encryption methodology. However, their schemes do not imply efficiently revocable inner product functional encryption schemes as the overhead cost depends on the width of the policy of ciphertext.

The result in this part is under submission.

Organization

The rest of this thesis is organized as follows. In Chapter 2, we specify some notations that will be used throughout this thesis, and recall definitions for some hard problems and intractability assumptions on which the security of our constructions relies. In Chapter 3, we give our anonymous broadcast encryption for bounded universe (AnoBEB) from lattices. Then, in Chapter 4, we present our Trace & Revoke scheme as an application of AnoBEB and the robust-IPP code. In Chapter 5, we introduce the notion of traceable functional encryption, and we also provide an instantiation for the inner product case. It is secure under BDDH assumption. Finally, in Chapter 6, we describe recent unpublished results regarding constructions of revocable inner product functional encryption from various standard asumptions before concluding in Chapter 7.

2 Preliminaries

This chapter provides some formal definitions as well as backgrounds that will be used in the forthcoming chapters.

Contents

2.1 Notations

In this section, we define some general notations that are used in this thesis.

Let $\mathbb{N}$, $\mathbb{R}$ denote the set of all natural numbers $\{0, 1, 2, \ldots\}$ and the set of reals number, respectively. For $n \in \mathbb{N}$ we define $[n] = \{1, 2, \ldots, n\}$, $[0, 1] = \{x \in \mathbb{R} \mid 0 \leq x \leq 1\}$. $\mathbb{Z}_N$ denotes the additive group of integers modulo N as well as the set $\{0, \ldots, N-1\}$. $\mathbb{Z}_N^*$ denotes the multiplicative group of invertible integers modulo N (i.e., those that are relatively prime to N). We denote vectors by using either bold lower-case letters or lower-case letters with an arrow over it as $\mathbf{v}$ and $\vec{v}$. A matrix is usually denoted by a capital letter and sometimes we denote a matrix by a bold capital letter. The transpose of a matrix (or vector), say matrix A, is denoted by A^T. We will treat a vector as a column vector. Namely, for any vector $\vec{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_q^n$. $g^{\vec{v}}$ stands for the vector of group elements $(g^{v_1}, \ldots, g^{v_n})^T \in \mathbb{G}^n$. For $\vec{a}, \vec{z} \in \mathbb{Z}_q^n$, we denote their inner product as

$$\langle \vec{a}, \vec{z} \rangle = \vec{a}^T \vec{z} = \sum_{i=1}^{n} a_i z_i.$$

Given $g^{\vec{a}}$ and $\vec{z}$, $\left(g^{\vec{a}}\right)^{\vec{z}} := g^{\langle \vec{a}, \vec{z} \rangle}$. Let PPT stand for "Probabilistic Polynomial Time".

Definition 2.1: Negligible function

A negligible function is a function $\nu : \mathbb{N} \to [0, 1]$ such that

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

A function is non-negligible if it is not a negligible function. It means that ν would satisfy the following:

$$\exists c \in \mathbb{N}, \forall n_0 \in \mathbb{N}, \exists n \geq n_0 : \nu(n) \geq n^{-c}.$$

Definition 2.2: Negligible probability and Hard problem

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

A problem is said to be hard if there exists no PPT algorithm solving it with non-negligible probability.

Assume that D_1 and D_2 are distributions over a countable set X, their statistical distance is defined to be $\frac{1}{2} \sum_{x \in X} |D_1(x) - D_2(x)|$. We say that two distributions D_1 and D_2 (two ensembles of distributions indexed by n) are statistically close if their statistical distance is negligible in n. We use the notation $x \hookleftarrow D$ to refer that the element x is sampled from the distribution D. We also let $U(X)$ denote the uniform distribution over X.

For two matrices A, B of compatible dimensions, let $(A \| B)$ (or sometimes $\begin{pmatrix} A \\ B \end{pmatrix}$) denote vertical concatenations of A and B.

For $A \in \mathbb{Z}_q^{m \times n}$, define $\mathrm{Im}(A) = \{A\mathbf{s} \mid \mathbf{s} \in \mathbb{Z}_q^n\} \subseteq \mathbb{Z}_q^m$. For $X \subseteq \mathbb{Z}_q^m$, let $\mathrm{Span}(X)$ denote the set of all linear combinations of elements of X and define $X^\perp$ to be $\{\mathbf{b} \in \mathbb{Z}_q^m \mid \forall \mathbf{c} \in X, \langle \mathbf{b}, \mathbf{c} \rangle = 0\}$.

2.2 Standard Assumptions

2.2.1 Prime order group assumptions

Let $\mathbb{G}$ be a group (written multiplicatively) of prime order q and let g be a generator of $\mathbb{G}$.

Definition 2.3: Discrete Logarithm Assumption

The Discrete Logarithm hypothesis states that given an element $h \in \mathbb{G}$ it is hard to find $\mu \in \mathbb{Z}_q$ such that $h = g^\mu$.

Definition 2.4: Decisional Diffie-Hellman Assumption

The Decision Diffie Hellman (DDH) problem consists in distinguishing the following distributions

$$\mathcal{D}_0 = \{(g^a, g^b, g^{ab}) \mid a, b \xleftarrow{\$} \mathbb{Z}_q\} \qquad \mathcal{D}_1 = \{(g^a, g^b, g^c) \mid a, b, c \xleftarrow{\$} \mathbb{Z}_q\}.$$

The distribution $\mathcal{D}_0$ consists of Diffie-Hellman (DH) tuples whereas $\mathcal{D}_1$ consists of random tuples. Roughly speaking, the DDH problem consists in distinguishing DH tuples from random tuples. The DDH assumption states that the two above distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ are indistinguishable.

2.2.2 Pairing group assumptions

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be multiplicatively written groups of prime order q, and let g_1, g_2 be generators of $\mathbb{G}_1, \mathbb{G}_2$, respectively. We write 1_T to denote the unit element of $\mathbb{G}_T$. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a function sending two elements from $\mathbb{G}_1$ and $\mathbb{G}_2$ into the group $\mathbb{G}_T$. We say that the tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e)$ is a bilinear group if the following properties hold:

- **Bilinearity**: for all $h_1 \in \mathbb{G}_1, h_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_q^*$, we have $e(h_1^a, h_2^b) = e(h_1, h_2)^{ab}$.
- **Non-degeneracy**: $e(g_1, g_2) \neq 1_T$.
- The function e can be efficiently computed.

Bilinear groups in which $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$ are called *symmetric bilinear groups* (type 1). In this case, we denote it $(\mathbb{G}, \mathbb{G}_T, q, e)$ and we assume that $g = g_1 = g_2$. If $\mathbb{G}_1 \neq \mathbb{G}_2$, it is called an asymmetric bilinear group.

- An asymmetric bilinear group is called type 2 if there is an efficiently computable homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$.

- An asymmetric bilinear group is called type 3 if there is no efficiently computable homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$.

Decisional Linear Assumption (DLIN): The decisional linear problem is defined as follows. We choose a group $\mathbb{G}$ of prime order p. We choose random generators g, f, ν of $\mathbb{G}$ and random exponents $c_1, c_2 \in \mathbb{Z}_p$. If the attacker is given

$$P = \{g, f, \nu, g^{c_1}, f^{c_2}\},$$

it must remain hard to distinguish $\nu^{c_1+c_2}$ from a random element of $\mathbb{G}$.

An algorithm $\mathcal{B}$ that outputs $z \in \{0, 1\}$ has advantage ϵ in solving the decisional linear problem in $\mathbb{G}$ if

$$\left| \Pr\left[\mathcal{B}\left(P, T = \nu^{c_1+c_2}\right) = 0\right] - \Pr[\mathcal{B}(P, T = R) = 0] \right| \geq \epsilon.$$

Definition 2.5: Decisional Linear Assumption

We say the decisional linear assumption holds if no poly-time algorithm (PPT) has a non-negligible advantage in solving the decisional linear problem.

q-Decisional Multi-Exponent Bilinear Diffie-Hellman Assumption (MEBDH): Let $\mathbb{G}$ be a bilinear group of prime order q. The q-MEBDH problem in $\mathbb{G}$ is stated as follows:

A challenger picks a generator $g \in \mathbb{G}$ and random exponents $r, \alpha, a_1, \ldots, a_q$. The attacker is then given

$$P = \left\{ \begin{array}{ll} & g, g^r, e(g,g)^\alpha \\ \forall_{1 \leq i,j \leq q} & g^{a_i}, g^{a_i r}, g^{a_i a_j}, g^{\alpha/a_i^2} \\ \forall_{1 \leq i,j,k \leq q, i \neq j} & g^{a_i a_j r}, g^{\alpha a_j/a_i^2}, g^{\alpha a_i a_j/a_k^2}, g^{\alpha a_i^2/a_j^2} \end{array} \right\}$$

it must remain hard to distinguish $e(g,g)^{\alpha \cdot r} \in \mathbb{G}_T$ from a random element in $\mathbb{G}_T$.

An algorithm $\mathcal{B}$ that outputs $z \in \{0, 1\}$ has advantage ϵ in solving q-decisional MEBDH in $\mathbb{G}$ if

$$\left| \Pr\left[\mathcal{B}\left(P, T = e(g,g)^{\alpha r}\right) = 0\right] - \Pr[\mathcal{B}(P, T = R) = 0] \right| \geq \epsilon.$$

Definition 2.6

We say that the q -decisional Multi-Exponent Bilinear Diffie-Hellman assumption holds if no poly-time algorithm has non-negligible advantage in solving the q-MEBDH problem.

Definition 2.7: Bilinear Decisional Diffie-Hellman Assumption

Given an asymmetric bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e)$, the Bilinear Decisional Diffie-Hellman (BDDH) problem consists in distinguishing the following distri-

butions, for generators g_1 and g_2

$$\begin{aligned}\mathcal{D}_0 &= \left\{\left(g_1^a, g_1^b, g_2^a, g_2^c, e\left(g_1, g_2\right)^{abc}\right) | a, b, c \xleftarrow{\$} \mathbb{Z}_q\right\} \\ \mathcal{D}_1 &= \left\{\left(g_1^a, g_1^b, g_2^a, g_2^c, e\left(g_1, g_2\right)^{z}\right) | a, b, c, z \xleftarrow{\$} \mathbb{Z}_q\right\}.\end{aligned}$$

The BDDH assumption states that no PPT adversary can distinguish $\mathcal{D}_0$ and $\mathcal{D}_1$ with non negligible advantage.

2.2.3 Hardness assumption of $k-$LWE

This section presents the problem, which is a variant of the learning with errors (LWE, for short) problem [Reg05] (proposed by Regev) called $k-$LWE. The quantity k is the number of secret keys leaked or corrupted by an adversary. This problem was introduced by Ling et al. at CRYPTO '14. In their paper [LPSS14], they showed that there is a reduction from LWE to $k-$LWE with a polynomial loss in k.

Let $\mathbf{B}$ consist of n linearly independent vectors $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\}$. The n-dimensional lattice Λ generated by the basis $\mathbf{B}$ is $\Lambda = L(\mathbf{B}) = \{\mathbf{Bc} = \sum_{i\in[n]} c_i \cdot \mathbf{b}_i \mid \mathbf{c} \in \mathbb{Z}^n\}$. The length of a matrix $\mathbf{B}$ is defined as the norm of its longest column: $\|\mathbf{B}\| = \max_{1\leq i\leq n} \|\mathbf{b}_i\|$. Here we view a matrix as simply the set of its column vectors.

For a lattice $L \subseteq \mathbb{R}^m$ and an invertible matrix $S \in \mathbb{R}^{m\times m}$, we define the Gaussian distribution of parameters L and S by $D_{L,S}(\mathbf{b}) = \exp(-\pi\|S^{-1}\mathbf{b}\|^2)$ for all $\mathbf{b} \in L$.

The q-ary lattice associated with a matrix $A \in \mathbb{Z}_q^{m\times n}$ is defined as $\Lambda^{\perp}(A) = \{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{x}^t \cdot A = \mathbf{0} \bmod q\}$. It has dimension m, and a basis can be computed in polynomial-time from A. For $\mathbf{u} \in \mathbb{Z}_q^m$, we define $\Lambda_{\mathbf{u}}^{\perp}(A)$ as the coset $\{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{x}^t \cdot A = \mathbf{u}^t \bmod q\}$ of $\Lambda^{\perp}(A)$.

Lemma 2.1: Theorem 3.1, [AP11]

There is a probabilistic polynomial-time algorithm that, on input positive integers $n, m, q \geq 2$, outputs two matrices $A \in \mathbb{Z}_q^{m\times n}$ and $T \in \mathbb{Z}^{m\times m}$ such that the distribution of A is within statistical distance $2^{-\Omega(n)}$ from $U(\mathbb{Z}_q^{m\times n})$; the rows of T form a basis of $\Lambda^{\perp}(A)$; each row of T has norm $\leq 3mq^{n/m}$.

Lemma 2.2: GPV algorithm, [GPV08]

There exists a probabilistic polynomial-time algorithm that given a basis $\mathbf{B}$ of an n-dimensional lattice $\Lambda = L(\mathbf{B})$, a parameter $s \geq \|\widetilde{\mathbf{B}}\| \cdot \omega\left(\sqrt{\log n}\right)$ (where $\widetilde{\mathbf{B}}$ is Gram-Schmidt orthogonalization of $\mathbf{B}$), outputs a sample from a distribution that is statistically close to $D_{\Lambda,s}$.

The Learning With Errors (LWE) problem was introduced by Regev [Reg05]. This problem is one of the most known problems in lattice-based cryptography, and it is used to construct many cryptosystems.

Definition 2.8: LWE problem, [Reg05]

Let $m \geq n \geq 1, q \geq 2$ and $\alpha \in (0, 1)$. The LWE problem consists in

distinguishing between the distributions $(\mathbf{A}, \mathbf{As}+\mathbf{e})$ and $U\left(\mathbb{Z}_q^{m\times n}\times\mathbb{Z}_q^m\right)$, where $\mathbf{A}\hookleftarrow U\left(\mathbb{Z}_q^{m\times n}\right), \mathbf{s}\hookleftarrow U\left(\mathbb{Z}_q^n\right)$ and $\mathbf{e}\hookleftarrow D_{\mathbb{Z}^m,\alpha q}$. For an algorithm $\mathcal{A}:\mathbb{Z}_q^{m\times n}\times\mathbb{Z}_q^m\to\{0,1\}$, we define:

$$\mathsf{Adv}^{\mathsf{LWE}}_{q,m,n,\alpha}(\mathcal{A}) = \left|\mathsf{Pr}[\mathcal{A}(\mathbf{A},\mathbf{As}+\mathbf{e})=1]-\mathsf{Pr}[\mathcal{A}(\mathbf{A},\mathbf{u})=1]\right|$$

where the probabilities are over $\mathbf{A}\hookleftarrow U\left(\mathbb{Z}_q^{m\times n}\right), \mathbf{s}\hookleftarrow U\left(\mathbb{Z}_q^n\right), \mathbf{u}\hookleftarrow U\left(\mathbb{Z}_q^m\right)$ and $\mathbf{e}\hookleftarrow D_{\mathbb{Z}^m,\alpha q}$ and the internal randomness of $\mathcal{A}$. We say that $\mathsf{LWE}_{q,m,n,\alpha}$ is hard if for all PPT algorithm $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{LWE}}_{q,m,n,\alpha}(\mathcal{A})$ is negligible.

The following problem, which is proposed by Ling et al. [LPSS14], is a variant problem of LWE. The LWE problem asks an adversary $\mathcal{A}$ (which is given the matrix $\mathbf{A}$) to distinguish the distribution $\mathbf{As}+\mathbf{e}$ and the uniform distribution over $\mathbb{Z}_q^m$. In $k-\mathsf{LWE}$ problem, given the matrix $\mathbf{A}$ and some extra information (k small hints $\mathbf{x}_i$ such that $\mathbf{x}_i^t\mathbf{A}=\mathbf{0}$), the adversary $\mathcal{A}$ also is asked to distinguish the distribution $\mathbf{As}+\mathbf{e}$ and the uniform distribution in orthogonal span of $\mathbf{x}_i$ plus some noises. Formally, it is defined as follows:

Definition 2.9: k-LWE problem, [LPSS14]

Let $S\in\mathbb{R}^{m\times m}$ be an invertible matrix and denote $\mathbb{T}^{m+1}=(\mathbb{R}/\mathbb{Z})^{m+1}$. The $(k,S)-\mathsf{LWE}$ problem is: given $A\hookleftarrow U(\mathbb{Z}_q^{m\times n})$, $\mathbf{u}\hookleftarrow U(\mathbb{Z}_q^n)$ and $\mathbf{x}_i\hookleftarrow D_{\Lambda^\perp_{-\mathbf{u}}(A),S}$ for $i\le k\le m$, the goal is to distinguish between the distributions (over $\mathbb{T}^{m+1}$)

$$\frac{1}{q}\cdot U\left(\mathrm{Im}\left(\frac{\mathbf{u}^t}{A}\right)\right)+\nu_\alpha^{m+1} \quad\text{and}\quad \frac{1}{q}\cdot U\left(\mathrm{Span}_{i\le k}(1\|\mathbf{x}_i)^\perp\right)+\nu_\alpha^{m+1},$$

where ν_α denotes the one-dimensional Gaussian distribution with standard deviation $\alpha>0$.

In [LPSS14], it was shown that this problem can be reduced to the LWE problem for a specific class of diagonal matrices S. In our work, we only need any such S where (k,S)-LWE is hard, and thus the use of S is implicit. For simplicity, we will use k-LWE and (k,S)-LWE interchangeably in this thesis.

Projective Sampling

Inspired by the notion of projective hash family [CS02], Ling et al. [LPSS14] proposed a new concept called projective sampling family. A construction of projective sampling family from $k-\mathsf{LWE}$ problem was built as well. The major purpose of their construction is to switch a secret key traitor tracing scheme into a public key one, where tracing signals are sampled from a distribution of spanned spaces by secret keys $\mathbf{x}_j$. In their scheme, each secret key $\mathbf{x}_j\in\mathbb{Z}_q^m$ is associated with a public matrix H_j (projective key). Given the projective keys H_j, any entity in the system can simulate the tracing signal in a computationally indistinguishable way (under the k-LWE assumption) in the sense that the simulated signal $U(\cap_j\mathrm{Im}(H_j))$ is indistinguishable from the original tracing signal $U\left(\mathrm{Span}_j(\mathbf{x}_j^+)^\perp\right)$ even for entities who know the secret keys $\mathbf{x}_j$. This implies that anyone in the system is allowed to execute the tracing procedure.

We recall the construction of H_j [LPSS14] as follows:

1. Given a matrix $A \in \mathbb{Z}_q^{m\times n}$ and an invertible matrix $A \in \mathbb{Z}_q^{m\times m}$, sampling signals are taken from a spanned space $U\left(\mathrm{Span}_{j\leq k}(\mathbf{x}_j^+)^\perp\right) + \lfloor\nu_{\alpha q}\rceil^{m+1}$, where $\mathbf{x}_j \hookleftarrow D_{\Lambda^\perp_{-\mathbf{u}}(A),S}$. We call vectors $\mathbf{x}_j \in \mathbb{Z}_q^m$ secret keys.

2. Sample $H \hookleftarrow U\left(\mathbb{Z}_q^{m\times(m-n)}\right)$, conditioned on $\mathrm{Im}(H) \subset \mathrm{Im}(A)$. Define the public projected value of $\mathbf{x}_j$ on H as $\mathbf{h}_j = -H^t \cdot \mathbf{x}_j$.

3. Define $H_j = (\mathbf{h}_j^t \parallel H) \in \mathbb{Z}_q^{(m+1)\times(m-n)}$ as the public projected key of $\mathbf{x}_j$.

Simulated signals are now sampled from the distribution $U(\cap_{j\leq k}\mathrm{Im}(H_j)) + \lfloor\nu_{\alpha q}\rceil^{m+1}$. Under the (k,S)-LWE hardness assumptions, the following two distributions:

$$U\left(\mathrm{Span}_{j\leq k}(\mathbf{x}_j^+)^\perp\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} \text{ and } U\left(\cap_{j\leq k}\mathrm{Im}(H_j)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1}$$

are indistinguishable. This implies that given projected keys H_j, anyone can take samples from the distribution $U\left(\mathrm{Span}_{j\leq k}(\mathbf{x}_j^+)^\perp\right) + \lfloor\nu_{\alpha q}\rceil^{m+1}$ although he does not have the secret keys $\mathbf{x}_j$.

We restate an important result that is frequently used in our proofs. This result comes directly from Theorem 25 and Theorem 27 in [LPSS14].

Lemma 2.3

We denote by $[t] = \{1,\ldots,t\}$ the set of the t first positive integers. Under the k-LWE assumption, for $k > t$, given t secret keys $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t$, for any $j \notin [t]$, the distrisbutions

$$U\left(\mathrm{Span}_{i\in[t]}(\mathbf{x}_i^+)^\perp\right) + \lfloor\nu_{\alpha q}\rceil^{m+1},\ U\left(\mathrm{Span}_{i\in[t]\cup\{j\}}(\mathbf{x}_i^+)^\perp\right) + \lfloor\nu_{\alpha q}\rceil^{m+1},$$

are indistinguishable (from Theorem 25 in [LPSS14]), and the distributions

$$U\left(\cap_{i\in[t]}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1},\ U\left(\cap_{i\in[t]\cup\{j\}}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1},$$

are indistinguishable as well (from Theorem 27 in [LPSS14]).

3 Anonymous Broadcast Encryption for Bounded Universe

Privacy is a primary ingredient for our digital life. There are many cryptographic primitives that have been designed to address privacy in authentication (blind signatures or anonymous credentials) or in confidentiality (homomorphic encryption, randomizable encryption), to name a few. When cryptographic protocols are deployed for emerging multi-user applications, the demand for privacy becomes more and more important. The focus of this chapter is to build an anonymous broadcast encryption scheme for bounded universe (AnoBEB). Barth, Boneh, and Waters [BBW06] and Libert, Paterson, and Quaglia [LPQ12] have shown the best solution for an anonymous broadcast encryption scheme from public-key encryption (PKE). However, their generic constructions for N users suffer a linear factor N in the ciphertext size. If we consider a relaxed condition where the adversary is the outsider who has no key in the target set, the best construction is of Fazio and Perera [FP12] in which the ciphertext size in $O(\mathsf{log}N)$. Kiayias and Samari [KS12] stated that achieving a sub-linear ciphertext in N is impossible. If we consider the case where the number of users is bounded, the problem of constructing an efficient broadcast encryption system to preserve user privacy remains challenging.

In this chapter, we provide a non-blackbox construction with optimal rate (=1) for the case when the number of users N is bounded by the security parameter. In the domain of anonymous broadcast encryption, this is the first construction that achieves the same efficiency as the underlying (LWE) public-key encryption. We will show in Chapter 4 that this scheme in bounded universe can be used to achieve a Trace & Revoke scheme in the general case where the number of users is not bounded. Our observation is that we can switch the traitor tracing scheme in [LPSS14] into an AnoBEB scheme.

Contents

3.1 Definitions

3.1.1 Broadcast Encryption

Broadcast encryption is a cryptographic primitive designed to efficiently distribute an encrypted content via a public channel to a designated set of users so that only privileged users can decrypt while the other users cannot learn anything about the content. We follow the definition in [BGW05].

Definition 3.1

Let $\mathcal{PT}$ and $\mathcal{CT}$ denote the plaintext and ciphertext spaces, respectively. A broadcast encryption scheme consists of algorithms (Setup, $\mathsf{Extract}$, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$) defined as below

$\mathsf{Setup}(1^n, N)$**:** Takes as input the security parameter n, it generates the global parameters param of the system, including N the maximal number of users (receivers are implicitly represented by integers in a universe of users $\mathcal{U} = \{1, \ldots, N\}$), and outputs a master public key ek and a master secret key MSK.

$\mathsf{Extract}(\mathsf{ek}, \mathsf{MSK}, i)$**:** Take as input the public key ek, the master secret key MSK and a user index $i \in \mathcal{U}$, the algorithm extracts the decryption key dk_i which is sent to the user i.

$\mathsf{Encrypt}(\mathsf{ek}, m, S)$**:** Take as input the public key ek, a message $m \in \mathcal{PT}$ and a set of previleged users $S \subseteq \mathcal{U}$, outputs a ciphertext $c \in \mathcal{CT}$, which is broadcasted to every member of S.

$\mathsf{Decrypt}(\mathsf{ek}, \mathsf{dk}_i, c, S)$**:** Take as input the public key ek, the decryption key dk_i of user i, a ciphertext $c \in \mathcal{CT}$ and the set of receivers $S \subseteq \mathcal{U}$. If $i \in S$, the algorithm outputs a message $m \in \mathcal{PT}$ or an invalid symbol $\perp$.

The correctness requirement is that, with overwhelming probability over the randomness used by the algorithms, we have:

$$\forall M \in \mathcal{PT}, \forall i \in \mathcal{S} : \mathsf{Decrypt}(\mathsf{ek}, \mathsf{dk}_i, \mathsf{Encrypt}(\mathsf{ek}, M, \mathcal{S})) = M.$$

Security of a broadcast encryption scheme:

Definition 3.2

The `CPA` security of a BE scheme Π is defined based on the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{B}$

- The challenger runs $\mathsf{Setup}(1^n, N)$ and gives the produced public key ek to the adversary $\mathcal{A}$.
- The adversary (adaptively) chooses indices $i \in \mathcal{U}$ to ask decryption keys. The challenger gives $\mathcal{A}$ all the dk_i for all required indices.
- The adversary then chooses two messages $M_0, M_1 \in \mathcal{PT}$ of equal length

and a set $\mathcal{S} \subset \mathcal{U}$ of users with restriction that no index $i \in \mathcal{S}$ required decryption key before. It then gives M_0, M_1 and $\mathcal{S}$ to the challenger.

- The challenger samples $b \hookleftarrow \{0,1\}$ and provides $c \hookleftarrow \mathsf{Encrypt}(\mathsf{ek}, M_b, \mathcal{S})$ to $\mathcal{A}$.
- The adversary $\mathcal{A}$ continues asking for decryption keys for any index i outside $\mathcal{S}$.
- Finally, the adversary returns its guess $b' \in \{0,1\}$ for the b. The adversary wins this game if $b = b'$.

We define $\mathsf{Succ}^{\mathtt{IND}}(\mathcal{A}) = \Pr[b' = b]$, the probability that $\mathcal{A}$ wins the game. We say that Π is semantically secure ($\mathtt{IND}$) if all polynomial time adaptive adversaries $\mathcal{A}$ have at most negligible advantage in the above game, where $\mathcal{A}$'s advantage is defined as

$$\mathsf{Adv}^{\mathtt{IND}}(\mathcal{A}) = |\mathsf{Succ}^{\mathtt{IND}}(\mathcal{A}) - \frac{1}{2}| = |\Pr[b' = b] - \frac{1}{2}|.$$

3.1.2 Anonymous Broadcast Encryption

A broadcast encryption scheme is called anonymous (AnoBE for short) if it allows addressing a message to a subset of the users, without revealing this privileged set even to users who successfully decrypt the message. We follow the definition in [LPQ12]:

Definition 3.3

Let $\mathcal{PT}$ and $\mathcal{CT}$ denote the plaintext and ciphertext spaces, respectively. Let $\mathcal{U} = \{1, \ldots, N\}$ be the universe of users. An anonymous broadcast encryption (AnoBE) consists of the following algorithms:

$\mathsf{Setup}(1^n, N)$**:** Takes as input the security parameter n and the maximal number of users N. It outputs a public key ek and a master secret key MSK.

$\mathsf{Extract}(\mathsf{ek}, \mathsf{MSK}, i)$**:** Takes as input the public key ek, the master secret key MSK and a user index $i \in \mathcal{U}$, the algorithm extracts the decryption keys dk_i which is sent to the corresponding user i.

$\mathsf{Encrypt}(\mathsf{ek}, M, \mathcal{S})$**:** Takes as input the public key ek, a message $M \in \mathcal{PT}$ and a set of target users $\mathcal{S} \subset \mathcal{U}$, outputs a ciphertext $c \in \mathcal{CT}$.

$\mathsf{Decrypt}(\mathsf{ek}, \mathsf{dk}_i, c)$**:** Takes as input the public key ek, the decryption key dk_i of user i and a ciphertext $c \in \mathcal{CT}$. The algorithm outputs the message $M \in \mathcal{PT}$ or an invalid symbol $\perp$.

The correctness requirement is that, with overwhelming probability over the randomness used by the algorithms, we have:

$$\forall M \in \mathcal{PT}, \forall i \in \mathcal{S} : \mathsf{Decrypt}(\mathsf{ek}, \mathsf{dk}_i, \mathsf{Encrypt}(\mathsf{ek}, M, \mathcal{S})) = M.$$

When the number of users N in the scheme is bounded by the security parameter (where $N \leq k$ for some k bounded by a security parameter n), we have the notion of *anonymous broadcast encryption for bounded universe* – AnoBEB.

Security of an anonymous broadcast encryption scheme:

Definition 3.4

The security of an AnoBEB scheme Π is defined based on the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{B}$. The challenger $\mathcal{B}$ runs $\mathsf{Setup}(1^n, N)$ to obtain a public key ek and a master secret key MSK and sends ek to adversary $\mathcal{A}$.

Phase 1. The adversary $\mathcal{A}$ adaptively issues decryption key extraction queries for any index $i \in \mathcal{U}$. The challenger runs $\mathsf{Extract}$ algorithm on index i and returns to $\mathcal{A}$ the decryption key $\mathsf{dk}_i = \mathsf{Extract}(\mathsf{ek}, \mathsf{MSK}, i)$.

Challenger. The adversary chooses a message $M \in \mathcal{PT}$ and two distinct subsets $\mathcal{S}_0, \mathcal{S}_1 \subset \mathcal{U}$ of users. We require that $\mathcal{A}$ has not issued key queries for any index $i \in \mathcal{S}_0 \triangle \mathcal{S}_1 = (\mathcal{S}_0 \setminus \mathcal{S}_1) \cup (\mathcal{S}_1 \setminus \mathcal{S}_0)$. The adversary $\mathcal{A}$ passes M and $\mathcal{S}_0, \mathcal{S}_1$ to the challenger $\mathcal{B}$. The challenger $\mathcal{B}$ randomly chooses a bit $b \in \{0, 1\}$, computes $c = \mathsf{Encrypt}(\mathsf{ek}, M, \mathcal{S}_b)$ and sends c to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ adaptively issues decryption key extraction queries on indices $i \notin \mathcal{S}_0 \triangle \mathcal{S}_1$ and obtains decryption keys dk_i.

Guess. The adversary outputs a guess $b' \in \{0, 1\}$ and wins the game if $b' = b$.

We denote by $\mathsf{Succ}^{\mathsf{ANO}}(\mathcal{A}) = \Pr[b' = b]$ the probability that $\mathcal{A}$ wins the game, and its advantage is

$$\mathsf{Adv}^{\mathsf{ANO}}(\mathcal{A}) = |\mathsf{Succ}^{\mathsf{ANO}}(\mathcal{A}) - \frac{1}{2}| = |\Pr[b' = b] - \frac{1}{2}|.$$

We say that a scheme Π is *anonymous against chosen plaintext attacks* – ANO if all polynomial-time adversaries $\mathcal{A}$ have a negligible advantage in the above game.

3.2 A Construction based on the Learning with Errors Assumption

Ling et al. [LPSS14] introduced the first lattice-based traitor tracing scheme based on the $k-\mathsf{LWE}$ assumption. They showed a polynomial-time reduction from $k-\mathsf{LWE}$ to LWE, so their scheme is as efficient as the LWE encryption. A natural question is why one cannot directly rely on their scheme to design an anonymous revoke or broadcast encryption scheme. Revoking users is a very difficult task and the following simple question is still open: for a constant number of revoked users, can we design a revoke scheme that is comparably efficient to the underlying encryption. Based on $k-\mathsf{LWE}$, it seems very hard, because for revocation, essentially one need to find a vector that is "orthogonal" to all the secret vectors of the non-revoked

users (so that they get the same message) and this is impossible for a large universe system. Now, concerning broadcast encryption, whenever relying on $k-\mathsf{LWE}$, one cannot allow the adversary to corrupt more than k-users, where $k \leq m$ is bounded by the underlying lattice dimension. Therefore, at best, one can aim at an anonymous broadcast encryption for a small universe.

The below construction of a AnoBEB scheme comes from a basic "tweaking purpose" idea: switching the tracing procedure in [LPSS14] to be functional as a broadcast encryption. We first recall that in the LPSS traitor tracing scheme, the linear tracing technique [CFN94] was applied: to detect a traitor in a group of suspect users, they first create a ciphertext so that every user in this group can decrypt successfully the ciphertext. In the subsequent steps, the tracer will disable, one by one, users in the group, preventing them from decrypting the ciphertext. We observe that if we switch the suspected users in LPSS scheme to be the legitimate users, and the removed users in the suspected set to the revoked users, then we get a broadcast encryption. Because the LPSS traitor tracing can deal with a bounded number (less than the dimension of the underlying lattice) of traitors, we also get a broadcast encryption for a bounded number of users, that we call broadcast encryption for bounded universe.

We now consider a construction in detail an anonymous broadcast encryption for bounded universe scheme (AnoBEB) from k-LWE problem.

Let N be the maximal number of users (receivers are implicitly represented by integers in $\mathcal{U} = \{1, \ldots, N\}$). Given a security parameter n, we assert that parameters q, m, α, S are chosen so that the (k, S)-LWE problem is hard to solve as presented in [LPSS14]. Since the adversary can corrupt any user, we require that $N \leq k$ (the system's bounded universe constraint).

Setup$(1^n, N)$**:** Takes as input the security parameter n and maximal number of users N. It uses Lemma 2.1 to generate 2 matrices $(A, T) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}^{m \times m}$ and picks $\mathbf{u}$ uniformly in $\mathbb{Z}_q^n$. We set a master secret key $\mathsf{MSK} = (A, T)$ and a public key $\mathsf{ek} = \{A^+, (H_j)_{j \leq N}\}$, where $A^+ = (\mathbf{u}^t \| A)$ and the projected keys H_j (corresponding to the secret keys $\mathbf{x}_j$, defined in Section 2.2.3) are added each time a secret key $\mathbf{x}_j$ is generated by the Extract. For a system of N users, one can run N times Extract inside the Setup to generate N secret keys.

Extract$(\mathsf{ek}, \mathsf{MSK}, j)$**:** Takes as input the public key ek, the master secret key MSK and a user index $j \in \mathcal{U}$, the algorithm calls the GPV algorithm (Lemma 2.2) using the basis $\Lambda^{\perp}(A)$ consisting of the rows of T and the standard deviation matrix S. It obtains a sample $\mathbf{x}_j$ from $D_{\Lambda^{\perp}_{-\mathbf{u}}(A), S}$. The algorithm outputs decryption key $\mathsf{dk}_j = \mathbf{x}_j^+ := (1 \| \mathbf{x}_j) \in \mathbb{Z}^{m+1}$ for user j.

Encrypt$(\mathsf{ek}, M, \mathcal{S})$**:** Takes as input the public key ek, a message $M \in \mathcal{PT} = \{0, 1\}$ and a set of users $\mathcal{S} \subseteq \mathcal{U}$. To encrypt M, one chooses a vector $\mathbf{y} \in \mathbb{Z}_q^{m+1}$ from the distribution $U(\cap_{i \in S} \mathrm{Im}(H_i))$, $\mathbf{e} \hookleftarrow \lfloor \nu_{\alpha q} \rceil^{m+1}$ and outputs $\mathbf{c} \in \mathcal{CT}$, which is broadcasted to every member of $\mathcal{S}$ as follows:

$$\mathbf{c} = \mathbf{y} + \mathbf{e} + \begin{pmatrix} M \lfloor q/2 \rfloor \\ \mathbf{0} \end{pmatrix},$$

whereas $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x.

$\mathsf{Decrypt}(\mathsf{ek}, \mathsf{dk}_j, \mathbf{c})$: Takes as input the public key ek, a decryption key $\mathsf{dk}_j = \mathbf{x}_j^+$ of user j and a ciphertext $\mathbf{c} \in \mathcal{CT}$. The function $\mathsf{Decrypt}$ will return 0 if $\langle \mathbf{x}_j^+, \mathbf{c} \rangle$ is closer 0 than to $\lfloor q/2 \rfloor$ modulo q, otherwise return 1.

Correctness. We require that for a given subset $\mathcal{S} \subseteq \mathcal{U}$ and all $j \in \mathcal{S}$, if $\mathbf{c} = \mathsf{Encrypt}(\mathsf{ek}, m, \mathcal{S})$ and dk_j is the decryption key for user $j \in \mathcal{S}$, we then recover $M = \mathsf{Decrypt}(\mathsf{ek}, \mathsf{dk}_j, \mathbf{c})$ with overwhelming probability. Indeed, since $\cap_{i \in \mathcal{S}} \mathrm{Im}(H_i) \subseteq \mathrm{Span}_{i \in \mathcal{S}}(\mathbf{x}_i^+)^\perp$, for each user $j \in \mathcal{S}$ and $\mathbf{y} \hookleftarrow U(\cap_{i \in \mathcal{S}} \mathrm{Im}(H_i))$, we have $\langle \mathbf{x}_j^+, \mathbf{y} \rangle = 0$. Therefore,

$$\begin{aligned} \langle \mathbf{x}_j^+, \mathbf{c} \rangle &= \langle \mathbf{x}_j^+, \mathbf{y} \rangle + \langle \mathbf{x}_j^+, \mathbf{e} \rangle + \langle \mathbf{x}_j^+, \begin{pmatrix} M\lfloor q/2 \rfloor \\ 0 \end{pmatrix} \rangle \mod q \\ &= \langle \mathbf{x}_j^+, \mathbf{e} \rangle + M\lfloor q/2 \rfloor \mod q, \end{aligned}$$

where $\mathbf{e} \hookleftarrow \lfloor \nu_{\alpha q} \rceil^{m+1}$. According to [LPSS14], the quantity $\langle \mathbf{x}_j^+, \mathbf{e} \rangle$ is relatively small modulo q with overwhelming probability. The procedure $\mathsf{Decrypt}$ returns the original message with overwhelming probability. Therefore, every user in $\mathcal{S}$ can decrypt successfully. We now consider the security of the scheme, essentially showing that an adversary which is allowed to corrupt any user outside $\mathcal{S}$, cannot break the semantic security of the scheme.

Theorem 3.1

Under the k-LWE assumption, for any $N \leq k$, the AnoBEB scheme Π constructed as above is $\mathtt{IND}$-secure.

Proof

We consider the sequence of the following games between a challenger $\mathcal{B}$ and an attacker $\mathcal{A}$.

Game G_0**:** This is the real world game, security as defined in the security model. The interaction between the challenger $\mathcal{B}$ and the adversary $\mathcal{A}$ takes place as follows:

Setup. The challenger generates matrix $A \hookleftarrow U(\mathbb{Z}_q^{m \times n})$ and $\mathbf{u} \hookleftarrow U(\mathbb{Z}_q^n)$. The challenger sends public key $\mathsf{ek} = \{A^+, (H_j)_{j \leq N}\}$, where each H_j is the projected key associated with a secret key $\mathbf{x}_j$ and $A^+ = (\mathbf{u}^t \| A)$. The public key then sent to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ queries decryption keys for several users $i \in \{1, \ldots, N\}$. $\mathcal{B}$ samples $\mathbf{x}_i \hookleftarrow D_{\Lambda_{-\mathbf{u}}^\perp(A), S}$ and gives $\mathbf{x}_i^+$ to $\mathcal{A}$, where $\mathbf{x}_i^+ := (1 \| \mathbf{x}_i) \in \mathbb{Z}^{m+1}$.

Challenger phase. The adversary selects two messages $M_0, M_1 \leftarrow \mathcal{PT} = \{0, 1\}$, a subset of users $\mathcal{S} \subset \mathcal{U}$ so that queried indices must be outside $\mathcal{S}$. $\mathcal{A}$ then sends M_0, M_1 and $\mathcal{S}$ to $\mathcal{B}$. The challenger picks at random a bit $b \hookleftarrow U(\{0, 1\})$, outputs a challenge ciphertext (of the message M_b) sampled from one of two following distributions:

$$\begin{aligned} \mathcal{D}_0 &= U\left(\cap_{i \in \mathcal{S}} \mathrm{Im}(H_i)\right) + \lfloor \nu_{\alpha q} \rceil^{m+1} + \begin{pmatrix} M_0\lfloor q/2 \rfloor \\ 0 \end{pmatrix}, \\ \mathcal{D}_1 &= U\left(\cap_{i \in \mathcal{S}} \mathrm{Im}(H_i)\right) + \lfloor \nu_{\alpha q} \rceil^{m+1} + \begin{pmatrix} M_1\lfloor q/2 \rfloor \\ 0 \end{pmatrix}. \end{aligned}$$

Phase 2. The adversary continues querying for decryption keys with the limiting condition that $\mathcal{A}$ only queries indices outside $\mathcal{S}$.
Guess. $\mathcal{A}$ gives a guess b' for b.

Game G_1**:** The challenger now makes one small change to the previous game. Namely, every steps in this game coincides with a corresponding step in the previous one, but the challenge ciphertext sampled from one of two distributions $\mathcal{D}_0^1$ and $\mathcal{D}_1^1$.

$$\begin{aligned}\mathcal{D}_0^1 &= U\left(\cap_{i\in\mathcal{S}\setminus\{j\}}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M_0\lfloor q/2\rfloor \\ 0\end{pmatrix},\\ \mathcal{D}_1^1 &= U\left(\cap_{i\in\mathcal{S}\setminus\{j\}}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M_1\lfloor q/2\rfloor \\ 0\end{pmatrix},\end{aligned}$$

whereas $j \in \mathcal{S}$. Applying Lemma 2.3, within the view of $\mathcal{A}$, there are two pairs of distributions

$$\begin{aligned}\mathcal{D}_0 &= U\left(\cap_{i\in\mathcal{S}}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M_0\lfloor q/2\rfloor \\ 0\end{pmatrix},\\ \mathcal{D}_0^1 &= U\left(\cap_{i\in\mathcal{S}\setminus\{j\}}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M_0\lfloor q/2\rfloor \\ 0\end{pmatrix}\end{aligned}$$

and

$$\begin{aligned}\mathcal{D}_1 &= U\left(\cap_{i\in\mathcal{S}}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M_1\lfloor q/2\rfloor \\ 0\end{pmatrix},\\ \mathcal{D}_1^1 &= U\left(\cap_{i\in\mathcal{S}\setminus\{j\}}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M_1\lfloor q/2\rfloor \\ 0\end{pmatrix}\end{aligned}$$

are indistinguishable under the assumption that k-LWE is hard to solve. Therefore, the difference of the advantage of the adversary $\mathcal{A}$ in the two consecutive games is negligible.

Similarly, we consider extra $\ell - 1$ games, where $\ell = |\mathcal{S}|$ and reach the final game.

Game G_ℓ**:** The challenger also makes one small change to the previous games, while every step in this game coincides with the previous one, but for the challenge ciphertext sampled from one of two distributions $\mathcal{D}_0^\ell$ and $\mathcal{D}_1^\ell$, as follows:

$$\begin{aligned}\mathcal{D}_0^\ell &= U\left(\mathbb{Z}_q^{m+1}\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M_0\lfloor q/2\rfloor \\ 0\end{pmatrix},\\ \mathcal{D}_1^\ell &= U\left(\mathbb{Z}_q^{m+1}\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M_1\lfloor q/2\rfloor \\ 0\end{pmatrix}.\end{aligned}$$

Obviously, the advantage of $\mathcal{A}$ in this game is equal to zero.

To summarize, we have a sequence of games where the final game **Game** G_ℓ has zero-advantage and the difference of each two successive games **Game** $\mathrm{G}_{\mathrm{i}-1}$, **Game** G_i, for all $2 \le i \le \ell$, is negligible, and ℓ is polynomial. Therefore, the scheme Π is IND−secure. ■

We next consider anonymity of the AnoBEB scheme (our main Theorem for this

section):

Theorem 3.2

Under the k-LWE assumption, for any $N \leq k$, the AnoBEB scheme is ANO-secure.

Proof

Intuitively, the anonymity requires that an adversary cannot distinguish between encryptions for two targets $\mathcal{S}_0, \mathcal{S}_1$ of its choice. If we consider an outsider adversary, defined in [FP12], which only corrupts users outside both $\mathcal{S}_0, \mathcal{S}_1$, then the proof is direct because from the k-LWE assumption, the encryption for $\mathcal{S}_0$ and for $\mathcal{S}_1$, both, look like random ciphertexts to the adversary. It is more challenging to consider a general adversary which can also corrupt the key in the intersection of $\mathcal{S}_0$ and $\mathcal{S}_1$. Fortunately, by applying Lemma 2.3 which informally states that the encryptions for a set $\mathcal{S}$ and for a set $\mathcal{S} \cup \{i\}$ are indistinguishable if the adversary does not corrupt the user i, even if the adversary corrupts users in $\mathcal{S}$. We then apply a hybrid argument which moves an encryption for the set $\mathcal{S}_0$ (or $\mathcal{S}_1$) to an encryption for the set $\mathcal{S}_0 \cup \mathcal{S}_1$ by adding one by one users in $\mathcal{S}_1 \setminus \mathcal{S}_0$ (or in $\mathcal{S}_1 \setminus \mathcal{S}_0$, respectively).

We will prove the above by considering a sequence of games, as following:

Game G_0**:** This is the real world game, security defined in the security model. We repeat the interaction between the challenger $\mathcal{B}$ and the adversary $\mathcal{A}$ as following:

Setup. The challenger generates a matrix $A \hookleftarrow U(\mathbb{Z}_q^{m \times n})$ and picks $\mathbf{u}$ uniformly in $\mathbb{Z}_q^n$. Then the public key is set to $\mathsf{ek} = \{A^+, (H_j)_{j \leq k}\}$, with $A^+ = (\mathbf{u}^t \| A)$, and given to $\mathcal{A}$.

Phase 1. When $\mathcal{A}$ asks for the decryption key for user i, $\mathcal{B}$ replies with $\mathbf{x}_i^+ = (1||\mathbf{x}_i)$, where $\mathbf{x}_i \hookleftarrow D_{\Lambda_{-\mathbf{u}}^{\perp}(A),S}$.

Challenger phase. $\mathcal{A}$ chooses a message M, two subsets $\mathcal{S}_0, \mathcal{S}_1$ with the restriction that no asked query is in $\mathcal{U} \setminus (\mathcal{S}_0 \triangle \mathcal{S}_1)$ and sends it to $\mathcal{B}$. The challenger picks randomly $b \in \{0, 1\}$ and gives $\mathcal{A}$ a ciphertext $\mathbf{c}$ taken from one of two distributions (distribution $\mathcal{D}_b$, over $\mathbb{T}^{m+1}$):

$$\mathcal{D}_0 = U\left(\cap_{i \in \mathcal{S}_0} \mathrm{Im}(H_i)\right) + \lfloor \nu_{\alpha q} \rceil^{m+1} + \begin{pmatrix} M\lfloor q/2 \rfloor \\ 0 \end{pmatrix},$$

$$\mathcal{D}_1 = U\left(\cap_{i \in \mathcal{S}_1} \mathrm{Im}(H_i)\right) + \lfloor \nu_{\alpha q} \rceil^{m+1} + \begin{pmatrix} M\lfloor q/2 \rfloor \\ 0 \end{pmatrix}.$$

Phase 2. In this step, $\mathcal{A}$ continues querying to get decryption keys with the limitations as mentioned before (query indices $i \in \left(\mathcal{U} \setminus (\mathcal{S}_0 \triangle \mathcal{S}_1)\right)$). $\mathcal{B}$ gets $\mathbf{x}_i^+$ from $D_{\Lambda_{-\mathbf{u}}^{\perp}(A),S}$ and answers $\mathcal{A}$.

Guess. $\mathcal{A}$ guesses b' for b.

Game G_1**:** In this game, the inputs and the settings of this game are identical to the ones of **Game** G_0. In the challenger phase, the adversary $\mathcal{A}$ received a

ciphertext from one of the two following distributions:

$$\begin{aligned}\mathcal{D}_0 &= U\left(\cap_{i\in\mathcal{S}_0}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M\lfloor q/2\rfloor \\ 0 \end{pmatrix},\\ \mathcal{D}_1^1 &= U\left(\cap_{i\in\mathcal{S}_1\cup\{j_1\}}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M\lfloor q/2\rfloor \\ 0 \end{pmatrix},\end{aligned}$$

where the projected key H_{j_1} corresponds to the secret key $\mathbf{x}_{j_1}^+ \hookleftarrow D_{\Lambda^\perp_{-\mathbf{u}}(A),S}$, $j_1 \in \mathcal{S}_0 \setminus \mathcal{S}_1$.

Here we notice that the adversary $\mathcal{A}$ does not know the key $\mathbf{x}_{j_1}^+$ because $\mathcal{A}$ can only choose the keys with index in $\mathcal{U} \setminus (\mathcal{S}_0 \triangle \mathcal{S}_1)$. Since k-LWE is hard, by applying Lemma 2.3, the two distributions

$$\begin{aligned}\mathcal{D}_1 &= U\left(\cap_{i\in\mathcal{S}_1}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M\lfloor q/2\rfloor \\ 0 \end{pmatrix},\\ \mathcal{D}_1^1 &= U\left(\cap_{i\in\mathcal{S}_1\cup\{j_1\}}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M\lfloor q/2\rfloor \\ 0 \end{pmatrix}\end{aligned}$$

are indistinguishable. This means that the difference between the advantage of $\mathcal{A}$ in **Game** G_1 and **Game** G_0 is negligible.

Game G_τ**:** We assume that $\kappa = |\mathcal{S}_0 \setminus \mathcal{S}_1|$ and $\mathcal{S}_0 \setminus \mathcal{S}_1 = \{j_1, j_2, \ldots, j_\kappa\}$. For each $2 \le \tau \le \kappa$, we consider a game in a sequence of $\kappa - 1$ games. We set $\mathcal{T}_1 = \mathcal{S}_1 \cup \{j_1\}$ and $\mathcal{T}_\tau = \mathcal{T}_{\tau-1} \cup \{j_\tau\}$. It implies that $\mathcal{T}_\kappa = \mathcal{S}_0 \cup \mathcal{S}_1$. In each game in this sequence, the inputs and the settings are identical to the ones of previous games. In the challenger phase, the adversary $\mathcal{A}$ receives a ciphertext from one of the two following distributions:

$$\begin{aligned}\mathcal{D}_0 &= U\left(\cap_{i\in\mathcal{S}_0}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M\lfloor q/2\rfloor \\ 0 \end{pmatrix},\\ \mathcal{D}_1^\tau &= U\left(\cap_{i\in\mathcal{T}_\tau}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M\lfloor q/2\rfloor \\ 0 \end{pmatrix}.\end{aligned}$$

Since adversary $\mathcal{A}$ does not know any key $\mathbf{x}_{j_\tau}^+$ in the set $\mathcal{S}_0 \setminus \mathcal{S}_1$ and the $k-$LWE problem is hard, we apply Lemma 2.3, the two distributions:

$$\begin{aligned}\mathcal{D}_1^{\tau-1} &= U\left(\cap_{i\in\mathcal{T}_{\tau-1}}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M\lfloor q/2\rfloor \\ 0 \end{pmatrix},\\ \mathcal{D}_1^\tau &= U\left(\cap_{i\in\mathcal{T}_\tau}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M\lfloor q/2\rfloor \\ 0 \end{pmatrix},\end{aligned}$$

are indistinguishable for each τ. This means that the difference between the advantage of $\mathcal{A}$ in any transition in the sequence of games **Game** G_τ, $1 \le \tau \le \kappa$ is negligible.

Game $\mathrm{G}_{\kappa+\eta}$**:** We assume that $\iota = |\mathcal{S}_1 \setminus \mathcal{S}_0|$ and $\mathcal{S}_1 \setminus \mathcal{S}_0 = \{j_1, j_2, \ldots, j_\iota\}$. For each $1 \le \eta \le \iota$, we consider a game in a sequence of ι games. We set $\mathcal{T}'_1 = \mathcal{S}_0 \cup \{j_1\}$ and $\mathcal{T}'_\eta = \mathcal{T}'_{\eta-1} \cup \{j_\eta\}$. It implies that $\mathcal{T}'_\iota = \mathcal{S}_0 \cup \mathcal{S}_1$. In each game in this sequence, the inputs and the settings are identical to the ones of previous games. In challenger phase, the adversary $\mathcal{A}$ receives a ciphertext from one of following

two distributions:

$$\mathcal{D}_0^{\eta} = U\left(\cap_{i\in\mathcal{T'}_{\eta}}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M\lfloor q/2\rfloor \\ 0 \end{pmatrix},$$

$$\mathcal{D}_1^{\kappa} = U\left(\cap_{i\in(\mathcal{S}_0\cup\mathcal{S}_1)}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M\lfloor q/2\rfloor \\ 0 \end{pmatrix}.$$

It means that we keep fix the distribution $\mathcal{D}_1^{\kappa}$ and replace the distribution $\mathcal{D}_0$ by

$$\mathcal{D}_0^{\eta} = U\left(\cap_{i\in\mathcal{T'}_{\eta}}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M\lfloor q/2\rfloor \\ 0 \end{pmatrix},$$

where we set $\mathcal{D}_0^{\eta} = \mathcal{D}_0$ in case $\eta = 0$. By the same argument as in previous games, in the view of the adversary $\mathcal{A}$, two distributions $\mathcal{D}_0^{\eta-1}$ and $\mathcal{D}_0^{\eta}$ are indistinguishable under the hardness of k-LWE, this means that the two following distributions

$$\mathcal{D}_0^{\eta-1} = U\left(\cap_{i\in\mathcal{T'}_{\eta-1}}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M\lfloor q/2\rfloor \\ 0 \end{pmatrix},$$

$$\mathcal{D}_0^{\eta} = U\left(\cap_{i\in\mathcal{T'}_{\eta}}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M\lfloor q/2\rfloor \\ 0 \end{pmatrix}$$

are indistinguishable for each $1 \leq \eta \leq \iota$. Therefore the difference between the advantage of $\mathcal{A}$ in the transitions of the sequence of games **Game** $\mathrm{G}_{\eta+\kappa}$, $1 \leq \eta \leq \iota$ is negligible. We recall that in the last game ($\eta = \iota$), $\mathcal{A}$ will receive a challenger ciphertext taken from

$$U\left(\cap_{i\in(\mathcal{S}_0\cup\mathcal{S}_1)}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M\lfloor q/2\rfloor \\ 0 \end{pmatrix},$$

$$U\left(\cap_{i\in(\mathcal{S}_0\cup\mathcal{S}_1)}\mathrm{Im}(H_i)\right) + \lfloor\nu_{\alpha q}\rceil^{m+1} + \begin{pmatrix} M\lfloor q/2\rfloor \\ 0 \end{pmatrix}.$$

Obviously, the advantage of adversary $\mathcal{A}$ in this game is equal to zero since these distributions are identical.

We conclude (as all sequences are polynomial size) that our scheme AnoBEB is ANO-secure under the hardness of k-LWE problem. ■

3.3 Efficiency of AnoBEB

Concerning efficiency, our scheme AnoBEB is exactly as efficient as the Ling *et al.*'s traitor tracing scheme in [LPSS14] which was shown in [LPSS14] to be as efficient as the standard LWE encryption.

Finally, we also note that, as shown in [LPSS14], example parameters are $k = m/10$, $\sigma = \widetilde{\Theta}(n)$, $q = \widetilde{\Theta}(n^5)$ and $m = \Theta(n \log n)$. We can therefore set our parameters to: $N = k$ and the efficiency of the AnoBEB scheme is approximately as efficient as the underlying LWE-PKE, inherently from the fact the LPSS k-LWE traitor tracing has approximately the same efficiency as the underlying LWE-PKE, as shown in [LPSS14].

4 Trace & Revoke Scheme from AnoBEB

Designing a Trace & Revoke system is a challenge problem. A Trace & Revoke system is not merely combining a traitor tracing system with a broadcast encryption system, as shown in the paper of Boneh and Waters in [BW06], building a Trace & Revoke from traceability and revocability system is very difficult to achieve.

This chapter is devoted to constructing a Trace & Revoke (TR) protocol from a robust IPP code and the AnoBEB which is presented at the previous chapter. We also give two explicit constructions of robust IPP code that are suitable for the Trace & Revoke system.

Contents

4.1 Definitions

4.1.1 Intuition

Boneh and Waters [BW06] have shown that it is very difficult to achieve a Trace & Revoke from traceability and revocability system by giving an example as follows. We consider a Trace & Revoke scheme TR for N users by combining a traitor tracing scheme TT and a revocation scheme R. In order to encrypt a message M, the content distributor first splits the message M into two components M_1 and M_2 and then the message M_1 is encrypted using TT system and the message M_2 is encrypted using R system. A user with the decryption key sk_i can recover the message M if it can be able to decrypt correctly under both schemes TT and R. If there is an Adversary $\mathcal{A}$ who corrupts two users P_1 and P_2, it builds a pirate decoder $\mathcal{D}$ with a decryption strategy as follows: it uses the decryption key of P_1 to decrypt the ciphertext which is generated by R and uses the decryption key of P_2 to decrypt the ciphertext which is generated by TT . When the tracer runs Tracing algorithm, it finds that P_2 is guilty. Clearly, P_2 should be removed from the system. Therefore, the scheme TR is insecure because the pirate decoder $\mathcal{D}$ still works fine.

4.1.2 Trace & Revoke Systems

Adapted from the definition of the Trace & Revoke system in [AKPS12], we will see that our AnoBEB is a Trace & Revoke system in the black-box model, namely a BE system with a black-box tracing algorithm. A Trace & Revoke (TR) system, in turn, consists of algorithms as follows:

$\mathsf{Setup}(\lambda, N)$**:** Takes as input the security parameter λ, maximal number of user N and a maximum malicious coalition size t. It outputs the global parameters param of the system, a public key ek, a master secret key MSK and a tracing key TK, where TK is public.

$\mathsf{Extract}(\mathsf{ek}, \mathsf{MSK}, i)$**:** Takes as input the public key ek, the master secret key MSK and a user index $i \in \mathcal{U}$, the algorithm extracts the decryption keys dk_i which is sent to the corresponding user i.

$\mathsf{Encrypt}(\mathsf{ek}, M, \mathcal{S})$**:** Takes as input the public key ek, a message $M \in \mathcal{PT}$ and a set of target users $\mathcal{S} \subset \mathcal{U} = \{1, 2, \ldots, N\}$, outputs a ciphertext $c \in \mathcal{CT}$. Remark that we do not impose any restriction on the target set $\mathcal{S}$ and therefore we can use $\mathsf{Encrypt}$ as a revoke algorithm: to revoke a set of revoked users $\mathcal{R} \subset \mathcal{U}$, we simply set the target set $\mathcal{S}$ to $\mathcal{U} \setminus \mathcal{R}$. It is assumed that the revoked user set is managed centrally by a revocation authority.

$\mathsf{Decrypt}(\mathsf{ek}, \mathsf{dk}_i, c)$**:** Takes as input the public key ek, the decryption key dk_i of user i and a ciphertext $c \in \mathcal{CT}$. The algorithm outputs the message $M \in \mathcal{PT}$ or an invalid symbol $\perp$.

$\mathsf{Tracing}(\mathcal{D}, \mathsf{ek}, \mathsf{TK})$**:** It takes as input the public key ek, the tracing key TK and has access to a pirate decoder $\mathcal{D}$. The tracing algorithm outputs the identity of at least one user that participated in building $\mathcal{D}$.

In this chapter, we shall assume that pirate devices are resettable, meaning that they do not maintain state during the tracing process. Moreover, we also assume that

the tracing procedure is being considered in the minimal black-box access model. It means that the tracer has access to $\mathcal{D}$ by using an oracle $\mathcal{O}^{\mathcal{D}}$. The oracle $\mathcal{O}^{\mathcal{D}}$ will be fed the input which has the form $(c, M) \in (\mathcal{CT}, \mathcal{PT})$. The tracer will get 1 from the output $\mathcal{O}^{\mathcal{D}}$ in the case that the decoder decrypts correctly the ciphertext c, i.e. $\mathcal{D}(c) = M$ and will get 0 in the other case. We require that the pirate device $\mathcal{D}$ decrypts correctly with a non-negligible probability (ε).

$$\Pr_{\substack{M \leftarrow\!\!\$\, U(\mathcal{PT}) \\ c \leftarrow\!\!\$\, \mathsf{Encrypt}(M)}} \left[\mathcal{O}^{\mathcal{D}}(c, M) = 1\right] \geq \varepsilon = \frac{1}{|\mathcal{PT}|} + \frac{1}{\lambda^c},$$

for some constant $c > 0$.

The traceability is defined via the following game between an attacker $\mathcal{A}$ and a challenger $\mathcal{B}$:

Tracing Game

1. The adversary $\mathcal{A}$ outputs a set $\mathcal{T} = \{u_1, u_2, \ldots, u_t\} \subset \{1, \ldots, N\}$ of colluding users.

2. The challenger $\mathcal{B}$ runs $\mathsf{Setup}(\lambda, N)$ and sends all public keys ek, tracing key TK and decryption keys $\mathsf{dk}_{u_1}, \ldots, \mathsf{dk}_{u_t}$ to the adversary $\mathcal{A}$.

3. The adversary $\mathcal{A}$ creates a resettable pirate decoder $\mathcal{D}$ so that the pirate decoder decrypts correctly the ciphertexts with at least ε.

4. The challenger $\mathcal{B}$ executes the procedure $\mathsf{Tracing}(\mathcal{D}, \mathsf{ek}, \mathsf{TK})$ and outputs a set $\mathcal{L} \subset \mathcal{T}$ that is accused.

We say that the adversary $\mathcal{A}$ wins the game if the set $\mathcal{L}$ is either empty, or is not a subset of $\mathcal{T}$ and $\Pi = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Tracing})$ is a TR scheme with tracing success probability α against t-coalition ε-pirates if no polynomial time attacker $\mathcal{A}$ can win the game described above with probability more than $1 - \alpha$. We denote by $\mathsf{Adv}_{\mathsf{TR}}$ the probability that adversary $\mathcal{A}$ wins this game.

We remake that any Anonymous Broadcast Encryption is also a TR system. Indeed, it is relatively straightforward to see that revocation is implied directly from the semantic security of BE system (since the sender has freedom of choosing the receiver set and will not include revoked users). About traceability of Anonymous BE systems, we can apply the linear tracing technique for the Anonymous BE system to capture this ability. We do not present arguments in the general case. Instead, we will apply directly this technique in the AnoBEB context.

1. For $i = 0$ to t, do the following:

   (a) Let $\mathsf{cnt} \leftarrow 0$.

   (b) Repeat the following steps $W \leftarrow 8\lambda(t/\varepsilon)^2$ times:

      i. $M \leftarrow\!\!\$\, U(\mathcal{PT})$

      ii. $c \leftarrow\!\!\$\, U\left(\mathrm{Span}(\vec{x}_1^+, \ldots, \vec{x}_i^+)^\perp\right) + \lfloor \nu_{\alpha q} \rceil^{m+1} + \begin{pmatrix} M\lfloor q/2 \rfloor \\ 0 \end{pmatrix}$

      iii. Call oracle $\mathcal{O}^{\mathcal{D}}$ on input c and if $\mathcal{O}^{\mathcal{D}}(c, M) = 1$ then $\mathsf{cnt} \leftarrow \mathsf{cnt} + 1$

   (c) Let $\widetilde{p}_i$ be the fraction of times that $\mathcal{D}$ decrypted the ciphertexts correctly. We have $\widetilde{p}_i = \mathsf{cnt}/W$.

2. Let $\mathcal{L}$ be the set of all $i \in \{1, \ldots, t\}$ for which $\widetilde{p}_i - \widetilde{p}_{i-1} \geq \varepsilon/4t$.

3. Output the set $\mathcal{L}$ as the set of guilty colluders.

We define p_i as the probability the pirate decoder $\mathcal{D}$ decrypts correctly the ciphertext for $i \in [0, t]$.

$$p_i \;=\; \Pr_{\substack{\vec{c} \leftarrow Tr_i \\ M \leftarrow U(\{0,1\})}} \left[\mathcal{O}^{\mathcal{D}} \left(c + \left[\frac{M \cdot \lfloor q/2 \rfloor}{\vec{0}} \right], M \right) = 1 \right],$$

where

$$Tr_i = U\left(\mathrm{Span}(\vec{x}_1^{+}, \ldots, \vec{x}_i^{+})^{\perp}\right) + \lfloor \nu_{\alpha q} \rceil^{m+1}.$$

A gap between p_{i-1} and p_i is meant to indicate that u_i is a traitor.

We will compute an approximation $\widetilde{p}_i$ of the probabilities p_i using $W = 8\lambda(t/\varepsilon)^2$ samples. Applying the additive form of the Chernoff bound $\Pr\left[|p - \widetilde{p}| > \varepsilon\right] < 2e^{-W\varepsilon^2}$, we have

$$\begin{aligned} \Pr\left[|p_i - \widetilde{p}_i| > \varepsilon/(16t)\right] &< 2e^{-2W(\varepsilon/(16t))^2} \\ &= 2e^{-2\cdot 8\lambda(t/\varepsilon)^2(\varepsilon/(16t))^2} \\ &= 2e^{-\lambda/16} < 2e^{-\lambda/64}, \end{aligned}$$

which is negligible in the security parameter λ for all $i = 1, \ldots, t$. Therefore, we may assume from here on that $|p_i - \widetilde{p}_i| \leq \varepsilon/(16t)$ for all $i = 1, \ldots, t$.

The confirmation and soundness properties of the tracing algorithm then follows directly from Theorems 24, 25 in [LPSS14].

However, public tracing is more challenging. Fortunately, our system also supports public traceability. It comes directly from the result in [LPSS14]. We thus achieve a Public TR scheme for a Bounded Universe.

4.1.3 Robust Identifying Parent Property codes

Before recalling the formal definition of Robust IPP code, we visit the notation of IPP code. Let Σ be an alphabet set containing q symbols.

If $\mathcal{C} = \{w_1, \ldots, w_N\} \subset \Sigma^{\ell}$ then $\mathcal{C}$ is called a $q-$ary code of size N and length ℓ. Each $w_i \in \mathcal{C}$ is called a codeword and we write $w_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,\ell})$ where $w_{i,j} \in \Sigma$ is called the $j-$th component of the codeword w_i. We denote by (ℓ, N, q) a code wof size N, length ℓ, and over an alphabet size q. Let Δ denote the minimum Hamming distance of the code $\mathcal{C}$.

Given a positive integer t, a subset of codewords $X = \{w_1, w_2, \ldots, w_t\} \subset \mathcal{C}$ is called a coalition of size t. Let $X_i = \{w_{1,i}, w_{2,i}, \ldots, w_{t,i}\}$ be the set of the $i-$th coordinates of the coalition X. If the cardinality of X_i is equal to 1, say $|X_i| = 1$, the coordinate i is called undetectable, else it is called detectable. The set of detectable coordinates for the coalition X is denoted by $D(X)$. The set of descendants of X, denoted $\mathsf{desc}(X)$, is defined by

$$\mathsf{desc}(X) = \left\{ x = (x_1, \ldots, x_\ell) \in \Sigma^{\ell} \mid x_j \in X_j, 1 \leq j \leq \ell \right\},$$

codewords in the coalition X is called parents of the set $\mathsf{desc}(X)$. Define a $t-$descendant ($t-$envelope) of the code $\mathcal{C}$, denoted $\mathsf{desc}_t(\mathcal{C})$, as follows:

$$\mathsf{desc}_t(\mathcal{C}) = \bigcup_{X \subset \mathcal{C}, |X| \leq t} \mathsf{desc}(X).$$

The $\mathsf{desc}_t(\mathcal{C})$ consists of all $\ell-$tuples that could be generated by some coalition of size at most t. Codes with identifiable parent property (IPP codes) are defined next.

Definition 4.1

Given a code $\mathcal{C} = (\ell, N, q)$, let $t \geq 2$ be an integer. The code $\mathcal{C}$ is called a $t-$IPP code if for all $x \in \mathsf{desc}_t(\mathcal{C})$, it holds that

$$\bigcap_{x \in \mathsf{desc}(X), X \subset \mathcal{C}, |X| \leq t} X \neq \emptyset.$$

Then, in a $t-$IPP code, given a descendant $x \in \mathsf{desc}_t(\mathcal{C})$, we can always identify at least one of its parent codewords error-free.

In [BS95], Boneh and Shaw considered a more general coalition, called wide-sense envelope of the coalition X. The set of descendants in their fingerprinting code is defined as follows:

$$\mathsf{desc}(X) = \Big\{ x = (x_1, \ldots, x_\ell) \in \Sigma^\ell \cup \{*\} \mid x_j \in X_j, j \notin D(X) \Big\},$$

where $D(X)$ consists of detectable coordinates of the coalition X. This means that any symbol of Σ or erased symbols $*$ are allowed in the detectable coordinates. Only detectable coordinates of descendant are allowed to modify the values by following the *marking assumption*. The notation Robust IPP code is an intermediate concept between the IPP and fingerprinting codes in the sense that robust IPP codes allow a limited number of coordinates to not follow their parents. These coordinates are allowed to deviate by breaking the marking assumption.

Let $X \subset \Sigma^\ell, |X| \leq t$ be a coalition. For $i = 1, \ldots, \ell$, let X_i be the set of the $i-$th coordinates of the elements of a coalition X. Assume that there is a descendant x in the set $\mathsf{desc}(X)$, following the marking assumption rule except εn coordinates that can deviate from this rule. Call a coordinate i of $x \in \mathsf{desc}(X)$ a mutation if $x_i \notin X_i$ and consider mutations of two types: erasures, where x_i is replaced by an erasure symbol $*$, and one replaced by an arbitrary symbol $y_i \in \Sigma - X_i$.

Denote by $\mathsf{desc}(X)_\varepsilon$ the set of all vectors x formed from the vectors in the coalition X so that $x_i \in X_i$ for $\ell(1-\varepsilon)$ coordinates i and x_i is a mutation in at most $\varepsilon\ell$ coordinates. Codes with robust identifiable parent property (Robust IPP codes) are defined below:

Definition 4.2

Code $\mathcal{C} \subset \Sigma^\ell$ is a $(t, \varepsilon) -$ IPP code (robust $t-$ IPP code) if

$$\bigcap_{X \subset \mathcal{C}, |X| \leq t, x \in \mathsf{desc}(X)_\varepsilon} X \neq \emptyset.$$

In words: the code $\mathcal{C}$ guarantees exact identification of at least one member of the coalition X of size at most t for any collusion with at most $\varepsilon\ell$ mutations. In the case $\varepsilon = 0$, the robust IPP becomes IPP code.

A robust IPP code is said to have the traceability property if for any $x \in \mathsf{desc}_\varepsilon(X)$, the codeword $c \in \mathcal{C}$ closest to x by the Hamming distance is always one of the parents of x, i.e.,

$$c \in \bigcap_{X \subset \mathcal{C}, |X| \leq t, x \in \mathsf{desc}(X)_\varepsilon} X.$$

This implies that a pirate can be provably identified by finding any vector $c \in \mathcal{C}$ such that the distance from c to x is shortest.

A robust IPP code with traceability property is called robust TA code. In this thesis, we shall use robust IPP with traceability property.

4.2 Construction

Overview.

We first consider code-based traitor tracing schemes. Combinatorial methods of designing a traitor tracing consist of two steps: first, construct a small scheme, then combine these schemes to achieve a general one. This method is proposed in the very first traitor tracing paper of Chor-Fiat-Naor [CFN94]. Kiayias and Yung [KY02] integrated a 2-user traitor tracing scheme with a collusion-secure code [BS95] into a TT scheme. This method can be summarized as follows: First, a 2-user traitor tracing scheme can be trivially obtained from applying a public-key encryption (PKE) twice, each for one user. Now, a message or a session key is divided into ℓ sub-keys. The sender then essentially encrypts each sub-key twice with PKE and gets sub-ciphertexts. Each recipient, provided sub-keys associated with a codeword of a collusion-secure code, can decrypt one of the two sub-ciphertexts for each sub-key and thus recover the whole message or session key which will be used to encrypt data.

Key assignment :	Table 0	$\mathsf{pk}_{0,1}$	$\mathsf{pk}_{0,2}$	$\mathsf{pk}_{0,3}$	$\mathsf{pk}_{0,4}$	$\mathsf{pk}_{0,5}$	...	$\mathsf{pk}_{0,\ell}$	
	Table 1	$\mathsf{pk}_{1,1}$	$\mathsf{pk}_{1,2}$	$\mathsf{pk}_{1,3}$	$\mathsf{pk}_{1,4}$	$\mathsf{pk}_{1,5}$	...	$\mathsf{pk}_{1,\ell}$	
	Codeword i	1	0	0	1	0	...	1	
	user i	$\mathsf{sk}_{1,1}$	$\mathsf{sk}_{0,2}$	$\mathsf{sk}_{0,3}$	$\mathsf{sk}_{1,4}$	$\mathsf{sk}_{0,5}$	...	$\mathsf{sk}_{1,\ell}$	
Encryption :	Session Key	$K_1 \oplus$	$K_2 \oplus$	$K_3 \oplus$	$K_4 \oplus$	$K_5 \oplus$	...	$\oplus K_\ell$	$= K$
	Ciphertext	$c_{0,1}$	$c_{0,2}$	$c_{0,3}$	$c_{0,4}$	$c_{0,5}$	...	$c_{0,\ell}$	
		$c_{1,1}$	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$c_{1,5}$	...	$c_{1,\ell}$	

Table 4.1: Traitor tracing with binary collusion secure code

Table 4.1 shows an example of a traitor tracing with binary collusion secure code. A legitimate user is assigned a codeword in the code. The authority will decompose a session key K into segments K_j according to the length ℓ of the code. In each sub-system, the segment of session key K_j will be encrypted twice alternately with public-keys $\mathsf{pk}_{0,j}$ or $\mathsf{pk}_{1,j}$. Each user i provided a secret-key $\mathsf{sk}_{0,j}$ or $\mathsf{sk}_{1,j}$ depending on the value of its codeword at position j. The user thus provided ℓ secret-keys and employs these secret-keys to recover the sub-session keys K_j, $j = 1, \ldots, \ell$ from one of two ciphertexts $c_{0,j}$ or $c_{1,j}$. Finally, the user combines them to obtain the original session key K. The tracing procedure consists of using the traceability in each 2-user scheme to extract a word associated with the pirate decoder. Thanks to the tracing capability of the collusion-secure code, one can then trace back one

of the traitors. This method is then generalized for $q-$ary IPP code by integrating it with a q-user traitor tracing scheme. The q-user traitor tracing scheme can also be obtained by applying q times PKE [PST06]. Now, if we have AnoBEB for q-user which is as efficient as the underlying PKE, we can save a factor q in cipher efficiency.

We next explain why it is difficult to get revocation with code-based schemes and how we can overcome the problem. We recall that the binary collusion secure code is well suitable for traitor tracing; its shortcoming is the incapacity of supporting revocation. In fact, to revoke a group of users, the authority has to disable the ability to decrypt with sub-keys in each position of the revoked group. In using the binary collusion secure code scenario, there are only two possibilities for sub-key of each position. Whenever the authority executes the revocation procedure, a large number of legitimate non-revoked users will be affected, and will not be able to decrypt anymore. A non-trivial remedy is for the designer of system to choose a code with big alphabet for example $q-$ary IPP code instead of a binary collusion secure code with alphabet size two. Revocation will decrease the number of valid slightly. Certainly, in this case, the possibility that legitimate users will be excluded off the system with revoked users must also be taken into account. A secret sharing scheme, in turn, is a mechanism that allows us to think about a solution: a legitimate user only needs to have a certain fraction of the sub-keys to be able to recover the original message. However, this reduced requirement gives an advantage to the pirates as well: they become stronger as they do not need to put all sub-keys in the pirate decoder; namely, they are permitted to delete sub-keys. The introduction of robust IPP of Barg et al. [BK13] which allows the identification of parents even if some positions are intentionally erased, allows for a tool to deal with the above problem. We propose a new generic method for designing a Trace & Revoke system from robust IPP codes and AnoBEB. As in the previous code-based method, the ciphertext size of the Trace & Revoke system is proportional to the length of the code and the ciphertext size of the AnoBEB.

4.2.1 Trace & Revoke scheme from AnoBEB and robust IPP code

Our aim is now to construct a Trace & Revoke (TR) scheme from AnoBEB. In our approach, we combine the $t-$ IPP robust code $\mathcal{C} = (\ell, N, q)$ with an AnoBEB scheme. It is described as follows:

Given a $q-$ary robust $t-$ IPP code $\mathcal{C} = (\ell, N, q)$ with length ℓ, size N, mimimum Hamming distance Δ over alphabet $\Sigma = \{1, \ldots, q\}$, we denote by $\mathcal{C} = \{w_1, \ldots, w_N\}$ the codewords of code $\mathcal{C}$ and $w_i = (w_{i,1}, \ldots, w_{i,\ell})$. Since $\mathcal{C}$ is a robust IPP code, each descendant $w \in \mathsf{desc}_\varepsilon(X), |X| \leq t$ will have at most $\varepsilon\ell$ erasure positions $*$.

We choose a $(\rho\ell, \ell)$-secret sharing scheme, where $\rho = 1-\varepsilon$ so that the non-revoked users can decrypt due to the secret sharing correctness. Let r be maximum number of revoked users. We require that the parameter r, with the purpose of revocation, ischosen so that

$$\Delta > \ell\Big(1 - \frac{1-\rho}{r}\Big). \tag{4.1}$$

We denote by $[N] = \{1, \ldots, N\}$ the set of N users. We define a *mixture* $S = (S_1, \ldots, S_\ell)$ over Σ^ℓ to be a sequence of ℓ subsets of Σ, i.e. $S_i \subseteq \Sigma$. Given a vector $\vec{\omega} = (\omega_1, \ldots, \omega_\ell) \in \Sigma^\ell$, the *agreement* between $\vec{\omega}$ and a mixture S is defined to be

the number of positions $i \in [\ell]$ for which $\omega_i \in S_i$:

$$\text{AGR}(\vec{\omega}, S) = \sum_{i=1}^{\ell} \mathbf{1}_{\omega_i \in S_i},$$

where $\mathbf{1}_{\omega_i \in S_i} = 1$ if $\omega_i \in S_i$ and $\mathbf{1}_{\omega_i \in S_i} = 0$ if otherwise.

We will construct a broadcast system Γ for the set $[N]$ as follows: we identify each user $i \in [N]$ with the codeword $w_i = (w_{i,1}, \ldots, w_{i,\ell})$ in $\mathcal{C}$, whereas $w_{i,j}$ is the j-th coordinate of the codeword $w_i \in \mathcal{C}$. By assigning each user i in Γ to a set with ℓ sub-keys, we have

$$\mathsf{dk}_i = (\mathsf{dk}_{1,w_{i,1}}, \ldots, \mathsf{dk}_{j,w_{i,j}}, \ldots, \mathsf{dk}_{\ell,w_{i,\ell}}).$$

In our system, at any coordinate component of the decryption key, we have at most q sub-keys. We consider a one-to-one correspondence between the set of q sub-keys and the set of decryption keys of q users in AnoBEB system. Consequently, to broadcast a message K to the set of N users, we decompose K into ℓ components (by using $(\rho\ell, \ell)-$secret sharing scheme) and we encrypt j^{th}-component with AnoBEB.

Formally, to build a broadcast system for N users, we concatenate ℓ instantiations of the scheme AnoBEB (for q users) according to an $q-$ary code $\mathcal{C}$. In particular, we will combine the code $\mathcal{C}$ with robust IPP code $\mathcal{C}$. Our construction consists of 4 algorithms: Setup, Extract, Encrypt and Decrypt.

Setup(n, N)**:** Takes as input the security parameter n and the size N of the code $\mathcal{C}$. By calling ℓ times the procedure AnoBEB.Setup(n, q), where ℓ is the length of the code $\mathcal{C}$, we obtain public keys ek_j and master secret keys MSK_j, $j = 1, \ldots, \ell$. We set $\mathsf{ek} = (\mathsf{ek}_1, \ldots, \mathsf{ek}_\ell)$ and $\mathsf{MSK} = (\mathsf{MSK}_1, \ldots, \mathsf{MSK}_\ell)$.

Extract$(\mathsf{ek}, \mathsf{MSK}, i)$**:** Takes as index $i \in [N]$ for each user, we use MSK to extract ℓ decryption keys for user i:

$$\mathsf{dk}_i = (\mathsf{sk}_{1,w_{i,1}}, \ldots, \mathsf{sk}_{j,w_{i,j}}, \ldots, \mathsf{sk}_{\ell,w_{i,\ell}}),$$

where $w_{i,j}$ is the value at position j of codeword w_i. Here,

$$\mathsf{sk}_{j,w_{i,j}} = \mathsf{AnoBEB.Extract}(\mathsf{ek}_j, \mathsf{MSK}_j, w_{i,j}) \in \mathbb{Z}^{m+1}, j \in [\ell].$$

Encrypt$(\mathsf{ek}, K, \mathcal{R})$**:** Takes as input a set of revoked users $\mathcal{R} \subset \mathcal{C}$, where the cardinality of $\mathcal{R}$ is at most r. The message K will be broadcasted to the target set $\mathcal{C} \setminus \mathcal{R}$. We call the procedure $\mathsf{Share}(\ell, \rho\ell, \ell)$ of $(\rho\ell, \ell)$-secret sharing scheme. The $\mathsf{Share}(\ell, \rho\ell, \ell)$ algorithm outputs a secret $K \in \mathcal{PT}^\ell$ and ℓ shares $K_1, \ldots, K_\ell$. At least $\rho\ell$ of the shares are needed to recover the message K. We broadcast using the following mixture

$$\mathcal{M} = (\mathcal{M}_1, \ldots, \mathcal{M}_\ell) = (\Sigma \setminus \mathcal{R}[1], \ldots, \Sigma \setminus \mathcal{R}[\ell]),$$

where $\mathcal{R}[j] = \cup_{i \in \mathcal{R}} w_{i,j}$. The ciphertext has the form

$$\begin{aligned} \vec{c} &= (c_1, \ldots, c_\ell) \in \mathcal{CT}^\ell, \\ &= \Big(\mathsf{AnoBEB.Encrypt}(\mathsf{ek}_1, K_1, \mathcal{M}_1), \ldots, \mathsf{AnoBEB.Encrypt}(\mathsf{ek}_\ell, K_\ell, \mathcal{M}_\ell)\Big), \end{aligned}$$

where each c_j is the output of the encryption algorithm AnoBEB.Encrypt with input K_j and $\mathcal{M}_j$. This means that each part of message, K_j is encrypted with keys $\{\mathsf{sk}_{j,w_{i,j}}\}_{i \notin \mathcal{R}}$.

$\mathsf{Decrypt}(\mathsf{ek}, \mathsf{dk}_i, \vec{c})$: Takes as input ciphertext $\vec{c} \in \mathcal{CT}^\ell$ and a decryption key dk_i of user i. For each $j \in [\ell]$, the user calls $\mathsf{AnoBEB.Decrypt}(\mathsf{ek}_j, \mathsf{sk}_{j,w_{i,j}}, c_j)$ of the AnoBEB scheme on sub-keys $\mathsf{sk}_{j,w_{i,j}}$ to obtain at least $\rho\ell$ values among the shared values K_j (as will be proved in the correctness). By calling the function $\mathsf{Combine}$ of the secret sharing scheme over pairs $\{(j, K_j)\}$, the user recovers the original message K.

$\mathsf{Tracing}$: We consider the mixture $\mathcal{M}$ as in $\mathsf{Encrypt}$ procedure. Let $\mathcal{T}$ be the subset of $\mathcal{U} \setminus \mathcal{R}$ with at most t elements (traitors). We assume that the tracing procedure will be considered in the minimal black-box access model [BF99]. In this model, the tracing authority has access to an oracle $\mathcal{O}^{\mathcal{D}}$ that itself internally uses $\mathcal{D}$. Oracle $\mathcal{O}^{\mathcal{D}}$ behaves as follows: it takes as input any pair $(\vec{c}, K) \in (\mathcal{CT}^\ell \times \mathcal{PT}^\ell)$ and returns 1 if $\mathcal{D}(\vec{c}) = K$ and 0 otherwise; the oracle only tells whether the decoder decrypts $\vec{c}$ to K or not. We assume that the pirate produces an ϵ-useful decoder $\mathcal{D}$ in the sense that it can decrypt any normal ciphertext in the form

$$\begin{aligned}\vec{c} &= (c_1, \ldots, c_\ell) \\ &= \Big(\mathsf{AnoBEB.Encrypt}(\mathsf{ek}_1, K_1, \mathcal{M}_1), \ldots, \mathsf{AnoBEB.Encrypt}(\mathsf{ek}_\ell, K_\ell, \mathcal{M}_\ell)\Big),\end{aligned}$$

with non-negligible probability (at least ϵ). We denote here $\mathcal{M}_j = \{j_\iota\}_{\iota \in Q}$, $Q \subseteq [q]$ or $\mathcal{M}_j = \emptyset$ for all $j = 1, \ldots, \ell$. We consider the tracing procedure as follows:

For $j = 1$ to ℓ, do the following:

1. While $\mathcal{M}_j \neq \emptyset$, do the following:
   (a) Let $\mathsf{cnt} \leftarrow 0$.
   (b) Repeat the following steps $W \leftarrow 8n(q/\epsilon)^2$ times:
      i. $c_j = \mathsf{AnoBEB.Encrypt}(\mathsf{ek}_j, K_j, \mathcal{M}_j)$.
      ii. Call oracle $\mathcal{O}^{\mathcal{D}}$ on input $\vec{c} = (c_1, \ldots, c_j, \ldots, c_\ell)$.
      If $\mathcal{O}^{\mathcal{D}}(\vec{c}, K) = 1$ then $\mathsf{cnt} \leftarrow \mathsf{cnt} + 1$.
   (c) Let $\widetilde{p}_{j,j_\iota}$ be the fraction of times that $\mathcal{D}$ decrypted the ciphertexts correctly. We have $\widetilde{p}_{j,j_\iota} = \mathsf{cnt}/W$.
   (d) $\mathcal{M}_j = \mathcal{M}_j \setminus \{j_\iota\}$.
2. If there exists an index $j_\iota \in \mathcal{M}_j$ for which $\widetilde{p}_{j,j_\iota} - \widetilde{p}_{j,j_{\iota'}} \geq \epsilon/4q\ell$ for all $j_{\iota'} \in \mathcal{M}_j$ then
   (a) the key j_ι is accused and $\omega_j = j_\iota$,
   (b) $c_j = \mathsf{AnoBEB.Encrypt}(\mathsf{ek}_j, K_j, \mathcal{M}_j)$
   else $c_j = \mathsf{random}$ and $\omega_j = *$.

End for.

From the pirate word $\vec{\omega} = (\omega_1, \ldots, \omega_\ell)$ found after the Loop finished, call tracing procedure in robust IPP code on input $\vec{\omega}$. The $\mathsf{Tracing}$ returns a traitor.

For the above $\mathsf{Tracing}$ algorithm, we note that the decryption probabilities of the pirate device do not change significantly in every iterations step because even if the tracer detects a non-negligible decryption probability of pirate decoder, it will reset the modified component to a normal component. After step 2, the tracer will find

out a letter of pirate word at position j. The value of a position is either a symbol in the alphabet or an erasure symbol.

We prove that the tracing algorithm returns at least $\rho\ell$ keys. Indeed, if the output of the algorithm provides $t < \rho\ell$ keys then the ciphertext in the final iteration step ℓ will appear as t normal components and the pirate device will be able to still correctly decrypt the ciphertext. This is a contradiction because in the setting of our system, by using $(\rho\ell, \ell)-$secret sharing scheme, it is impossible for any decoder device to decrypt the ciphertext with less than $\rho\ell$ normal components successfully. Therefore, our tracing algorithm will output at least $\rho\ell$ pirate keys. We thus get at the end of Step 2 a pirate word with $\rho\ell$ components without $*$. Since the scheme Γ employs robust IPP code $\mathcal{C}$, the tracer uses the property of robust IPP for the pirate word which was found from the black-box tracing to identify at least one user who contributed to building the pirate device.

As the tracing procedure in robust IPP code (like IPP code) does not require any secret information and we only use the $\mathsf{AnoBEB.Encrypt}$ to produce the tracing signals, the combined scheme Γ supports full public traceability.

Correctness. We consider the correctness of our TR system: for all users $i \in [N]$ and all messages K. Whenever user i, $i \notin \mathcal{R}$ is given a ciphertext $\vec{c}$, he can decrypt successfully. Indeed, since $\mathcal{C}$ is the code having the minimum Hamming distance that satisfies inequality (4.1) above, any user i in $\mathcal{C} \setminus \mathcal{R}$, we have $\text{AGR}(w_i, \mathcal{M}) \geq \rho\ell$. This comes from the fact that:

$$\text{AGR}(w_i, \mathcal{M}) \geq \ell - r(\ell - \Delta) \geq \rho\ell.$$

It implies user i has at least $\rho\ell$ sub-keys that agree with the mixture $\mathcal{M}$ and recovers at least $\rho\ell$ sub-messages K_i. By calling the function $\mathsf{Decrypt}$, he will receive the original message K.

We will show below the efficiency of our Trace & Revoke scheme. We will consider the parameters of the scheme to be the number of decryption keys per each user and the length of the ciphertext.

After the black-box tracing procedure, we get a pirate word. With a given pirate word, to ensure that the identify algorithm can return efficiently at least a traitor from a $t-$collusion, the constructed robust IPP codes must have traceability property. According to Proposition 3.1 in [BK13], the minimum Hamming distance of the code must satisfy

$$\Delta/\ell > 1 - \left(\frac{1-\varepsilon}{t^2} - \frac{\varepsilon}{t}\right),$$

whereas $0 < \varepsilon < (t+1)^{-1}$. This means that the codeword having closest distance to the pirate word is always a traitor.

Therefore, in order for the system to capture simultaneously tracing and revoking features, the q-ary robust IPP codes with minimum Hamming distance Δ must satisfy:

$$\begin{aligned} \Delta &> \ell \cdot \max\left\{1 - \frac{1-\rho}{r}, 1 - \left(\frac{1-\varepsilon}{t^2} - \frac{\varepsilon}{t}\right)\right\} \\ &= \ell \cdot \left(1 - \min\left\{\frac{1-\rho}{r}, \frac{1-\varepsilon}{t^2} - \frac{\varepsilon}{t}\right\}\right). \end{aligned}$$

The number of keys per user is ℓ and the ciphertext size is at most ℓ times the ciphertext size of AnoBEB. To summarize, we just proved the following theorem.

Theorem 4.1

Given

- $\mathcal{C} = (\ell, N, q)$, a $q-$ary code of Hamming distance Δ and $0 < \varepsilon < (t+1)^{-1}$;
- a $(\rho\ell, \ell)-$secret sharing scheme, for $\rho \in (0,1)$;
- an anonymous broadcast encryption for q users AnoBEB;

satisfying the following condition

$$\Delta/\ell > 1 - \min\left\{\frac{1-\rho}{r}, \frac{1-\varepsilon}{t^2} - \frac{\varepsilon}{t}\right\}. \tag{4.2}$$

Then Γ, constructed as above, is a TR scheme for N users in which we can revoke up to r users and trace succesfully at least one traitor from any coalition up to t traitors.

We remark that any code $\mathcal{C}$ verifying condition (4.2) is a robust IPP code, as

$$\Delta/\ell > 1 - \left(\frac{1-\varepsilon}{t^2} - \frac{\varepsilon}{t}\right).$$

Ciphertext Size. We now consider the ciphertext size of the scheme Γ. This turns out to evaluate the length of the Robust IPP code ℓ (times the size of an AnoBEB ciphertext). So we next estimate ℓ. But, we can not directly use any analysis in the paper [BK13] because of two reasons:

- In [BK13], a proof of existence of robust IPP codes was given, but there was no immediate explicit construction given there, neither given any the analysis of the length of the code.
- Robust IPP codes only deal with the number of traitor. In our scheme, we need, moreover, to take into account of the number of the revoked users that satisfying condition (4.2) above, so that in total we have an extended code requirement to consider (namely, robust IPP code supporting revocations).

Explicit construction of codes. The relative distance of the code $\mathcal{C}$ is defined by $\delta := \Delta/\ell$. We present two constructions.
Construction 1: We will consider a code with δ satisfying the Gilbert-Varshamov bound. To do this, let us pick

$$1 - \min\left\{\frac{1-\rho}{r}, \frac{1-\varepsilon}{t^2} - \frac{\varepsilon}{t}\right\} < \delta \leq 1 - \frac{1}{q}.$$

According to the Gilbert-Varshamov theorem (Theorem 4.10, [Rot06]), there exists a $q-$ary code $\mathcal{C}$ with rate $R(\mathcal{C}) = \frac{1}{\ell}\log_q N$ satisfying

$$R(\mathcal{C}) \geq 1 - H_q(\delta) - o(1),$$

where $H_q(\delta)$ is the q-ary entropy function $H_q : [0,1] \to \mathbb{R}$ defined by

$$H_q(\delta) = \delta \log_q \frac{q-1}{\delta} + (1-\delta)\log_q \frac{1}{1-\delta}.$$

We choose

$$d = \max\left\{\frac{r}{1-\rho}, \frac{t^2}{(1-\varepsilon)-\varepsilon t}\right\}.$$

Therefore

$$1 - 1/d < \delta \leq 1 - \frac{1}{q}.$$

To ensure the obtained code is not a random code, we apply the derandomization procedure of Porat-Rothschild [PR08]. This means that we give an explicit construction for the code $\mathcal{C}$. It progresses as follows:

We choose $\delta = 1 - \frac{1}{d+1}$. Obviously, we do not want large δ because that can only reduce the size of the code. To satisfy $\delta \leq 1 - \frac{1}{q}$ we need $q \geq d+1$. Since $q \geq d+1$, we choose $q = \Theta(d)$. Next, we need to estimate the value of $1 - H_q(\delta)$. We will use the fact that $\log(1+x) \approx x$ for small x extensively below.

$$\begin{aligned}
1 - H_q(\delta) &= 1 - \delta\log_q(q-1) + \delta\log_q\delta + (1-\delta)\log_q(1-\delta) \\
&= 1 - \log_q(q-1) + (1-\delta)\log_q[(q-1)(1-\delta)] + \delta\log_q\delta \\
&= \frac{\log\left(\frac{q}{q-1}\right)}{\log q} + \frac{\log[(q-1)/(d+1)]}{(d+1)\log q} - \frac{d}{d+1}\frac{\log(1+1/d)}{\log q} \\
&\approx \frac{1}{(q-1)\log q} + \frac{\log[\Theta(1)]}{(d+1)\log q} - \frac{1}{(d+1)\log q} \\
&= \Theta\left(\frac{1}{d\log q}\right).
\end{aligned}$$

Since $R(\mathcal{C}) \geq 1 - H_q(\delta) - o(1)$, we omit small terms and obtain $R(\mathcal{C}) = 1 - H_q(\delta)$. Moreover, $R(\mathcal{C}) = \frac{1}{\ell}\log_q N$, it implies the length of the code is

$$\ell = \frac{\log_q N}{R(\mathcal{C})} = \frac{\log_q N}{1 - H_q(\delta)} = O\left(d\log q\log_q N\right) = O(d\log N).$$

In short, we obtain

$$q = \Theta(d) \text{ and } \ell = O(d\log N).$$

Finally, we get

$$\ell = O\left(\max\left\{\frac{r}{1-\rho}, \frac{t^2}{(1-\epsilon)-\epsilon t}\right\}\log N\right).$$

Construction 2: In another construction, we consider our code $\mathcal{C}$ in the Reed-Solomon setting: we also pick

$$d = \max\left\{\frac{r}{1-\rho}, \frac{t^2}{(1-\epsilon)-\epsilon t}\right\}.$$

The Reed-Solomon code has $\delta = \frac{\ell-k+1}{\ell} = 1 - \frac{k}{\ell} + \frac{1}{\ell}$, whereas k is the dimension of code $\mathcal{C}$. In this case, if we choose $\ell = kd$ then $\delta > 1 - 1/d$. Hence, to use Reed-Solomon code we need to pick $q \geq \ell = kd$ such that $q^k \geq N$ or, equivalently, $\ell\log q \geq d\log N$.

For example, we can pick $q = \ell \approx \frac{2d\log N}{\log(d\log N)}$ and $k \approx \frac{\log N}{\log q}$. In this case, the length of the code is

$$\ell = O\left(\frac{2d\log N}{\log(d\log N)}\right).$$

4.2.2 Correctness and Security

Semantic Security. We now consider the security of the scheme Γ.

Theorem 4.2

Assume that the scheme AnoBEB is IND-secure, then the scheme Γ is also IND-secure.

Proof

We consider a sequence of games starting with **Game $\mathbf{G_0}$** as follows:

Game $\mathbf{G_0}$: This is the real game as defined in the security model. The challenger generates ℓ public keys $\{\mathsf{ek}_i\}_{i=1}^{\ell}$ and chooses robust IPP code $\mathcal{C} = \{w_1, \ldots, w_N\}$ which he then gives to the adversary $\mathcal{A}_\Gamma$. In **Phase 1**, $\mathcal{A}_\Gamma$ queries descryption keys for user $i \in \{1, \ldots, N\}$ and obtains dk_i, where

$$\mathsf{dk}_i = (\mathsf{sk}_{1,w_{i,1}}, \ldots, \mathsf{sk}_{j,w_{i,j}}, \ldots, \mathsf{sk}_{\ell,w_{i,\ell}}),$$

where $\mathsf{sk}_{j,w_{i,j}}$ is a decryption key extracted from the scheme Π by calling algorithm

$$\Pi.\mathsf{Extract}(\mathsf{ek}_j, \mathsf{MSK}_j, w_{i,j}).$$

In the **Challenger phase**, the adversary selects two messages $m^0, m^1 \in \mathcal{PT}^\ell$ and a subset of revoked users $\mathcal{R} \subset \mathcal{C}$. The challenger picks at random a $b \hookleftarrow \{0,1\}$, calls the procedure $\mathsf{Share}(\ell, \rho\ell, \ell)$ to get ℓ shares $m_1^b, \ldots, m_\ell^b$ for the message m^b and outputs a ciphertext $\Pi.\mathsf{Encrypt}(\mathsf{ek}_j, m_j^b, \mathcal{M}_j)_{j=1}^{\ell}$, where

$$(\mathcal{M}_1, \ldots, \mathcal{M}_\ell) = (\Sigma - \mathcal{R}[1], \ldots, \Sigma - \mathcal{R}[\ell]), \mathcal{R}[j] := \bigcup_{i|w_i \in \mathcal{R}} \{w_{i,j}\}.$$

In **Phase 2**, $\mathcal{A}_\Gamma$ received the ciphertext, sampled from one of two computationally indistinguishable distributions

$$\begin{aligned}
\mathcal{D}_0 &= \Big(\Pi.\mathsf{Encrypt}(\mathsf{ek}_1, m_1^0, \mathcal{M}_1), \ldots, \Pi.\mathsf{Encrypt}(\mathsf{ek}_\ell, m_\ell^0, \mathcal{M}_\ell)\Big) \\
\mathcal{D}_1 &= \Big(\Pi.\mathsf{Encrypt}(\mathsf{ek}_1, m_1^1, \mathcal{M}_1), \ldots, \Pi.\mathsf{Encrypt}(\mathsf{ek}_\ell, m_\ell^1, \mathcal{M}_\ell)\Big),
\end{aligned}$$

$\mathcal{A}_\Gamma$ outputs a guess b' for b. Let $\mathsf{Adv}_{\mathcal{A}_\Gamma}^{\mathbf{Game\ 0}}(\mathcal{D}_0, \mathcal{D}_1)$ be the advantage of $\mathcal{A}_\Gamma$ with two given distributions $\mathcal{D}_0$ and $\mathcal{D}_1$. The advantage is defined by:

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{A}_\Gamma}^{\mathbf{Game\ 0}}(\mathcal{D}_0, \mathcal{D}_1) &= \Big|2\Pr\left[\mathcal{A}_\Gamma(\mathcal{D}_b) = b\right] - 1\Big| \\
&= \Big|\Pr\left[\mathcal{A}_\Gamma(\mathcal{D}_0) = 1\right] - \Pr\left[\mathcal{A}_\Gamma(\mathcal{D}_1) = 1\right]\Big|.
\end{aligned}$$

Game $\mathbf{G_1}$: The challenger now makes one small change to the **Game $\mathbf{G_0}$**. Namely, instead of encrypting the first share m_1^0 with all keys in the mixture $\mathcal{M}_1$, we encrypt m_1^1 with the mixture $\mathcal{M}_1$. This means that the

challenger only changes the first coordinate in $\mathcal{D}_0$ and does not do anything with $\mathcal{D}_1$. In this game, all steps are the same as in **Game $\mathbf{G_0}$** except as mentioned about the above ciphertext. Thus, $\mathcal{A}_\Gamma$ will receive a challenger ciphertext, sampled from one of two computationally indistinguishable distributions $\mathcal{D}_0^1$ and $\mathcal{D}_1$, where

$$\mathcal{D}_0^1 = \Big(\Pi.\mathsf{Encrypt}(\mathsf{ek}_1, m_1^1, \mathcal{M}_1), \quad \Pi.\mathsf{Encrypt}(\mathsf{ek}_2, m_2^0, \mathcal{M}_2) \\ , \quad \ldots, \Pi.\mathsf{Encrypt}(\mathsf{ek}_\ell, m_\ell^0, \mathcal{M}_\ell)\Big).$$

We denote the advantage of the adversary in this game by $\mathsf{Adv}_{\mathcal{A}_\Gamma}^{\mathbf{Game\ 1}}(\mathcal{D}_0^1, \mathcal{D}_1)$. The, from this, we can see that

$$\begin{aligned} \Big|\Pr\left[\mathcal{A}_\Gamma(\mathcal{D}_0) = 1\right] \; - \; \Pr\left[\mathcal{A}_\Gamma(\mathcal{D}_1) = 1\right]\Big| \\ \leq \; \Big|\Pr\left[\mathcal{A}_\Gamma(\mathcal{D}_0) = 1\right] - \Pr\left[\mathcal{A}_\Gamma(\mathcal{D}_0^1) = 1\right]\Big| \\ + \; \Big|\Pr\left[\mathcal{A}_\Gamma(\mathcal{D}_0^1) = 1\right] - \Pr\left[\mathcal{A}_\Gamma(\mathcal{D}_1) = 1\right]\Big|. \end{aligned}$$

Therefore, we have

$$\mathsf{Adv}_{\mathcal{A}_\Gamma}^{\mathbf{Game\ 0}}(\mathcal{D}_0, \mathcal{D}_1) \leq \mathsf{Adv}_{\mathcal{A}_\Gamma}^{\mathbf{Game\ 1}}(\mathcal{D}_0^1, \mathcal{D}_1) + \varepsilon_1,$$

where ε_1 is a quantity, defined by

$$\varepsilon_1 := \Big|\Pr_{x \leftarrow \mathcal{D}_0}[\mathcal{A}_\Gamma(x) = 1] - \Pr_{x \leftarrow \mathcal{D}_0^1}[\mathcal{A}_\Gamma(x) = 1]\Big|.$$

Claim 1. ε_1 is bounded by an avantage of the attacker in AnoBEB scheme, namely

$$\varepsilon_1 \leq \mathsf{Adv}_{\mathsf{AnoBEB}}.$$

Indeed, assume the contrary, that there exists a polynomial time attacker $\mathcal{A}_{\mathsf{DIST}}$ which is able to distinguish between the two distributions $\mathcal{D}_0$ and $\mathcal{D}_0^1$ with a non-negligible probability. We then build a simulator S to break the Π scheme as follows:

The simulator takes as input a public key ek_Π and generates $(\ell - 1)$ pairs of public key and secret key $\{\mathsf{ek}_i, \mathsf{MSK}_i\}_{i=2}^{\ell}$. S passes $\mathsf{ek} = (\mathsf{ek}_\Pi, \mathsf{ek}_2, \ldots, \mathsf{ek}_\ell)$ to $\mathcal{A}_{\mathsf{DIST}}$. S also collects some parameters such as: the shares $\{m_1^0, \ldots, m_\ell^0\}$, $\{m_1^1, \ldots, m_\ell^1\}$ and the family of mixture $\{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\ell\}$. By querying the challenger of the scheme Π with the shares m_1^0, m_1^1 and the mixture $\mathcal{M}_1$, S it receives a ciphertext of the form, $\mathsf{Encrypt}(\mathsf{ek}_1, m_1^b, \mathcal{M}_1)$, where bit b was chosen randomly by the challenger. The others ciphertexts $\{\mathsf{Encrypt}(\mathsf{ek}_j, m_j^0, \mathcal{M}_j)\}_{j=2}^{\ell}$ generated by the simulator as well to establish a full ciphertext

$$\Big(\mathsf{Encrypt}(\mathsf{ek}_1, m_1^b, \mathcal{M}_1), \mathsf{Encrypt}(\mathsf{ek}_2, m_2^0, \mathcal{M}_2), \ldots, \mathsf{Encrypt}(\mathsf{ek}_\ell, m_\ell^0, \mathcal{M}_\ell)\Big).$$

By our assumption, $\mathcal{A}^{\mathsf{DIST}}$ can distinguish efficiently the two distributions above, as soon as $\mathcal{A}^{\mathsf{DIST}}$ outputs bit b, the simulator S will return the same value b. We see that if $m_1^0 = m_1^1$, the two distributions $\mathcal{D}_0$ and $\mathcal{D}_0^1$ coincide.

To summarize, we already built an efficient simulator to break the scheme Π and it is a contradiction because Π is $\mathtt{IND}-$secure.

Game $\mathbf{G_2}$: This game is identical with **Game $\mathbf{G_1}$** with the difference that the challenger changes the second coordinate in $\mathcal{D}_0^1$ by $\Pi.\mathsf{Encrypt}(\mathsf{ek}_2, m_2^1, \mathcal{M}_2)$ and still does not do anything with $\mathcal{D}_1$. Thus, $\mathsf{Adv}_{\mathcal{A}_\Gamma}$ will receive a challenger ciphertext, sampled from one of two computationally indistinguishable distributions $\mathcal{D}_0^2$ and $\mathcal{D}_1$, where

$$\mathcal{D}_0^2 = \Big(\Pi \ . \ \mathsf{Encrypt}(\mathsf{ek}_1, m_1^1, \mathcal{M}_1), \\ \Pi.\mathsf{Encrypt}(\mathsf{ek}_2, m_2^1, \mathcal{M}_2), \ldots, \Pi.\mathsf{Encrypt}(\mathsf{ek}_\ell, m_\ell^0, \mathcal{M}_\ell)\Big).$$

We denote the advantage of the adversary in this game by $\mathsf{Adv}_{\mathcal{A}_\Gamma}^{\mathbf{Game\ 2}}(\mathcal{D}_0^2, \mathcal{D}_1)$. And from this, we have

$$\mathsf{Adv}_{\mathcal{A}_\Gamma}^{\mathbf{Game\ 1}}(\mathcal{D}_0^1, \mathcal{D}_1) \leq \mathsf{Adv}_{\mathcal{A}_\Gamma}^{\mathbf{Game\ 2}}(\mathcal{D}_0^2, \mathcal{D}_1) + \varepsilon_2,$$

where ε_2 is a quantity, defined by

$$\varepsilon_2 := \Big| \Pr_{x \leftarrow \mathcal{D}_0^1}[\mathcal{A}_\Gamma(x) = 1] - \Pr_{x \leftarrow \mathcal{D}_0^2}[\mathcal{A}_\Gamma(x) = 1] \Big|.$$

By an argument analogous to that of **Claim 1**, we get $\varepsilon_2 \leq \mathsf{Adv}_{\mathsf{AnoBEB}}$.

Game $\mathbf{G_\ell}$: We substitute the ℓ^{th} coordinate of the distribution $\mathcal{D}_0^\ell$ by

$$\Pi.\mathsf{Encrypt}(\mathsf{ek}_\ell, m_\ell^1, \mathcal{M}_\ell)$$

and still introduce no change to the distribution $\mathcal{D}_1$. $\mathsf{Adv}_{\mathcal{A}_\Gamma}$ will receive a challenger ciphertext, sampled from one of two computationally identical distributions $\mathcal{D}_0^\ell$ and $\mathcal{D}_1$. We denote advantage of the adversary in this game by $\mathsf{Adv}_{\mathcal{A}_\Gamma}^{\mathbf{Game\ \ell}}(\mathcal{D}_0^\ell, \mathcal{D}_1)$. Then, from this, we have

$$\mathsf{Adv}_{\mathcal{A}_\Gamma}^{\mathbf{Game\ \ell-1}}(\mathcal{D}_0^{\ell-1}, \mathcal{D}_1) \leq \mathsf{Adv}_{\mathcal{A}_\Gamma}^{\mathbf{Game\ \ell}}(\mathcal{D}_0^\ell, \mathcal{D}_1) + \varepsilon_\ell = \varepsilon_\ell,$$

where ε_ℓ is a quantity, defined by

$$\varepsilon_\ell := \Big| \Pr_{x \leftarrow \mathcal{D}_0^{\ell-1}}[\mathcal{A}_\Gamma(x) = 1] - \Pr_{x \leftarrow \mathcal{D}_0^\ell}[\mathcal{A}_\Gamma(x) = 1] \Big| \leq \mathsf{Adv}_{\mathsf{AnoBEB}}.$$

Putting the above arguments altogether and applying the triangle inequality we have:

$$\begin{aligned}\left|\mathsf{Adv}^{\mathbf{Game}\ 0}_{\mathcal{A}_\Gamma}(\mathcal{D}_0,\mathcal{D}_1)\right| &= \left|\mathsf{Adv}^{\mathbf{Game}\ 0}_{\mathcal{A}_\Gamma}(\mathcal{D}_0,\mathcal{D}_1) - \mathsf{Adv}^{\mathbf{Game}\ \ell}_{\mathcal{A}_\Gamma}(\mathcal{D}_0^\ell,\mathcal{D}_1)\right| \\ &\leq \sum_{i=1}^{\ell}\varepsilon_\ell \leq \ell.\mathsf{Adv}_{\mathsf{AnoBEB}}.\end{aligned}$$

Further Remarks.

- The length of the above robust IPP codes is approximately the length of the best collusion secure code which is essentially $O(t^2(\log\frac{N}{\epsilon}))$ [Tar03], where ϵ is the error probability in identifying traitors (we note that an interesting property in IPP and robust IPP codes is that one achieves zero error in identifying traitors). Suppose that one can construct an AnoBEB which is as efficient as the underlying PKE (which is the case for LWE encryption as we achieve in this work), then our proposed robust IPP code based Trace & Revoke schemes has the same ciphertext size as the state of the art collusion secure code based traitor tracing schemes. Note that, unlike our case, one cannot revoke users in the collusion secure code based traitor tracing schemes.

- We compare now our schemes with other Trace & Revoke schemes in the bounded collusion model. If the tracing model is relaxed to be the black-box confirmation with the assumption that the tracer gets a suspect set that contains all the traitors, then the Agrawal et al.'s scheme from CCS '17 [ABP+17] is the most efficient with the ciphertext size of $\widetilde{O}(r+t+n)$ where n the security parameter. However, transforming from black-box confirmation to black-box tracing requires an exponential lost and thus is impractical: one have to make a correct guess on the set that contains exactly the traitors. Focusing on the standard black-box tracing in the bounded collusion model, the result in [NWZ16] gives the referred schemes. As stated in [ABP+17], when based on the bounded collusion FE of [GVW12], the resulting scheme in [NWZ16] has a ciphertext size growing at least as $\widetilde{O}((r+t)^5\mathcal{P}oly(n))$; by relying on learning with errors, this blowup can be improved to $\widetilde{O}((r+t)^4\mathcal{P}oly(n))$, but at the cost of relying on heavy machinery such as attribute based encryption [GVW13] and fully homomorphic encryption [GKP+13]. Our Trace & Revoke achieves the ciphertext size complexity of $\widetilde{O}((r+t^2)(n^2)\log N)$ (the code length multiplied by the LWE ciphertext size), for a system of N users. Our construction thus gives the most efficient Trace & Revoke scheme for standard black-box tracing in the bounded collusion model.

- We provided a construction of AnoBEB which is as efficient as the underlying LWE PKE. We raise an open question of constructing AnoBEB schemes from other standard encryptions such as ElGamal, RSA, Paillier encryptions without a significant loss in the ciphertext size.

- Boneh-Naor [BN08] and Billet-Phan [BP08] provided a solution to tracing traitors from imperfect pirate device, with short ciphertext size. Their schemes were built from robust collusion secure codes and PKE. We can completely follow these methods to obtain a traitor tracing scheme from a robust IPP code and an AnoBEB with short ciphertext size. However, we target the more challenging case of Trace & Revoke in this thesis.

5 Traceable Inner Product Functional Encryption

This chapter is devoted to introducing a new primitive, called *Traceable Functional Encryption*. This work's motivation is as follows: Functional Encryption (FE) has been widely studied in the last decade, as it provides a very useful tool for restricted access to sensitive data. From a ciphertext, it allows specific users to learn a function of the underlying plaintext. In practice, many users may be interested in the same function on the data, say the mean value of the inputs, for example. The conventional definition of FE associates each function to a secret *decryption functional key* and therefore, all the users get the same secret key for the same function. This induces an important problem: if one of these users (called a *traitor*) leaks or sells the decryption functional key to be included in a *pirate* decryption tool, then there is no way to trace back its identity. However, in the new primitive, the functional decryption key will not only be specific to a function but to a user too, in such a way that if some users collude to produce a pirate decoder that successfully evaluates a function on the plaintext, from the ciphertext only, one can trace back at least one of them.

We will propose a concrete construction Traceable Functional Encryption for Inner Product. Our observation is that the ElGamal-based IPFE from Abdalla et al. in PKC '15 shares many similarities with the Boneh-Franklin traitor tracing from CRYPTO '99. We can then combine these two schemes in a very efficient way, with the help of pairings, to obtain a Traceable IPFE with black-box confirmation.

Contents

5.1 Traceable Functional Encryption

5.1.1 Definition

We begin by describing the syntactic definition of traceable functional encryption (TFE) for circuits. A functionality (circuit) $F \in \mathcal{F}_\lambda$ describes the function of a plaintext that can be derived from the ciphertext. More precisely, a functionality is defined as follows.

Definition 5.1

Let $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda\in\mathbb{N}}$ denote ensembles where each $\mathcal{Y}_\lambda$ and $\mathcal{S}_\lambda$ is a finite set. Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ denotes an ensemble where each $\mathcal{F}_\lambda$ is a finite collection of circuits, and each circuit $F \in \mathcal{F}_\lambda$ takes as input a message $y \in \mathcal{Y}_\lambda$ and outputs $F(y) \in \mathcal{S}_\lambda$.

Definition 5.2

A traceable functional encryption scheme $\mathcal{TFE}$ for an ensemble $\mathcal{F}$ consists of five algorithms $(\mathsf{Setup}, \mathsf{Extract}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Tracing})$ defined as follows:

$\mathsf{Setup}(1^\lambda)$**:** Takes as input a security parameter λ and outputs a master key pair $(\mathsf{PK}, \mathsf{MSK})$.

$\mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, F)$**:** Given an identity ID of a user, a circuit $F \in \mathcal{F}_\lambda$ and the master secret key MSK, this algorithm outputs an individual functional secret key $\mathsf{sk}_{F,\mathsf{ID}}$.

$\mathsf{Encrypt}(\mathsf{PK}, y)$**:** Takes as input the public key PK and a message $y \in \mathcal{Y}_\lambda$, this randomized algorithm outputs a ciphertext CT.

$\mathsf{Decrypt}(\mathsf{PK}, \mathsf{sk}_{F,\mathsf{ID}}, \mathsf{CT})$**:** Given the public key PK, a secret key $\mathsf{sk}_{F,\mathsf{ID}}$ and a ciphertext CT, this algorithm outputs $F(y) \in \mathcal{S}_\lambda$, if CT encrypts y, or an invalid symbol $\perp$.

$\mathsf{Tracing}^{\mathcal{D}_F}(\mathsf{MSK}, F, \mu(.), y^0, y^1)$**:** The tracing algorithm takes as input the master secret key MSK, a circuit $F \in \mathcal{F}_\lambda$, two messages $y^0, y^1 \in \mathcal{Y}_\lambda$ which are obtained from $\mathcal{D}_F$ and a function $\mu(.)$ representing the probability that the decoder can distinguish between the ciphertexts of y^0 and of y^1. The algorithm interacts with a confiscated pirate decoder $\mathcal{D}_F$, as a black-box, and outputs an identity or an invalid symbol $\perp$.

For correctness, we require that for all $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$, all $y \in \mathcal{Y}_\lambda$, each $F \in \mathcal{F}_\lambda$ and all identities ID, $\mathsf{sk}_{F,\mathsf{ID}} \leftarrow \mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, F)$, if $\mathsf{CT} \leftarrow \mathsf{Encrypt}(\mathsf{PK}, y)$, then one should get $\mathsf{Decrypt}(\mathsf{PK}, \mathsf{sk}_{F,\mathsf{ID}}, \mathsf{CT}) = F(y)$, with overwhelming probability.

Requirement on the pirate decoder

- The classical requirement is that the pirate decoder $\mathcal{D}_F$ is a device that is able to decrypt successfully any normal ciphertext generated by the $\mathsf{Encrypt}$ algorithm with high probability. Yet, in another approach, the tracer is only able to interact with $\mathcal{D}_F$ through an oracle $\mathcal{O}_F^{\mathcal{D}}$ by sending a message-ciphertext pair (tracing signal) to $\mathcal{O}_F^{\mathcal{D}}$ and gets a response that is a bit indicating whether $\mathcal{D}_F$ can successfully decrypt the ciphertext into the provided message (evaluated with the function F). We say that the tracing algorithm is executing in minimal access black-box mode.

$$\mathcal{O}_F^{\mathcal{D}}(\mathsf{CT}, y) = \begin{cases} 1 & \text{if } \mathcal{D}_F(\mathsf{CT}) = F(y) \\ 0 & \text{otherwise.} \end{cases}$$

- We consider the same setting for the pirate as in [GKW18b]: of course, this is not required the pirate decoder $\mathcal{D}_F$ to output entire message (or an indicator bit as in minimal access model) nor to decrypt with high probability every ciphertexts which are taken from random messages. Instead, it is enough that the pirate decoder can distinguish the encryption of two messages y^0, y^1 which are chosen by itself (see [GKW18b]): Adapted from [GKW18b], we define a μ-useful Pirate Distinguisher $\mathcal{D}_F$ associated to a unique function F as below

$$\left| \mathsf{Pr} \left[\mathcal{D}_F\left(\mathsf{CT}_b\right) = b : \begin{array}{l} (\mathsf{MSK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(\cdot) \\ \{\mathsf{sk}_{F,i} \leftarrow \mathsf{Extract}(i, \mathsf{MSK}, F)\}_{i\in[n]} \\ (\mathcal{D}_F, y^0, y^1) \leftarrow \mathcal{A}(\mathsf{PK}, \{\mathsf{sk}_{F,i}\}_{i\in[t]}) \\ \text{st. } F(y^0) \neq F(y^1) \\ b \xleftarrow{\$} \{0,1\}, \mathsf{CT}_b \leftarrow \mathsf{Encrypt}(\mathsf{PK}, y^b) \end{array} \right] - \frac{1}{2} \right| \geq \mu(\lambda),$$

  where the function $\mu(\cdot)$ is a non-negligible function in λ.

  This very strong notion of Pirate Distinguisher has been introduced in [GKW18b]. It requires the pirate distinguisher to be able to distinguish the encryption of two different messages y^0, y^1. To adapt to the functional encryption, as the goal of the pirate is to compute the function on the message, we require that the pirate distinguisher be able to distinguish the encryption of y^0, y^1 such that $F(y^0) \neq F(y^1)$.

  As shown in [GKW18b], this notion is stronger than the classical Pirate Decoder which is able to correctly decrypt random messages with non-negligible probability. When considering the case of functional encryption, a pirate decoder for a function F is useful if it can compute $F(y)$ from the encryption of y, for a random message y. Clearly, pirate distinguisher is also stronger than pirate decoder in this case. Indeed, one can build a distinguisher $\mathcal{D}_F$ from a decoder Dec_F: randomly choose y^0, y^1 such that $F(y^0) \neq F(y^1)$, then when receiving the challenge ciphertext CT, call Dec_F and check whether this is $F(y^0)$ or $F(y^1)$ to output the correct guess, if this is none of them, output a random guess. In this work, we will deal with this notion of pirate distinguisher which is actually the strongest notion (*i.e.*, minimal requirement) about the usefulness of pirate decoders.

5.1.2 Security

We consider the IND security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{B}$ as follows:

Definition 5.3

A traceable functional encryption scheme $\mathcal{T}-\mathcal{FE}$ for an ensemble $\mathcal{F}$, $\mathcal{T}-\mathcal{FE} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Tracing})$ is semantically secure under chosen-plaintext attacks (or $\mathsf{IND}-\mathsf{CPA}$ security) if no PPT adversary has non-negligible advantage in the following game:

- The challenger $\mathcal{B}$ runs $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$ and the public key PK is given to the adversary $\mathcal{A}$.
- The adversary adaptively makes secret key queries to the challenger. That is, the adversary $\mathcal{A}$ chooses some pairs of identities ID and functions $F \in \mathcal{F}_\lambda$. $\mathcal{A}$ sends them to $\mathcal{B}$ and then obtains $\mathsf{sk}_{F,\mathsf{ID}} \leftarrow \mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, F)$ from $\mathcal{B}$.
- The adversary $\mathcal{A}$ chooses distinct messages $y_0, y_1 \in \mathcal{Y}_\lambda$ such that $F(y_0) = F(y_1)$ for all F already asked. This restriction is required in all functional encryption to avoid trivial attacks. Whenever $\mathcal{B}$ receives the messages, it randomly picks $\beta \xleftarrow{\$} \{0,1\}$ and then transfers to $\mathcal{A}$ a ciphertext $\mathsf{CT}_\beta = \mathsf{Encrypt}(\mathsf{PK}, y_\beta)$.
- Adversary $\mathcal{A}$ continues making further decryption key queries for other pairs of identities ID and functions F, and receives $\mathsf{sk}_{F,\mathsf{ID}}$ from $\mathcal{B}$. Again, it is also required that $F(y_0) = F(y_1)$ to avoid trivial attacks.
- Adversary $\mathcal{A}$ eventually returns a guess β' for a bit β and wins if $\beta' = \beta$.

A weaker version has been defined, when the messages y_0, y_1 for the challenge ciphertext are chosen before the Setup algorithm started, then the $\mathcal{T}-\mathcal{FE}$ scheme is said selectively-security against chosen-plaintext attacks, which is denoted by $\mathsf{sel}-\mathsf{IND}-\mathsf{CPA}$.

Traceability. The security game between the attacker $\mathcal{A}$ and the challenger $\mathcal{B}$ takes place as follows:

1. The challenger $\mathcal{B}$ runs $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$ and the public key PK sent to the adversary $\mathcal{A}$. $\mathcal{B}$ also creates a table $\mathcal{T}$ to store pairs of identities of users who queried keys and functions F, for all $F \in \mathcal{F}_\lambda$. It means that the table $\mathcal{T}$ stores (ID, F). Initially $\mathcal{T}$ is set empty.
2. The adversary adaptively makes secret key queries to the challenger. Concretely, the adversary $\mathcal{A}$ chooses some pairs of identities ID and functions $F \in \mathcal{F}_\lambda$ to query functional secret keys. The challenger $\mathcal{B}$ stores all these pairs in the table $\mathcal{T}$ and replies with the secret keys $\mathsf{sk}_{F,\mathsf{ID}}$ for those pairs.
3. The adversary $\mathcal{A}$ outputs $(F^*, \mathcal{D}_{F^*})$ and two messages y^0, y^1, where $\mathcal{D}_{F^*}$ is a pirate distinguisher for the function F^*.

4. After receiving the messages y^0, y^1 from $\mathcal{A}$, the challenger $\mathcal{B}$ runs the algorithm $\mathsf{Tracing}^{\mathcal{D}_{F^*}}(\mathsf{MSK}, F^*, 1^\mu, y^0, y^1)$ and outputs an identity ID^*.

We say that the adversary $\mathcal{A}$ wins the game if the output of $\mathsf{Tracing}$ is either an invalid symbol $\mathsf{ID}^* = \perp$ or the identity ID^* did not ask for F^*: $(\mathsf{ID}^*, F^*) \notin \mathcal{T}$.

When the adversary $\mathcal{A}$ is allowed to ask secret keys for the only target function F^* (but for any ID), and so for (ID, F^*), the security of Tracing algorithm will then be called *one-target security*.

We will explain that the one-target security also covers the case where the adversary outputs any function F such that the target function F^* is computable from F with public information. In such a case, when the pirate outputs the function F and the decoder $\mathcal{D}_F$ (together with two messages), one can define a decoder $\mathcal{D}_{F^*}$ that calls $\mathcal{D}_F$ and then applies the computation of F^* from F on the output, then do tracing on this $\mathcal{D}_{F^*}$, applying also the public transformation to the messages.

5.2 Construction for Inner-Product Case

Intuition. We exploit the similarities between the Boneh and Franklin's traitor tracing scheme [BF99] and the Abdalla et al.'s IPFE scheme [ABDP15] to integrate the Boneh-Franklin tracing technique into the IPFE scheme of Abdalla et al. [ABDP15] which allows in particular to personalize functional decryption keys. Interestingly, our method of personalizing keys and adding traceability does not need a huge extra cost as it is usually required for others primitives such as broadcast encryption.

We first informally recall the main ingredients of the IPFE of Abdalla et al. [ABDP15], that encrypts a plaintext vector $\vec{y} = (y_1, \ldots, y_k)$ as follows: the master secret key $\mathsf{MSK} = \vec{s} = (s_1, \ldots, s_k)$ and the public key $\mathsf{PK} = \left(\mathbb{G}, \left(h_i = g^{s_i}\right)_{i\in[k]}\right)$ respectively allow to generate functional decryption keys and ciphertexts:

$$\mathsf{sk}_{\vec{x}} = \langle \vec{s}, \vec{x} \rangle = \sum_{i\in[k]} s_i \cdot x_i, \qquad \mathsf{CT}_{\vec{y}} = \left(g^r, \left(h_i^r \cdot g^{y_i}\right)_{i\in[k]}\right).$$

Here, we are working in a cyclic $\mathbb{G}$ of prime order q, with a generator g. The master secret key MSK is a vector $\vec{s}$ with components s_i are taken from $\mathbb{Z}_q$. The public key PK consists of k group elements h_i. The vector $\vec{x} = (x_1, \ldots, x_k)$ with components x_i is taken from $\mathbb{Z}_q$ is used to extract a functional decryption key $\mathsf{sk}_{\vec{x}}$. A ciphertext, which is generated for a plaintext $\vec{y}$, denoted by $\mathsf{CT}_{\vec{y}}$. The $\mathsf{Decrypt}$ algorithm computes

$$\prod_{i\in[k]} \left(h_i^r \cdot g^{y_i}\right)^{x_i} \times \left(g^r\right)^{-\mathsf{sk}_{\vec{x}}} = \frac{g^{\langle \vec{s},\vec{x}\rangle r} \cdot g^{\langle \vec{x},\vec{y}\rangle}}{g^{\langle \vec{s},\vec{x}\rangle r}} = g^{\langle \vec{x},\vec{y}\rangle}$$

and gets $\langle \vec{x}, \vec{y} \rangle$, which is supposed to be relatively small, to allow the computation of the discrete logarithm.

For the mean value, the vector $\vec{x}$ is $(1, \ldots, 1)$. If many users are interested in the mean value then they all get the same functional decryption key $\mathsf{sk}_{\vec{x}}$ and there will be no way to trace the source of the leakage if this secret key is used somewhere.

In order to personalize functional decryption keys for each vector $\vec{x}$, we have got inspired from the seminal technique of Boneh-Franklin: we associate to each user a representation of $g^{\langle \vec{s},\vec{x}\rangle}$ in the basis of $\left(b_i = g^{t_i}\right)_{i\in[k]}$, with t_i is taken from $\mathbb{Z}_q$.

Therefore, by adding b_i^r in the ciphertext, each user can compute $g^{\langle \vec{s},\vec{x}\rangle r}$ as above and the decryption works in the same manner. Concretely, each user ID is associated to a public codeword $\vec{\theta}_{\mathsf{ID}} = (\theta_1, \ldots, \theta_k)$ and then, the personal secret key will be simply set to: $\mathsf{tk}_{\vec{x},\mathsf{ID}} = \langle \vec{s}, \vec{x}\rangle / \langle \vec{t}, \vec{\theta}_{\mathsf{ID}}\rangle$. The master secret key MSK consists of two vectors $\vec{s} = (s_1, \ldots, s_k)$ and $\vec{t} = (t_1, \ldots, t_k)$. The public key

$$\mathsf{PK} = \left(\mathbb{G}, \left(b_i = g^{t_i}\right)_{i\in[k]}, \left(h_i = g^{s_i}\right)_{i\in[k]}\right).$$

For each plaintext $\vec{y}$, the ciphertext is

$$\mathsf{CT}_{\vec{y}} = \left(\left(b_i^r\right)_{i\in[k]}, \left(h_i^r \cdot g^{y_i}\right)_{i\in[k]}\right).$$

The $\mathsf{Decrypt}$ algorithm then outputs

$$\prod_{i\in[k]} \left(h_i^r \cdot g^{y_i}\right)^{x_i} \times \prod_{i\in[k]} \left(b_i^r\right)^{-\mathsf{tk}_{\vec{x},\mathsf{ID}}\theta_i} = \frac{g^{\langle \vec{s},\vec{x}\rangle r} \cdot g^{\langle \vec{x},\vec{y}\rangle}}{g^{\langle \vec{s},\vec{x}\rangle r}} = g^{\langle \vec{x},\vec{y}\rangle}.$$

The use of pairings. The above technique of personalizing secret keys seems to work well as in the Boneh-Franklin traitor tracing. However, there exists an issue specific to the setting of the functional encryption, that goes beyond the framework of Boneh-Franklin traitor tracing. Suppose that we are considering a scheme for two users with identities ID_1 and ID_2. The first user queries the secret keys corresponding to vectors $\vec{x}_1$ and $\vec{x}_2$ and gets $\mathsf{tk}_{\vec{x}_1,\mathsf{ID}_1} = \frac{\langle \vec{s},\vec{x}_1\rangle}{\langle \vec{t},\vec{\theta}_{\mathsf{ID}_1}\rangle}$ and $\mathsf{tk}_{\vec{x}_2,\mathsf{ID}_1} = \frac{\langle \vec{s},\vec{x}_2\rangle}{\langle \vec{t},\vec{\theta}_{\mathsf{ID}_1}\rangle}$. The second user only queries secret key to vector $\vec{x}_1$ and gets $\mathsf{tk}_{\vec{x}_1,\mathsf{ID}_2} = \frac{\langle \vec{s},\vec{x}_1\rangle}{\langle \vec{t},\vec{\theta}_{\mathsf{ID}_2}\rangle}$. From these three secret keys $\mathsf{tk}_{\vec{x}_1,\mathsf{ID}_1}, \mathsf{tk}_{\vec{x}_2,\mathsf{ID}_1}$ and $\mathsf{tk}_{\vec{x}_1,\mathsf{ID}_2}$, it is possible to compute the secret key $\mathsf{tk}_{\vec{x}_2,\mathsf{ID}_2} = \frac{\mathsf{tk}_{\vec{x}_2,\mathsf{ID}_1} \cdot \mathsf{tk}_{\vec{x}_1,\mathsf{ID}_2}}{\mathsf{tk}_{\vec{x}_1,\mathsf{ID}_1}}$ for the vector $\vec{x}_2$ and identity ID_2. To avoid this attack, we will put the scalar $t_{\vec{x},\mathsf{ID}}$ in the exponent $\mathsf{sk}_{\vec{x},\mathsf{ID}} = g^{\mathsf{tk}_{\vec{x},\mathsf{ID}}}$ and the decryption will then be performed in the target group of the pairing. The goal of the rest of this chapter is to prove this modification actually leads to a secure scheme.

Enhancing the security of IPFE. It is worth noticing that, by putting the secret key in the exponent, we may enhance the security of the functional encryption. In the Abdalla et al.'s scheme [ABDP15], whenever the adversary queries more than k secret keys, it can get the whole MSK by solving a system of linear equations. In our scheme, there is no way, unless breaking discrete logarithm, to get this master key as it is only put in the exponent. We will though not exploit further this advantage in this work, as we will focus on traceability.

Construction. We will describe concretely a traceable functional encryption for inner product scheme $(\mathcal{T} - \mathcal{FE})$ for n users. Let $\mathbb{G}$ be a bilinear group of large prime of order q. Additionally, let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ denote a bilinear map, where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of order q, written multiplicatively.

$\mathsf{Setup}(1^\lambda, 1^k)$**:** This algorithm generates a bilinear setting $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e)$ for sufficiently large prime order q and g_1, g_2 respectively are generators of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$. The bilinear map e over $\mathbb{G}_1$, $\mathbb{G}_2$ can be calculated efficiently.

- Randomly choose $t_1, \ldots, t_k \xleftarrow{\$} \mathbb{Z}_q$, set $\vec{t} = (t_1, \ldots, t_k)$ and $b_1 = g_1^{t_1}, \ldots, b_k = g_1^{t_k}$.

- For each $i \in \{1, \ldots, k\}$, randomly choose $s_i \xleftarrow{\$} \mathbb{Z}_q$. We set $\vec{s} = (s_1, \ldots, s_k)$ and set $G = e(g_1, g_2) \in \mathbb{G}_T$ and $H_i = G^{s_i} \in \mathbb{G}_T$ for all $i = 1, \ldots, k$.
- We consider a linear code $\mathbf{\Gamma}$ over the alphabet $\mathbb{Z}_q$ with n codewords $\mathbf{\Gamma} = \{\vec{\theta}_1, \ldots, \vec{\theta}_n\}$, corresponding to n users in our system. Each codeword has the length k.
- The public key is $\mathsf{PK} = \Big(\mathbb{G}, \mathbf{\Gamma}, g_1, g_2, G, H_1, \ldots, H_k, b_1, \ldots, b_k\Big)$.
- The master secret key is $\mathsf{MSK} = \{\vec{s}, \vec{t}\}$.

$\mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, \vec{x})$**:** Takes as input an identity ID, the master secret key MSK and a characteristic vector $\vec{x} = (x_1, \ldots, x_k) \in \mathbb{Z}_q^k$. Choose a (new) vector (codeword) $\vec{\theta}_{\mathsf{ID}} = (\theta_1, \ldots, \theta_k) \in \mathbf{\Gamma}$. A secret key is an element $g_2^{\mathsf{tk}_{\vec{x},\mathsf{ID}}} \in \mathbb{G}_2$ such that $\mathsf{tk}_{\vec{x},\mathsf{ID}} \cdot \vec{\theta}_{\mathsf{ID}}$ is a representation of $g_1^{\langle \vec{s},\vec{x} \rangle}$ in the basis of $(b_1, b_2, \ldots, b_k)$. That is $g_1^{\langle \vec{s},\vec{x} \rangle} = \prod_{i=1}^k b_i^{\mathsf{tk}_{\vec{x},\mathsf{ID}}\theta_i} = b_1^{\mathsf{tk}_{\vec{x},\mathsf{ID}}\theta_1} \cdots b_k^{\mathsf{tk}_{\vec{x},\mathsf{ID}}\theta_k}$. Concretely, set $\mathsf{tk}_{\vec{x},\mathsf{ID}} = \dfrac{\langle \vec{s},\vec{x} \rangle}{\langle \vec{t},\vec{\theta}_{\mathsf{ID}} \rangle}$ and define $\mathsf{sk}_{\vec{x},\mathsf{ID}} = g_2^{\mathsf{tk}_{\vec{x},\mathsf{ID}}}$ for $\vec{\theta}_{\mathsf{ID}}$.

$\mathsf{Encrypt}(\mathsf{PK}, \vec{y})$**:** Takes as input the public key PK and a message $\vec{y} = (y_1, \ldots, y_k) \in \mathbb{Z}_q^k$. To encrypt $\vec{y}$, sample $r \xleftarrow{\$} \mathbb{Z}_q$ and compute

$$\mathsf{CT} = (H_1^r G^{y_1}, \ldots, H_k^r G^{y_k}, b_1^r, \ldots, b_k^r).$$

$\mathsf{Decrypt}(\mathsf{PK}, \mathsf{sk}_{\vec{x},\mathsf{ID}}, \mathsf{CT})$**:** Takes as input the public key PK, the secret key $\mathsf{sk}_{\vec{x},\mathsf{ID}} = g_2^{\mathsf{tk}_{\vec{x},\mathsf{ID}}}$ for $\vec{\theta}_{\mathsf{ID}} = (\theta_1, \ldots, \theta_k)$ and a ciphertext CT, the algorithm computes

$$E = \frac{\Big(H_1^r G^{y_1}\Big)^{x_1} \cdots \Big(H_k^r G^{y_k}\Big)^{x_k}}{e\Big(\Big(b_1^r\Big)^{\theta_1} \cdots \Big(b_k^r\Big)^{\theta_k}, g_2^{\mathsf{tk}_{\vec{x},\mathsf{ID}}}\Big)}.$$

Finally, it returns the discrete logarithm of E in basis $G = e(g_1, g_2)$.

Correctness: For all $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k)$, all $\vec{y} \in \mathbb{Z}_q^k$ and $\vec{x} \in \mathbb{Z}_p^k$, for $\mathsf{sk}_{\vec{x},\mathsf{ID}} = (g_2^{\mathsf{tk}_{\vec{x},\mathsf{ID}}}, \vec{\theta}_{\mathsf{ID}}) \leftarrow \mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, \vec{x})$ and $\mathsf{CT} \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \vec{y})$, we have that

$$\begin{aligned}
\frac{\Big(H_1^r G^{y_1}\Big)^{x_1} \cdots \Big(H_k^r G^{y_k}\Big)^{x_k}}{e\Big(\Big(b_1^r\Big)^{\theta_1} \cdots \Big(b_k^r\Big)^{\theta_k}, g_2^{\mathsf{tk}_{\vec{x},\mathsf{ID}}}\Big)} &= \frac{G^{\langle \vec{x},\vec{y} \rangle} \cdot \Big(G^{x_1 s_1 + \cdots + x_k s_k}\Big)^r}{e\Big(g_1^{t_1 r \theta_1} \cdots g_1^{t_k r \theta_k}, g_2^{\frac{\langle \vec{s},\vec{x} \rangle}{\langle \vec{t},\vec{\theta}_{\mathsf{ID}} \rangle}}\Big)} \\
&= \frac{G^{\langle \vec{x},\vec{y} \rangle} \cdot \cancel{G^{r\langle \vec{s},\vec{x} \rangle}}}{\cancel{e(g_1, g_2)^{r\langle \vec{s},\vec{x} \rangle}}} G^{\langle \vec{x},\vec{y} \rangle} = e(g_1, g_2)^{\langle \vec{x},\vec{y} \rangle}.
\end{aligned}$$

5.3 Security Analysis

5.3.1 Semantic Security

Theorem 5.1

Under the BDDH assumption, the above $\mathcal{T} - \mathcal{FE}$ achieves the selective security

(sel−IND−CPA).

Proof

We assume that there exists an adversary $\mathcal{A}$ can distinguish distributions of ciphertexts in the real game with non-negligible advantage. We build a simulator $\mathcal{B}$ that solves the BDDH problem. It means that $\mathcal{B}$ takes as input a tuple $\left(g_1^a, g_1^b, g_2^a, g_2^c, T\right) \in \mathbb{G}_1^2 \times \mathbb{G}_2^2 \times \mathbb{G}_T$, it must decide whether the input is BDDH tuple where $T = e\left(g_1, g_2\right)^{abc}$ or random tuple where $T = e\left(g_1, g_2\right)^z$. We set

$$\begin{aligned} \mathcal{D}_0 &= \left\{ \left(g_1^a, g_1^b, g_2^a, g_2^c, e\left(g_1, g_2\right)^{abc}\right) | a, b, c \xleftarrow{\$} \mathbb{Z}_q \right\} \\ \mathcal{D}_1 &= \left\{ \left(g_1^a, g_1^b, g_2^a, g_2^c, e\left(g_1, g_2\right)^z\right) | a, b, c, z \xleftarrow{\$} \mathbb{Z}_q \right\}. \end{aligned}$$

The algorithm $\mathcal{B}$ progresses as follows:

- Firstly, $\mathcal{B}$ is provided two distinct messages $\vec{y}_0$ and $\vec{y}_1$.
- $\mathcal{B}$ chooses $\mathbf{\Gamma} = \{\vec{\theta_1}, \ldots, \vec{\theta_n}\}$ is a linear code of size n and length k, as well as $t_1, \ldots, t_k \xleftarrow{\$} \mathbb{Z}_q$. Set $\vec{t} = (t_1, \ldots, t_k)$ and $b_i = g_1^{t_i}$, for $i = 1$ to k.
- $\mathcal{B}$ finds a $(k-1)$-basis of subspace $(\vec{y}_0 - \vec{y}_1)^\perp$ because the adversary $\mathcal{A}$ can only ask secret keys for vectors $\vec{x}$ in $(\vec{y}_0 - \vec{y}_1)^\perp$. We denote this basis by $(\vec{z}_1, \ldots, \vec{z}_{k-1})$. For $i = 1, \ldots, k-1$, $\mathcal{B}$ randomly chooses $u_i \xleftarrow{\$} \mathbb{Z}_q$.
- We consider the canonical basis $(\vec{e}_1, \ldots, \vec{e}_k)$ of $\mathbb{Z}_q^k$. A linear transformation from basis $(\vec{z}_1, \ldots, \vec{z}_{k-1}, (\vec{y}_0 - \vec{y}_1))$ to $(\vec{e}_1, \ldots, \vec{e}_k)$ is given by: $\vec{e}_i = \alpha_i \left(\vec{y}_0 - \vec{y}_1\right) + \sum_{j=1}^{k-1} \lambda_{i,j} \vec{z}_j$, where the coefficients $\alpha_i, \lambda_{i,j}$ can be found efficiently by $\mathcal{B}$. Note that $\langle \vec{e}_i, \vec{y}_0 - \vec{y}_1 \rangle = \alpha_i \times ||\vec{y}_0 - \vec{y}_1||^2$. Then $\vec{\alpha} = \sum_i \alpha_i \vec{e}_i = 1/||\vec{y}_0 - \vec{y}_1||^2 \times \sum_i \langle \vec{e}_i, \vec{y}_0 - \vec{y}_1 \rangle \vec{e}_i = 1/||\vec{y}_0 - \vec{y}_1||^2 \times (\vec{y}_0 - \vec{y}_1)$.
- From the challenge tuple, and random scalars $u_1, \ldots, u_{k-1} \xleftarrow{\$} \mathbb{Z}_q$, set $G = e(g_1, g_2^c)$ and

$$\begin{aligned} H_i &= e\left(\left(g_1^a\right)^{\alpha_i} \cdot g_1^{\sum_{j=1}^{k-1} u_j \lambda_{i,j}}, g_2^c \right) \\ &= e(g_1, g_2^c)^{a\alpha_i + \sum_{j=1}^{k-1} u_j \lambda_{i,j}} = G^{a\alpha_i + \sum_{j=1}^{k-1} u_j \lambda_{i,j}} \end{aligned}$$

  for $i = 1, \ldots, k$, which implicitly defines $s_i = a\alpha_i + \sum_{j=1}^{k-1} u_j \lambda_{i,j}$. The public key is set to $\mathsf{PK} = \left(\mathbb{G}, \mathbf{\Gamma}, g_1, g_2, H_1, \ldots, H_k, b_1, \ldots, b_k\right)$.
- For any vector $\vec{x} = (x_1, \ldots, x_k) \in (\vec{y}_0 - \vec{y}_1)^\perp$, $\mathcal{B}$ computes $\kappa_{\vec{x}} = \langle \vec{s}, \vec{x} \rangle = \sum_{j=1}^{k-1} \sum_{i=1}^{k} x_i u_j \lambda_{i,j}$ and, for identities ID, $\mathsf{sk}_{\vec{x},\mathsf{ID}} = g_2^{\mathsf{tk}_{\vec{x},\mathsf{ID}}}$, where $\mathsf{tk}_{\vec{x},\mathsf{ID}} = \frac{\kappa_{\vec{x}}}{\langle \vec{t}, \vec{\theta}_{\mathsf{ID}} \rangle}$. It sends the value $g_2^{\mathsf{tk}_{\vec{x},\mathsf{ID}}}$ to $\mathcal{A}$. Vector θ_{ID} is a codeword in $\mathbf{\Gamma}$.

- The challenger randomly picks $\beta \xleftarrow{\$} \{0,1\}$ and, from the challenge tuple where T is the last element in $\mathbb{G}_T$, gives $\mathcal{A}$ a ciphertext $\mathsf{CT} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_{2k})$, where $\mathsf{ct}_j = T^{\alpha_j} \cdot e\Big(\left(g_1^b\right)^{\sum_{i=1}^{k-i} u_i \lambda_{j,i}}, g_2^c\Big) \cdot G^{y_{\beta,j}}$ and $\mathsf{ct}_{j+k} = \left(g_1^b\right)^{t_j}$, for $j = 1, \ldots, k$.

- At the end, the adversary outputs his guess β' for β. If $\beta' = \beta$ then $\mathcal{B}$ returns 1 for "BDDH tuple". Otherwise returns 0 for "random tuple". We will show that $\mathcal{B}$ can break BDDH assumption. To do so, we need to prove that the difference below is negligible

$$\begin{aligned} & \Big|\Pr[\mathcal{B}(\mathcal{D}_0) = 1] - \Pr[\mathcal{B}(\mathcal{D}_1) = 1]\Big| \\ = & \Big|\Pr[\beta = \beta' \mid T = e\left(g_1, g_2\right)^{abc}] - \Pr[\beta = \beta' \mid T = e\left(g_1, g_2\right)^{z}]\Big|. \end{aligned}$$

We find that:

1. When $T = e\left(g_1, g_2\right)^{abc}$ then we have $\mathsf{ct}_j = T^{\alpha_j} \cdot e\Big(\left(g_1^b\right)^{\sum_{i=1}^{k-1} u_i \lambda_{j,i}}, g_2^c\Big) \cdot G^{y_{\beta,j}} = H_j^b G^{y_{\beta,j}}$, for $j = 1, \ldots, k$. Therefore

$$\mathsf{CT} = \Big(H_1^b G^{y_{\beta,1}}, \ldots, H_k^b G^{y_{\beta,k}}, b_1^b, \ldots, b_k^b\Big).$$

   It implies that $\mathcal{B}$ perfectly simulates the real game. Since $\mathcal{A}$ can break the semantic security with non-negligible probability, we have $\Pr[\beta = \beta' \mid T = e\left(g_1, g_2\right)^{abc}] = \mathsf{Adv}(\mathcal{A}) + 1/2$.

2. When $T = e(g_1, g_2)^z = G^v$ is random element, the challenger will send $\mathcal{A}$ the ciphertext of message $\vec{y}_\beta + v\vec{\alpha} = \vec{y}_\beta + v/||\vec{y}_0 - \vec{y}_1||^2 \times (\vec{y}_0 - \vec{y}_1) = \mu\vec{y}_0 + (1-\mu)\vec{y}_1$, for some random $\mu \in \mathbb{Z}_q$. This makes β perfectly unpredictable: $\Pr[\beta = \beta' \mid T = e\left(g_1, g_2\right)^z] = 1/2$.

We conclude the advantage is non-negligible as

$$\begin{aligned} & \Big|\Pr[\beta = \beta' \mid T = e\left(g_1, g_2\right)^{abc}] - \Pr[\beta = \beta' \mid T = e\left(g_1, g_2\right)^{z}]\Big| \\ = & \Big|\mathsf{Adv}(\mathcal{A}) + \frac{1}{2} - \frac{1}{2}\Big| = \mathsf{Adv}(\mathcal{A}). \end{aligned}$$

■

This section will be devoted to present a black-box confirmation traitor-tracing algorithm. The purpose of this algorithm is to verify sets of secret keys which are suspected by a Tracer. The tracing algorithm takes as input the master secret key MSK and it can access the table $\mathcal{T}$ (see the Tracing security game) to take a set of secret keys for which it wants to check its suspicion. We will use the scalar form $\mathsf{tk}_{\vec{x},\mathsf{ID}}$ of the secret keys instead of the group element form $\mathsf{sk}_{\vec{x},\mathsf{ID}}$. But as we only consider possible legitimate secrete keys in this form, the scalars are known to the authority.

Suppose that Tracer is provided a set of t secret keys (for the suspected traitors), say $\mathcal{K}_{\mathsf{suspect}} = \{\mathsf{tk}_1, \ldots, \mathsf{tk}_t\}$ which are derived from a fixed vector $\vec{x} = (x_1, \ldots, x_k)$. Here, we have slightly abused the notation, as we are in the one-target security. When

the vector $\vec{x}$ is explicit, we use the notation $\{\mathsf{tk}_1, \ldots, \mathsf{tk}_t\}$ instead of $\{\mathsf{tk}_{\vec{x},1}, \ldots, \mathsf{tk}_{\vec{x},t}\}$, the pirate decoder $\mathcal{D}_{\vec{x}}$ is replaced by $\mathcal{D}$ and we use integers to represent identities of users. A codeword will be $\vec{\theta_i}$ which is attached to a user with identity i. The goal of the Tracer is to verify whether there is any traitor in $\mathcal{K}_{\mathsf{suspect}}$. Before go further we need to define some notations.

- Set $\mathcal{K}_i = \{\mathsf{tk}_1, \ldots, \mathsf{tk}_i\} \subseteq \mathcal{K}_{\mathsf{suspect}}$, for all $i \in [t]$ and $\mathcal{K}_0 = \emptyset$.
- We define spaces of tracing signals (ciphertexts) $\mathsf{Tr}_0, \mathsf{Tr}_1, \ldots, \mathsf{Tr}_t$ such that each signal from Tr_i can be decrypted successfully by any secret key in $\mathcal{K}_i$. More concretely, for each i from 0 to t, the tracing signal for a message $\vec{y} = (y_1, \ldots, y_k)$ is taken from the distribution $\mathsf{Tr}_i^{\vec{x}}(\vec{y})$ (or $\mathsf{Tr}_i(\vec{y})$ for simplicity, when $\vec{x}$ is explicit) that is defined as follows

$$\left\{ \left(H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, g_1^{z_1}, \ldots, g_1^{z_k} \right) \;\middle|\; \begin{array}{l} a \xleftarrow{\$} \mathbb{Z}_q, \vec{z} \xleftarrow{\$} \mathbb{Z}_q^k, \\ \langle \vec{z}, \mathsf{tk}_j \vec{\theta_j} \rangle = a \langle \vec{s}, \vec{x} \rangle, \forall j \in [i] \end{array} \right\},$$

  where $\vec{z} = (z_1, \ldots, z_k)$. $G, H_1, \ldots, H_k$ are group elements of $\mathbb{G}_T$ and belong to the public key PK. Set $\mathcal{Q}(a) = e(g_1, g_2)^{a\langle \vec{s}, \vec{x} \rangle}$, as $\vec{s}$ and $\vec{x}$ are fixed.
- Every user j with secret key in $\mathcal{K}_i$ can output the same

$$\frac{(H_1^a G^{y_1})^{x_1} \cdots (H_k^a G^{y_k})^{x_k}}{e(g_1^{\langle \vec{z}, \vec{\theta_j} \rangle}, g_2^{\mathsf{tk}_j})} = \frac{\mathcal{P}(\vec{y}, a)}{\mathcal{Q}(a)},$$

  where $\mathcal{P}(\vec{y}, a) = (H_1^a G^{y_1})^{x_1} \cdots (H_k^a G^{y_k})^{x_k} = \mathcal{Q}(a) \times G^{\langle \vec{y}, \vec{x} \rangle}$.
- Define distribution of normal ciphertext for a message $\vec{y} = (y_1, \ldots, y_k)$, denoted $\mathsf{Norm}(\vec{y})$: randomly draw $r \xleftarrow{\$} \mathbb{Z}_q$ and output ciphertext $(H_1^r G^{y_1}, \ldots, H_k^r G^{y_k}, b_1^r, \ldots, b_k^r)$.
- For $i = 0, \ldots, t$, we set $p_i = \Pr[\mathcal{D}(\mathsf{CT}) = b \,|\, b \xleftarrow{\$} \{0,1\}, \mathsf{CT} \leftarrow \mathsf{Tr}_i(\vec{y}_b)]$, where $\vec{y}_0, \vec{y}_1$ are chosen by $\mathcal{D}$. When $i = 0$, in $\mathsf{Tr}_i(\vec{y}_b)$, a and $\vec{z}$ are perfectly independent, and so under the DDH assumption, the H_i^a hides the $y_{b,i}$. So we have $p_0 = 1/2 + \mathsf{negl}(\lambda)$.

Definition 5.4

A tracing traitor algorithm is black-box confirmation if it satisfies:

1. Confirmation: If suspected set of users actually contains the entire set of traitors then output of Tracing algorithm always returns at least an identity i such that $\mathsf{tk}_i \in \mathcal{K}_{\mathsf{suspect}}$ is guilty. Formally, with the condition $\mathcal{K}_{\mathcal{D}} \subseteq \mathcal{K}_{\mathsf{suspect}}$, the Tracing algorithm returns at least an identity i such that the secret key $\mathsf{tk}_i \in \mathcal{K}_{\mathsf{suspect}}$ as guilty. We denote by $\mathcal{K}_{\mathcal{D}}$ a set of secret keys used to build the pirate decoder $\mathcal{D}$.
2. Soundness: The honest users will never be accused if the Tracing algorithm outputs an identity as guilty; it is impossible for traitors to deceive Tracing algorithm to blame innocent users. Said differently if Tracing algorithm outputs an identity i such that tk_i is guilty then $\mathsf{tk}_i \in \mathcal{K}_{\mathcal{D}}$.

5.3.2 Security of Tracing Algorithm

The tracing algorithm needs to use the following lemmas.

Lemma 5.1

Under the DDH assumption in $\mathbb{G}_1$, no adversary corruping t users $1, \ldots, t$ can distinguish the distribution of tracing signals $\mathsf{Tr}_t(\vec{y})$ with the distribution of normal ciphertexts $\mathsf{Norm}(\vec{y})$, for any adversarially chosen $\vec{y}$.

Proof

Suppose that an adversary $\mathcal{A}$ can distinguish the distribution of tracing signals $\mathsf{Tr}_t(\vec{y})$ with the distribution of normal ciphertexts Norm. We will build a simulator $\mathcal{B}$ breaks the DDH assumption in $\mathbb{G}_1$. The simulator has inputs: 4-tuples $(\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) \in \mathbb{G}_1^4$, where $\mathfrak{g}_2 = \mathfrak{g}_1^c$ and c is unknown. It decides whether this is a DDH tuple or a random tuple:

1. Take randomly t codewords $\vec{\theta}_1, \ldots, \vec{\theta}_t$ from the code $\mathbf{\Gamma}$.
2. Take randomly A from $\mathbb{Z}_q$ such that $\mathfrak{g}_1^A \mathfrak{g}_2 \neq 1$.
3. Take randomly $\vec{a} = (a_1, \ldots, a_k), \vec{e} = (e_1, \ldots, e_k) \xleftarrow{\$} \mathbb{Z}_q^k$ such that $\langle \vec{\theta}_i, \vec{a} - A\vec{e} \rangle = 0$, for all $i = 1, \ldots, t$.
4. Set $b_i = \mathfrak{g}_1^{a_i} \mathfrak{g}_2^{e_i}$, for all $i = 1, \ldots, k$.
5. Take randomly $\vec{\alpha} = (\alpha_1, \ldots, \alpha_k) \xleftarrow{\$} \mathbb{Z}_q^k$ such that $\langle \vec{\alpha}, \vec{a} - A\vec{e} \rangle = 0$. Take randomly $g_2 \xleftarrow{\$} \mathbb{G}_2$ and it sets $g_1 = \mathfrak{g}_1^A \mathfrak{g}_2$, $G = e(g_1, g_2)$. We set $H_i = e(u_1^A u_2, g_2)^{\alpha_i}$ for all $i \in [k]$. The public key is $\mathsf{PK} = (\mathbb{G}, \mathbf{\Gamma}, g_1, g_2, G, H_1, \ldots, H_k, b_1, \ldots, b_k)$, where $\mathbb{G}$ is a bilinear group.
6. The simulator $\mathcal{B}$ calculates secret key for queries $(\vec{x}, i)$, $\mathsf{tk}_{\vec{x},i} = \dfrac{\langle \vec{\alpha}, \vec{x} \rangle}{\langle \vec{\theta}_i, \vec{e} \rangle}$, for $i \in [t]$ and functions $\vec{x}$ then gives all $g_2^{\mathsf{tk}_{\vec{x},i}}$ to the adversary $\mathcal{A}$. It is clear that $\mathsf{tk}_{\vec{x},i} \vec{\theta}_i$ is a representation of $(\mathfrak{g}_1^A \mathfrak{g}_2)^{\langle \vec{\alpha}, \vec{x} \rangle}$ in the base $(b_1, \ldots, b_k)$.
7. Take randomly $a \xleftarrow{\$} \mathbb{Z}_q$. The simulator constructs the ciphertext for a message $\vec{y}$ as below

$$\mathsf{CT} = (H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, (u_1^{a_1} u_2^{e_1})^a, \ldots, (u_1^{a_k} u_2^{e_k})^a),$$

   where $\vec{y} = (y_1, \ldots, y_k)$.
8. Send the ciphertext CT to the adversary $\mathcal{A}$. If $\mathcal{A}$ decides the ciphertext comes from normal distribution (i.e. $\mathcal{A}$ returns 1) then $\mathcal{B}$ returns "DDH tuple", else returns "random tuple".

We first show that the public key PK which is generated by the simulator $\mathcal{B}$ is indistinguishable from the corresponding public key in the real algorithm.

- We will prove that distribution of tuples $(b_1, \ldots, b_k) \in \mathbb{G}_1^k$ is uniform. Indeed, , write $b_i = g_1^{t_i}$ then, for each $(t+k)$-tuple $(\vec{0}, t_1, \ldots, t_k)$ where $t_1, \ldots, t_k \xleftarrow{\$} \mathbb{Z}_q$ and $\vec{0} = (0, \ldots, 0) \in \mathbb{Z}_q^t$ the below system of equations has a solution

$$\left(\begin{array}{ccc|ccc} \ldots & \vec{\theta}_1 & \ldots & \ldots & -A\vec{\theta}_1 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \ldots & \vec{\theta}_t & \ldots & \ldots & -A\vec{\theta}_t & \ldots \\ \hline 1 & \ldots & 0 & c & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 1 & 0 & \ldots & c \end{array}\right) \times \left(\begin{array}{c} \vec{a} \\ \hline \vec{e} \end{array}\right) = \left(\begin{array}{c} \vec{0} \\ \hline t_1 \\ \vdots \\ t_k \end{array}\right).$$

  We denote by $\mathbf{\Gamma}_0$ a matrix with its rows are vectors $\vec{\theta}_1, \ldots, \vec{\theta}_t$. The rank of this matrix is t.

  Indeed, it is equivalent that

$$\begin{pmatrix} \mathbf{\Gamma}_0 & -A\mathbf{\Gamma}_0 \\ \mathbf{I}_k & c\mathbf{I}_k \end{pmatrix} \times \begin{pmatrix} \vec{a} \\ \vec{e} \end{pmatrix} = \begin{pmatrix} \vec{0} \\ \vec{t} \end{pmatrix}$$

  has solutions. Here $\mathbf{I}_k$ is the $(k \times k)$-unit matrix. We set

$$\mathbf{\Omega} = \begin{pmatrix} \mathbf{\Gamma}_0 & -A\mathbf{\Gamma}_0 \\ \mathbf{I}_k & c\mathbf{I}_k \end{pmatrix}.$$

  Since A is chosen such that $1 \neq \mathfrak{g}_1^A \mathfrak{g}_2 = \mathfrak{g}_1^{A+c}$, $((t+k) \times 2k)$-matrix $\mathbf{\Omega}$ has rank $k+t$. Therefore, $\dim \mathsf{Im}\mathbf{\Omega} = \operatorname{rank} \mathbf{\Omega}$ and the dimension of $\mathsf{Ker}\mathbf{\Omega} = 2k - (k+t) = k - t$. Therefore, the above system of linear equations with unkowns $(\vec{a}, \vec{e})$ exists a solution. It implies that $(b_1, \ldots, b_k)$ is uniform over $\mathbb{G}_1^k$.

- Concerning H_i, in the real game $H_i = e(g_1, g_2)^{\alpha_i}$ for randomly chosen but known g_1, g_2 while in the simulation game, $H_i = e(u_1^A u_2, g_2)^{\alpha_i}$ for randomly chosen A and $(\alpha_i)_i$ in a span of dimension $k-1$. Under the DDH in the $\mathbb{G}_1$, $u_1^A u_2$ is indistinguishable from random, and thus H_i follows from a correct distribution in the computational sense.

We now show that, for any adversarially chosen $\vec{y}$, if $(\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) \in \mathbb{G}_1^4$ is a DDH tuplee then the ciphertext is a normal ciphertext of $\vec{y}$ and when it is a random tuple then the ciphertext comes from $\mathsf{Tr}_t(\vec{y})$. Therefore, if the adversary can distinguish these two distributions then $\mathcal{B}$ can break the DDH assumption in $\mathbb{G}_1$: $|\Pr[\mathcal{B}(\mathcal{D}_0) = 1] - \Pr[\mathcal{B}(\mathcal{D}_1) = 1]|$ is non-negligible. By definition, it is equivalent to

$$\left|\Pr[\mathcal{A}(\mathsf{CT}) = 1 \mid (\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) \xleftarrow{\$} \mathcal{D}_0] - \Pr[\mathcal{A}(\mathsf{CT}) = 1 \mid (\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) \xleftarrow{\$} \mathcal{D}_1]\right|$$

is non-negligible. Here, CT is a ciphertext generated as in Step 7. We find that:

1. When $(\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) \xleftarrow{\$} \mathcal{D}_0$, we will prove that

   $$\Pr[\mathcal{A}(\mathsf{CT}) = 1 \mid (\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) \xleftarrow{\$} \mathcal{D}_0] = \Pr[\mathcal{A}(\mathsf{CT}) = 1 \mid \mathsf{CT} \xleftarrow{\$} \mathsf{Norm}].$$

   Indeed, suppose that $(\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) = (\mathfrak{g}_1, \mathfrak{g}_2, \mathfrak{g}_1^z, \mathfrak{g}_2^z)$, where z is unknown. The ciphertexts in Step 7 is then:

   $$\begin{aligned}
   \mathsf{CT} &= \left(H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, (u_1^{a_1} u_2^{e_1})^a, \ldots, (u_1^{a_k} u_2^{e_k})^a\right) \\
   &= \left(H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, (\mathfrak{g}_1^{za_1} \mathfrak{g}_2^{ze_1})^a, \ldots, (\mathfrak{g}_1^{za_k} \mathfrak{g}_2^{ze_k})^a\right) \\
   &= \left(H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, (\mathfrak{g}_1^{a_1} \mathfrak{g}_2^{e_1})^{z \cdot a}, \ldots, (\mathfrak{g}_1^{a_k} \mathfrak{g}_2^{e_k})^{z \cdot a}\right) \\
   &= \left(H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, b_1^{z \cdot a}, \ldots, b_k^{z \cdot a}\right),
   \end{aligned}$$

   which is in the space of normal ciphertext. It is sufficient thus to show that, with the decryption with the secret key $\mathsf{tk}_{\vec{x},i}$, the decryption will gives $G^{\langle \vec{x}, \vec{y} \rangle}$. Indeed,

   $$\begin{aligned}
   E &= \frac{\left(H_1^a G^{y_1}\right)^{x_1} \cdots \left(H_k^a G^{y_k}\right)^{x_k}}{e\left(\left(u_1^{a_1} u_2^{e_1}\right)^{a\theta_1} \cdots \left(u_1^{a_k} u_2^{e_k}\right)^{a\theta_k}, g_2^{\mathsf{tk}_{\vec{x},i}}\right)} \\
   &= \frac{G^{\langle \vec{x}, \vec{y} \rangle} \cdot e(u_1^A u_2, g_2)^{ax_1\alpha_1} \cdots e(u_1^A u_2, g_2)^{ax_k\alpha_k}}{e\left(\left(\mathfrak{g}_1^A \mathfrak{g}_2\right)^{za\langle \vec{e}, \vec{\theta} \rangle}, g_2^{\frac{\langle \vec{x}, \vec{\alpha} \rangle}{\langle \vec{\theta}, \vec{e} \rangle}}\right)} \\
   &= \frac{G^{\langle \vec{x}, \vec{y} \rangle} \cdot e(\mathfrak{g}_1^A \mathfrak{g}_2, g_2)^{azx_1\alpha_1} \cdots e(\mathfrak{g}_1^A \mathfrak{g}_2, g_2)^{azx_k\alpha_k}}{e\left(\left(\mathfrak{g}_1^A \mathfrak{g}_2\right)^{za\langle \vec{e}, \vec{\theta} \rangle}, g_2^{\frac{\langle \vec{x}, \vec{\alpha} \rangle}{\langle \vec{\theta}, \vec{e} \rangle}}\right)} \\
   &= \frac{G^{\langle \vec{x}, \vec{y} \rangle} \cdot e(\mathfrak{g}_1^A \mathfrak{g}_2, g_2)^{az\langle \vec{x}, \alpha \rangle}}{e\left(\left(\mathfrak{g}_1^A \mathfrak{g}_2\right)^{za\langle \vec{e}, \vec{\theta} \rangle}, g_2^{\frac{\langle \vec{x}, \vec{\alpha} \rangle}{\langle \vec{\theta}, \vec{e} \rangle}}\right)} = G^{\langle \vec{x}, \vec{y} \rangle}.
   \end{aligned}$$

2. When $(\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) \xleftarrow{\$} \mathcal{D}_1$, we will prove that

   $$\Pr[\mathcal{A}(\mathsf{CT}) = 1 \mid (\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) \xleftarrow{\$} \mathcal{D}_1] = \Pr[\mathcal{A}(\mathsf{CT}) = 1 \mid \mathsf{CT} \xleftarrow{\$} \mathsf{Tr}_t].$$

   Indeed, suppose that $(\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) = (\mathfrak{g}_1, \mathfrak{g}_2, \mathfrak{g}_1^{\gamma_1}, \mathfrak{g}_2^{\gamma_2})$, where $\gamma_1 \neq \gamma_2$ and $\mathfrak{g}_2 = \mathfrak{g}_1^c$. The ciphertexts in Step 7 is then:

$$
\begin{aligned}
\mathsf{CT} &= \left(H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, (u_1^{a_1} u_2^{e_1})^a, \ldots, (u_1^{a_k} u_2^{e_k})^a\right) \\
&= \left(H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, (\mathfrak{g}_1^{\gamma_1 a_1} \mathfrak{g}_2^{\gamma_2 e_1})^a, \ldots, (\mathfrak{g}_1^{\gamma_1 a_k} \mathfrak{g}_2^{\gamma_2 e_k})^a\right) \\
&= \left(H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, \mathfrak{g}_1^{a(\gamma_1 a_1 + c\gamma_2 e_1)}, \ldots, \mathfrak{g}_1^{a(\gamma_1 a_k + c\gamma_2 e_k)}\right) \\
&= \left(H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, \mathfrak{g}_1^{z_1}, \ldots, \mathfrak{g}_1^{z_k}\right),
\end{aligned}
$$

where $z_i = a(\gamma_1 a_k + c\gamma_2 e_k)$ for all $i \in [k]$.

We show that for any traitor with the key $\mathsf{tk}_{\vec{x},i}$, $i = 1$ to t, it decrypts to the same message. Indeed:

$$
\begin{aligned}
E &= \frac{\left(H_1^a G^{y_1}\right)^{x_1} \cdots \left(H_k^a G^{y_k}\right)^{x_k}}{e\left(\left(\mathfrak{g}_1^{\gamma_1 a_1} \mathfrak{g}_2^{\gamma_2 e_1}\right)^{a\theta_1} \cdots \left(\mathfrak{g}_1^{\gamma_1 a_k} \mathfrak{g}_2^{\gamma_2 e_k}\right)^{a\theta_k}, g_2^{\mathsf{tk}_{\vec{x},i}}\right)} \\
&= \frac{G^{\langle \vec{x}, \vec{y} \rangle} \cdot e(u_1^A u_2, g_2)^{a x_1 \alpha_1} \cdots e(u_1^A u_2, g_2)^{a x_k \alpha_k}}{e\left(\mathfrak{g}_1^{\langle \vec{a}, \vec{\theta} \rangle a \gamma_1} \mathfrak{g}_2^{\langle \vec{e}, \vec{\theta} \rangle a \gamma_2}, g_2^{\frac{\langle \vec{x}, \vec{\alpha} \rangle}{\langle \vec{\theta}, \vec{e} \rangle}}\right)} \\
&= \frac{G^{\langle \vec{x}, \vec{y} \rangle} \cdot e(\mathfrak{g}_1^{A\gamma_1} \mathfrak{g}_2^{\gamma_2}, g_2)^{a x_1 \alpha_1} \cdots e(\mathfrak{g}_1^{A\gamma_1} \mathfrak{g}_2^{\gamma_2}, g_2)^{a x_k \alpha_k}}{e\left(\mathfrak{g}_1^{\langle \vec{e}, \vec{\theta} \rangle a A \gamma_1} \mathfrak{g}_2^{\langle \vec{e}, \vec{\theta} \rangle a \gamma_2}, g_2^{\frac{\langle \vec{x}, \vec{\alpha} \rangle}{\langle \vec{\theta}, \vec{e} \rangle}}\right)} \\
&= \frac{G^{\langle \vec{x}, \vec{y} \rangle} \cdot e(\mathfrak{g}_1^{A\gamma_1} \mathfrak{g}_2^{\gamma_2}, g_2)^{a \langle \vec{x}, \alpha \rangle}}{e\left(\left(\mathfrak{g}_1^{A\gamma_1} \mathfrak{g}_2^{\gamma_2}\right)^{a \langle \vec{e}, \vec{\theta} \rangle}, g_2^{\frac{\langle \vec{x}, \vec{\alpha} \rangle}{\langle \vec{\theta}, \vec{e} \rangle}}\right)} = \frac{G^{\langle \vec{x}, \vec{y} \rangle} \cdot e(\mathfrak{g}_1^{A\gamma_1} \mathfrak{g}_2^{\gamma_2}, g_2)^{a \langle \vec{x}, \alpha \rangle}}{e\left(\mathfrak{g}_1^{A\gamma_1} \mathfrak{g}_2^{\gamma_2}, g_2\right)^{a \langle \vec{x}, \vec{\alpha} \rangle}} = G^{\langle \vec{x}, \vec{y} \rangle}.
\end{aligned}
$$

Here $\vec{\theta_i} = (\theta_1, \ldots, \theta_k)$.

Finally, we will prove that the distribution of ciphertext CT is uniform over the space of signals Tr_t. It requires that the system of equations

$$
\begin{pmatrix}
\mathbf{\Gamma}_0 & -A\mathbf{\Gamma}_0 \\
\mathbf{I}_k & c\mathbf{I}_k \\
a\gamma_1 \mathbf{I}_k & ac\gamma_2 \mathbf{I}_k
\end{pmatrix} \times \begin{pmatrix} \vec{a} \\ \vec{e} \end{pmatrix} = \vec{\gamma}
$$

is consistent, where $\vec{\gamma}$ is a fixed vector in $\mathbb{Z}_q^{t+2k}$. It is equivalent that the

below $(t+2k, 2k)$-matrix

$$\left(\begin{array}{ccc|ccc} \dots & \vec{\theta}_1 & \dots & \dots & -A\vec{\theta}_1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \dots & \vec{\theta}_t & \dots & \dots & -A\vec{\theta}_t & \dots \\ \hline 1 & \dots & 0 & c & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 1 & 0 & \dots & c \\ \hline a\gamma_1 & \dots & 0 & ac\gamma_2 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & a\gamma_1 & 0 & \dots & ac\gamma_2 \end{array}\right)$$

has full rank (i.e. rank $= 2k$). Indeed, This is straightforward because the last $2k$ rows of the above matrix are linear independent due to $\gamma_1 \neq \gamma_2$.

We conclude that

$$\begin{aligned} &\Big|\Pr\ [\ \ \mathcal{B}(\mathcal{D}_0) = 1] - \Pr[\mathcal{B}(\mathcal{D}_1) = 1]\Big| \\ &\quad = \ |\Pr[\mathcal{A}(\mathsf{CT}) = 1 \mid \mathsf{CT} \xleftarrow{\$} \mathsf{Normal}] - \Pr[\mathcal{A}(\mathsf{CT}) = 1 \mid \mathsf{CT} \xleftarrow{\$} \mathsf{Tr}_t]| = \mathsf{Adv}(\mathcal{A}), \end{aligned}$$

which is non-negligible. ■

Lemma 5.2: Hybrid Lemma

Considering the one-target security for an adversarially chosen target function $\vec{x}$. Under the DDH assumption over group $\mathbb{G}_1$, for all $1 \leq i_0 \leq t$, no adversary can distinguish the distribution of tracing signals $\mathsf{Tr}_{i_0}(\vec{y})$ with the distribution of $\mathsf{Tr}_{i_0-1}(\vec{y})$ unless it owns the secret key sk_{i_0}.

Proof

Suppose that an adversary $\mathcal{A}$ can distinguish the distribution of tracing signals Tr_{i_0} with Tr_{i_0-1}. We build a simulator $\mathcal{B}$ that breaks DDH assumption. The simulator has input a 4-tuple $(\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) \in \mathbb{G}_1^4$, where $\mathfrak{g}_2 = \mathfrak{g}_1^c$ and c is unknown. It must output "DDH tuple" or "random tuple".

1. Take randomly t codewords $\vec{\theta}_1, \dots, \vec{\theta}_t$ from the code $\mathbf{\Gamma}$ corresponding to t traitors and also take t codewords $\vec{\theta}_1^{(s)}, \dots, \vec{\theta}_t^{(s)}$ from the code $\mathbf{\Gamma}$ corresponding to t suspected users. We are considering the adversary $\mathcal{A}$ does not know the secret key sk_{i_0} or the pirate decoder does not contain $\mathsf{sk}_{\vec{x},i_0}$ in itself.

2. Take randomly A from $\mathbb{Z}_q$ such that $\mathfrak{g}_1^A \mathfrak{g}_2 \neq 1$.

3. Take randomly $\vec{a} = (a_1, \dots, a_k), \vec{e} = (e_1, \dots, e_k) \xleftarrow{\$} \mathbb{Z}_q^k$ such that $\langle \vec{\theta}_i, \vec{a} - A\vec{e} \rangle = 0$, for all $i = 1, \dots, t$, $\langle \vec{\theta}_i^{(s)}, \vec{a} - A\vec{e} \rangle = 0$, for all $i = 1, \dots, t$, $i \neq i_0$ and $\langle \vec{\theta}_{i_0}^{(s)}, \vec{a} - A\vec{e} \rangle \neq 0$.

4. Set $b_i = \mathfrak{g}_1^{a_i}\mathfrak{g}_2^{e_i}$, for all $i = 1, \ldots, k$. Take randomly $\vec{v} = (v_1, \ldots, v_k) \xleftarrow{\$} \mathbb{Z}_q^k$ such that $\langle \vec{\theta}_i^{(s)}, \vec{v} \rangle = 0$, for all $i = 1, \ldots, i_0$.

5. Take randomly $\vec{\alpha} = (\alpha_1, \ldots, \alpha_k) \xleftarrow{\$} \mathbb{Z}_q^k$ such that $\langle \vec{\alpha}, \vec{a} - A\vec{e} \rangle = 0$ and $\langle \vec{\alpha}, \vec{v} \rangle = 0$.

6. When $\mathcal{B}$ receives a target function $\vec{x}$ from $\mathcal{A}$. It calculates $\tau_i = \dfrac{\langle \vec{\alpha}, \vec{x} \rangle}{\langle \vec{\theta}_i, \vec{e} \rangle}$, for $i = 1, \ldots, t$ and then give all $g_2^{\tau_i}$ to the adversary $\mathcal{A}$ to create a Pirate Decoder $\mathcal{D}_{\vec{x}}$. Moreover, $\tau_i^{(s)} = \dfrac{\langle \vec{\alpha}, \vec{x} \rangle}{\langle \vec{\theta}_i^{(s)}, \vec{e} \rangle}$, for $i = 1, \ldots, i_0 - 1$. It is clear that $\tau_i \vec{\theta}_i$ and $\tau_i^{(s)} \vec{\theta}_i^{(s)}$ are representations of $(\mathfrak{g}_1^A \mathfrak{g}_2)^{\langle \vec{\alpha}, \vec{x} \rangle}$ in the base $(b_1, \ldots, b_k)$.

7. Take randomly $G, H_1, \ldots, H_k \xleftarrow{\$} \mathbb{G}_T$, $a \xleftarrow{\$} \mathbb{Z}_q$. When the simulator receives a message $\vec{y}$, it constructs the ciphertext

$$\mathsf{CT} = (H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, \mathfrak{g}_1^{v_1} u_1^{a_1} u_2^{e_1}, \ldots, \mathfrak{g}_1^{v_k} u_1^{a_k} u_2^{e_k}),$$

where $\vec{y} = (y_1, \ldots, y_k)$. It then sends the ciphertext to the adversary $\mathcal{A}$. If $\mathcal{A}$ returns the ciphertext comes from $\mathsf{Tr}_{i_0}(\vec{y})$ distribution then $\mathcal{B}$ returns DDH tuple, else returns random tuple.

By the similar argument as in Lemma 5.3.2, the ciphertext CT in Step 7 of the algorithm $\mathcal{B}$ comes from the distribution $\mathsf{Tr}_{i_0}(\vec{y})$ if the input of $\mathcal{B}$ is actually DDH tuples and from the distribution $\mathsf{Tr}_{i_0-1}(\vec{y})$ otherwise. ■

Based on the lemmas 5.3.2 and 5.3.2, we can design a tracing algorithm that relies on the linear technique tracing:

- Initial step: Tracer constructs distributions of tracing signal $\mathsf{Tr}_t, \ldots, \mathsf{Tr}_0$.
- Do experiments on the pirate distinguisher $\mathcal{D}$ finitely many times. We start testing $\mathcal{D}$ by taking tracing signals CT from the distribution Tr_t. We measure the rate that $\mathcal{D}$ outputs correctly his guess, denoted by $\widetilde{p}_t$. Experiments can be done because we can prove that the pirate distinguisher cannot distinguish distributions Tr_t and Norm (see Lemma 5.3.2).
- At step i, for $i = t - 1, \ldots, 0$. We do experiment on the pirate distinguisher $\mathcal{D}$ with tracing signals taken from Tr_i. From Lemma 5.3.2, the pirate distinguisher cannot see any change from previous step $i+1$ to this step i unless it holds the secret key tk_{i+1}. More formally, we also measure the rate $\widetilde{p}_i$ that $\mathcal{D}$ outputs correctly his guess and show that if $\mathcal{D}$ does not contain tk_{i+1} then there is no significant difference between $\widetilde{p}_{i+1}$ and $\widetilde{p}_i$.
- At the final step, $\mathcal{D}$ will be tested with tracing signals taken from Tr_0. $\mathcal{D}$ answers correctly only negligibly close to $1/2$.
- We output the traitor i such that the gap between $\widetilde{p}_i$ and $\widetilde{p}_{i-1}$ is the largest value among all indices i.

Below, we present the tracing algorithm in more details. We note that $\vec{y}_0, \vec{y}_1$ are vectors which are chosen by the pirate distinguisher $\mathcal{D}$.

For $i = t$ downto 0, do the following:

1. Let $\mathsf{cnt} \leftarrow 0$.
2. For $j = 1$ to $N = 8\lambda t^2/\mu$, do the following:
   i. $b \xleftarrow{\$} \{0,1\}$.
   ii. $\mathsf{CT} \xleftarrow{\$} \mathsf{Tr}_i(\vec{y}_b)$.
   iii. Send CT to $\mathcal{D}$. If $\mathcal{D}(\mathsf{CT}) = b$ then $\mathsf{cnt} \leftarrow \mathsf{cnt} + 1$.
3. End for.
4. Let $\widetilde{p}_i$ be the fraction of times that $\mathcal{D}$ did the correct guess. We have $\widetilde{p}_i = \mathsf{cnt}/N$.

End for.

Output identities i such that $|\widetilde{p}_i - \widetilde{p}_{i-1}| \geq \dfrac{\mu(\lambda)}{4t}$.

Below, we state and prove confirmation and soundness property of our Tracing algorithm.

Lemma 5.3: Confirmation property

Under the DDH assumption in $\mathbb{G}_1$, the Tracing algorithm has the confirmation property.

Proof

We want to prove that in the case of that all the traitors are in the set of suspected users, i.e. $\mathcal{K}_{\mathcal{D}} \subseteq \mathcal{K}_{\mathsf{suspect}}$, the Tracing algorithm always returns the identity of a guilty. It means that the output of Tracing algorithm is not empty with high probability. We denote $\mathcal{A}$ an adversary who used the secret keys in $\mathcal{K}_{\mathcal{D}}$ to output the pirate distinguisher $\mathcal{D}$. Since the adversary $\mathcal{A}$ can create a μ-useful pirate distinguisher $\mathcal{D}$, it implies that $\left|p_{\mathsf{Norm}} - \frac{1}{2}\right| \geq \mu(\lambda)$, where

$$p_{\mathsf{Norm}} = \mathsf{Pr}\left[\mathcal{D}\left(\mathsf{CT}_b\right) = b : \begin{array}{c}(\mathsf{MSK},\mathsf{PK}) \leftarrow \mathsf{Setup}(\cdot) \\ \{\mathsf{sk}_i \leftarrow \mathsf{Extract}(i,\mathsf{MSK},\vec{x})\}_{i\in[n]} \\ (\mathcal{D},\vec{y}^0,\vec{y}^1) \leftarrow \mathcal{A}(\mathsf{PK},\{\mathsf{sk}_i\}_{i\in[t]}) \\ \text{st. } \langle\vec{x},\vec{y}^0\rangle \neq \langle\vec{x},\vec{y}^1\rangle \\ b \xleftarrow{\$} \{0,1\}, \mathsf{CT}_b \leftarrow \mathsf{Norm}(\mathsf{PK},\vec{y}^b)\end{array}\right].$$

We denote S the set of indices $i \in [t]$ such that $|p_i - p_{i-1}| > \mu(\lambda)/4t$. The set S is well defined in the sense that $S \neq \emptyset$. Indeed, as we know that p_0 is negligibly close to 1/2, and Lemma 5.3.2 showed that no adversary $\mathcal{A}$ can distinguish the distribution Norm from Tr_t, then $|p_t - p_0| \geq \mu(\lambda) - \mathsf{negl}(\lambda) > \mu(\lambda)/2$. Then, there exists an index i such that $|p_i - p_{i-1}| > \mu(\lambda)/2t$. Thus S is a non empty set. Applying Chernoff bound for all $i \in S$, we have on experimental probabilities

$$\mathrm{Pr}\left[|\widetilde{p}_i - \widetilde{p}_{i-1}| < \frac{\mu(\lambda)}{4t}\right] \leq \mathsf{negl}(\lambda),$$

Therefore, with overwhelming probability, there exists an index i such that

$$|\widetilde{p}_i - \widetilde{p}_{i-1}| \geq \frac{\mu(\lambda)}{4t}.$$

The latter is thus returned with overwhelming probability.

■

Lemma 5.4: Soundness property

Under the DDH assumption in $\mathbb{G}_1$, the Tracing algorithm has the soundness property.

Proof

We now prove the soundness property of Tracing algorithm. Suppose that the Tracing algorithm outputs an identity j, where $\mathsf{tk}_j \in \mathcal{K}_{\mathsf{suspect}}$, we will prove that $\mathsf{tk}_j \in \mathcal{K}_{\mathcal{D}}$. According to Chernoff bound, thanks to $N = 8\lambda t^2/\mu(\lambda)$ to calculate $\widetilde{p}_i$, for all i, we have

$$\Pr\left[|\widetilde{p}_i - p_i| > \frac{\mu(\lambda)}{16t}\right] < 2 \cdot e^{-\lambda/64}.$$

Therefore, with high probability we have $|\widetilde{p}_i - p_i| \leq \mu(\lambda)/16t$, for all $i = 0, \ldots, t$.

By definition, whenever the Tracing algorithm outputs j as a guilty, we have $|\widetilde{p}_j - \widetilde{p}_{j-1}| \geq \mu(\lambda)/4t$, and thus $|p_j - p_{j-1}| \geq \mu(\lambda)/8t$. In other words, the pirate distinguisher can distinguish the two tracing signals Tr_j and Tr_{j-1} with advantage at least $\mu(\lambda)/8t$. It implies that $\mathcal{D}$ contains the secret key tk_j, $\mathsf{tk}_j \in \mathcal{D}$. This follows from the fact that if $\mathcal{D}$ does not know the secret key tk_j, $\mathsf{tk}_j \notin \mathcal{D}$, the two tracing signals Tr_j and Tr_{j-1} are indistinguishable. More concretely, under the hardness of the DDH problem in group $\mathbb{G}_1$, it is impossible for the pirate to distinguish Tr_j and Tr_{j-1} without tk_j. This is stated and proved in Lemma 5.3.2. ■

Theorem 5.2

Under the DDH assumption, our tracing scheme is one-target security in black-box confirmation model.

Proof

We recall that in the black-box confirmation model we will verify a set suspected secret keys $\mathcal{K}_{\mathsf{suspect}} = \{\mathsf{tk}_1, \ldots, \mathsf{tk}_t\}$ which are also derived from the vector $\vec{x}$. We will prove that Tracing algorithm always outputs an identity of a traitor whenever $\mathcal{K}_{\mathcal{D}} \cap \mathcal{K}_{\mathsf{suspect}} \neq \emptyset$. It means that Tracer always wins in the game with the pirate distinguisher $\mathcal{D}$. Indeed, we consider the following two cases:

- In the first case $\mathcal{K}_{\mathcal{D}} \subseteq \mathcal{K}_{\mathsf{suspect}}$. It means that all traitors are in suspicious

set $\mathcal{K}_{\mathsf{suspect}}$. Tracing algorithm will output a guilty identity i by the confirmation property. According to soundness property, the identity is a traitor ($\mathsf{tk}_i \in \mathcal{K}_{\mathcal{D}}$).

- In case $\mathcal{K}_{\mathcal{D}} \not\subseteq \mathcal{K}_{\mathsf{suspect}}$ and $\mathcal{K}_{\mathcal{D}} \cap \mathcal{K}_{\mathsf{suspect}} \neq \emptyset$. Because $\mathcal{K}_{\mathcal{D}} \cap \mathcal{K}_{\mathsf{suspect}} \neq \emptyset$, tracing algorithm will output an identity i so that $\mathsf{sk}_i \in \mathcal{K}_{\mathsf{suspect}}$. It implies i is a traitor ($\mathsf{tk}_i \in \mathcal{K}_{\mathcal{D}}$) by the soundness property. ■

6 Revocable Inner Product Functional Encryption

In this chapter, we are interested in the revocation for functional encryption. In a functional encryption scheme, each functional decryption key is specified by a function and an identity. The objective is to be able to revoke any user from decrypting any function. Such a primitive was called a *revocable functional encryption* in [NWZ16]. Their construction of revocable functional encryption is based on indistinguishability obfuscation, which is a non-standard assumption. So far, there is still no revocable functional encryption, which is built from standard assumption. We thus propose some pairing-based constructions for revocable inner product functional encryption.

Contents

6.1 Revocable Functional Encryption

6.1.1 Motivation

Designing a revocable inner product functional encryption that achieves constant size in both ciphertext and functional key is a challenging problem, unknown even for conventional revoke systems. Depending on the specific practical applications, we will focus on constructions with constant-size functional keys or constant-size ciphertexts.

- In Section 6.2, we give two pairing-based constructions of revocable functional encryption for inner product with short private keys. The constructions are efficient where the size of ciphertext is only $O(r)$ (r is the number of revoked users) and the private key size is constant (only 3 group elements). We use the technique that combines the inner product functional encryption of Abdalla et al. at PKC '15 [ABDP15] with the two-equation technique by Lewko, Sahai, and Waters at SP '11 [LSW10].
- In Section 6.3, by using the n-equation technique of Attrapadung et al. at PKC '10 [AL10], we provide two schemes of revocable functional encryption for inner product with the constant size ciphertext, independent of the number of revokers.

Related and concurrent work. Recently Abdalla et al. [ACGU20] proposed several pairing-based constructions of attribute-based inner product functional encryption. However, their schemes does not imply efficiently revocable inner product functional encryption schemes as the overhead cost depends on the size of the policy of ciphertext.

We observe that revoking all functions at once may not be suitable for some practical situations like in the PayTV context. In the final part, we will discuss the extension of revocable inner product functional encryption to the fine-grained revocation in which we can disable the decryption of functional keys on a selective set of revoked functions instead of all functions for each identity.

6.1.2 Definition

We recall the notion of revocable functional encryption scheme which is introduced by Nishimaki et al. [NWZ16]. This is a functional encryption scheme in which the encryption algorithm can take as input a list $\mathcal{R}$ of identities such that all functional decryption keys of each revoker user will be disabled. We will then extend this notion to the fine-grained revocation in which we can disable the decryption of functional keys on a selective set of revoked functions instead of all functions for each identity.

Definition 6.1

A revocable functional encryption scheme rFE consists of four algorithms

$$(\mathsf{Setup}, \mathsf{Extract}, \mathsf{Encrypt}, \mathsf{Decrypt})$$

which is defined as follows:

$\mathsf{Setup}(1^\lambda)$**:** Takes as input a security parameter λ and outputs a master key pair

$(\mathsf{PK}, \mathsf{MSK})$.

$\mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, F)$**:** Given an identity ID of an user, a circuit $F \in \mathcal{F}_\lambda$ and the master secret key MSK, this algorithm outputs a secret key $\mathsf{sk}_{F,\mathsf{ID}}$.

$\mathsf{Encrypt}(\mathsf{PK}, y, \mathcal{R})$**:** Takes as input the public key PK and a message $y \in \mathcal{Y}_\lambda$, and a list of identities $\mathcal{R}$, this randomized algorithm outputs a ciphertext CT.

$\mathsf{Decrypt}(\mathsf{PK}, \mathsf{sk}_{F,\mathsf{ID}}, \mathsf{CT})$**:** Given the master public key PK, a secret key $\mathsf{sk}_{F,\mathsf{ID}}$ and a ciphertext CT, this algorithm outputs $F(y) \in \mathcal{S}_\lambda$ or an invalid symbol $\perp$.

For correctness, we require that for all $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$, all $y \in \mathcal{Y}_\lambda$, each $F \in \mathcal{F}_\lambda$ and all identities $\mathsf{ID} \notin \mathcal{R}$, $\mathsf{sk}_{F,\mathsf{ID}} \leftarrow \mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, F)$, if $\mathsf{CT} \leftarrow \mathsf{Encrypt}(\mathsf{PK}, y, \mathcal{R})$, then one should get $\mathsf{Decrypt}(\mathsf{PK}, \mathsf{sk}_{F,\mathsf{ID}}, \mathsf{CT}) = F(y)$, with overwhelming probability.

6.1.3 Security

Indistinguishability. We consider the IND security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{B}$ as follows:

Definition 6.2

A revocable functional encryption scheme rFE for a list of identities $\mathcal{R}$,

$$\mathsf{rFE} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Encrypt}, \mathsf{Decrypt})$$

is semantically secure under chosen-plaintext attacks (or $\mathsf{IND-CPA}$ security) if no PPT adversary has non-negligible advantage in the following game:

- The challenger $\mathcal{B}$ runs $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$ and the master public key PK is given to the adversary $\mathcal{A}$.
- The adversary adaptively makes secret key queries to the challenger. That is, the adversary $\mathcal{A}$ chooses some pairs of identities ID and functions $F \in \mathcal{F}_\lambda$. $\mathcal{A}$ sends them to $\mathcal{B}$ and then obtains $\mathsf{sk}_{F,\mathsf{ID}} \leftarrow \mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, F)$ from $\mathcal{B}$.
- The adversary $\mathcal{A}$ chooses distinct messages $y_0, y_1 \in \mathcal{Y}_\lambda$ and a set $\mathcal{R}$ of revoked identities such that:
  1. The set $\mathcal{R} \subseteq \mathcal{C}$, where we denote $\mathcal{C}$ as all identities that were queried by the adversary $\mathcal{A}$.
  2. For each identity $\mathsf{ID} \in (\mathcal{C} - \mathcal{R})$ and every function F which is assigned to ID, the following conditions need to be satisfied:

     $$F(y_0) = F(y_1).$$

     This restriction is required in all functional encryption to avoid trivial attacks.

Whenever $\mathcal{B}$ receives the messages, it randomly picks $\beta \hookleftarrow \{0,1\}$ and then transfers to $\mathcal{A}$ a ciphertext $\mathsf{CT}_\beta = \mathsf{Encrypt}(\mathsf{PK}, y_\beta, \mathcal{R})$.

- Adversary $\mathcal{A}$ eventually returns a guess $\beta^{'}$ for a bit β and wins if $\beta^{'} = \beta$.

Selective security. When the messages y_0, y_1 and a list of revoked users $\mathcal{R}$ are chosen before the Setup algorithm started, then the rFE scheme is said selectively-security against chosen-plaintext attacks, which is denoted by sel−IND−CPA.

6.2 Revocable Functional Encryption for Inner Product with Constant-size Secret Keys

In this section, we present two pairing-based constructions of revocable inner product functional encryption. The first construction is based on MEBDH assumption and it achieves selective security. The second construction is based on simple assumption such as BDDH and DLIN and it achieves adaptive security because it applies the Waters' dual system method [Wat09]. The technique is used in both constructions is based on the idea of applying 2-equation technique [LSW10] for IPFE scheme of Abdalla et al. [ABDP15] and they all obtain short ciphertext ($O(r)$ where r is the number of revoked users) and private keys.

6.2.1 Construction based on q-type Assumption

We will construct a revocable functional encryption for Inner Product with constant-size secret keys.

Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ a bilinear map, where $\mathbb{G}, \mathbb{G}_T$ are cyclic groups of order q, written multiplicatively.

$\mathsf{Setup}(1^\lambda, 1^k)$**:** This algorithm generates a bilinear pairing $(\mathbb{G}, \mathbb{G}_{T'}, q, e)$ for sufficiently large prime order q and g, h are two generators of the group $\mathbb{G}$. The bilinear map e over $(\mathbb{G}, \mathbb{G})$ can be calculated efficiently.

- For each $i \in \{1, \ldots, k\}$, randomly choose $s_i \xleftarrow{\$} \mathbb{Z}_q$. We set $\vec{s} = (s_1, \ldots, s_k)$.
- We take randomly $\alpha, b \xleftarrow{\$} \mathbb{Z}_q$.
- The public key is

$$\mathsf{PK} = \left(\mathbb{G}, g, g^b, g^{b^2}, h^b, e(g,g)^\alpha, e(g,g)^{s_1\alpha}, \ldots, e(g,g)^{s_k\alpha}\right).$$

- The master secret key is $\mathsf{MSK} = \{\vec{s}, \alpha\}$.

$\mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, \vec{x})$**:** Takes as input an identity $\mathsf{ID} \in \mathbb{Z}_q$, the master secret key MSK and a characteristic vector $\vec{x} = (x_1, \ldots, x_k) \in \mathbb{Z}_q^k$. For each identity ID, we take randomly $t \xleftarrow{\$} \mathbb{Z}_q$ and the secret key $\mathsf{sk}_{\vec{x},\mathsf{ID}}$ is a tuple:

$$\mathsf{sk}_0 = g^{\alpha\langle\vec{s},\vec{x}\rangle} g^{b^2 t}, \mathsf{sk}_1 = \left(g^{b\cdot\mathsf{ID}} h\right)^t, \mathsf{sk}_2 = g^{-t}.$$

$\mathsf{Encrypt}(\mathcal{R}, \mathsf{PK}, \vec{y})$: Takes as input the set $\mathcal{R}$ of identities $\{\mathsf{ID}_1, \ldots, \mathsf{ID}_m\}$ that secret keys will be revoked, the public key PK, a message $\vec{y} = (y_1, \ldots, y_k) \in \mathbb{Z}_q^k$. We sample $r \xleftarrow{\$} \mathbb{Z}_q$ chooses random $r_1, \ldots, r_m$ such that $r = r_1 + \ldots + r_m$ and compute

$$\begin{aligned}
\mathsf{CT}_0 &= (\mathsf{C}_1, \mathsf{C}_2, \ldots, \mathsf{C}_k) \\
&= (e(g,g)^{y_1} e(g,g)^{s_1 \alpha r}, \ldots, e(g,g)^{y_k} e(g,g)^{s_k \alpha r}) \\
\mathsf{CT}_1 &= g^r, \\
\mathsf{CT}_{j,1} &= g^{b \cdot r_j}, \\
\mathsf{CT}_{j,2} &= \left(g^{b^2 \cdot \mathrm{ID}_j} h^b\right)^{r_i}.
\end{aligned}$$

for all $j \in [m]$.

$\mathsf{Decrypt}(\mathsf{PK}, \mathsf{sk}_{\vec{x},\mathsf{ID}}, \mathsf{CT})$: In the case $\mathsf{ID} \neq \mathsf{ID}_j$ for all $j \in [m]$, we first compute

$$\begin{aligned}
e(\mathsf{sk}_1, \mathsf{CT}_{j,1})^{\frac{1}{\mathsf{ID}-\mathsf{ID}_j}} \quad & e(\mathsf{sk}_2, \mathsf{CT}_{j,2})^{\frac{1}{\mathsf{ID}-\mathsf{ID}_j}} = \\
&= e\left(\left(g^{b \cdot \mathsf{ID}} h\right)^t, g^{b \cdot r_j}\right)^{\frac{1}{\mathsf{ID}-\mathsf{ID}_j}} e\left(g^{-t}, \left(g^{b^2 \cdot \mathrm{ID}_j} h^b\right)^{r_j}\right)^{\frac{1}{\mathsf{ID}-\mathsf{ID}_j}} \\
&= e(g,g)^{\frac{tb^2 r_j \mathsf{ID}}{\mathsf{ID}-\mathsf{ID}_j}} e(g,g)^{\frac{-tb^2 r_j \mathsf{ID}_j}{\mathsf{ID}-\mathsf{ID}_j}} \\
&= e(g,g)^{tb^2 r_j}.
\end{aligned}$$

Since $r = r_1 + \ldots + r_m$, this implies that

$$A_1 = \prod_{j=1}^{m} \left(e(\mathsf{sk}_1, \mathsf{CT}_{j,1})^{\frac{1}{\mathsf{ID}-\mathsf{ID}_j}} e(\mathsf{sk}_2, \mathsf{CT}_{j,2})^{\frac{1}{\mathsf{ID}-\mathsf{ID}_j}}\right) = e(g,g)^{tb^2 r}.$$

We then computes

$$A_2 = e\left(\mathsf{sk}_0, \mathsf{CT}_1\right) = e\left(g^{\alpha \langle \vec{s}, \vec{x} \rangle} g^{b^2 t}, g^{\cdot r}\right) = e(g,g)^{\alpha r \langle \vec{s}, \vec{x} \rangle} e(g,g)^{tb^2 r}.$$

Finally, we compute

$$\begin{aligned}
\frac{\prod_{i=1}^k \mathsf{C}_i^{x_i}}{A_2 / A_1} &= \frac{\prod_{i=1}^k e(g,g)^{x_i y_i} e(g,g)^{s_i x_i \alpha r}}{e(g,g)^{\alpha r \langle \vec{s}, \vec{x} \rangle}} \\
&= \frac{e(g,g)^{\langle \vec{x}, \vec{y} \rangle} e(g,g)^{\alpha r \langle \vec{s}, \vec{x} \rangle}}{e(g,g)^{r \langle \vec{s}, \vec{x} \rangle}} \\
&= e(g,g)^{\langle \vec{x}, \vec{y} \rangle},
\end{aligned}$$

and output $\langle \vec{x}, \vec{y} \rangle = \log_{e(g,g)} e(g,g)^{\langle \vec{x}, \vec{y} \rangle}$.

Theorem 6.1

Under the decisional q-MEBDH assumption, our construction is sel−IND−CPA.

Proof

Suppose that there exists an adversary $\mathcal{A}$ who breaks the selective security game of our scheme. We will build a simulator $\mathcal{B}$ that breaks the decisional q-MEBDH assumption.

The simulator $\mathcal{B}$ takes as input a q-MEBDH challenge P, T. The simulator then proceeds in the game as follows.

The adversary $\mathcal{A}$ submits a revocation set $\mathcal{R} = \{\mathsf{ID}_1, \ldots, \mathsf{ID}_m\}$, $(m \leq q)$ and two messages $\vec{y}_0, \vec{y}_1 \in \mathbb{Z}_q^k$ to the simulator.

$\mathcal{B}$ finds a basis $\{\mathbf{z}_i\}_{i\in[k-1]}$ of $(\vec{y}_1 - \vec{y}_0)^{\perp}$. For each vector $\mathbf{z}_i$, chooses a random element $d_i \in \mathbb{Z}_p$ as secret key and set $e(g,g)^{d_i}$ as the master public key for $\mathbf{z}_i$ and also, $\mathcal{B}$ sets $e(g,g)^{\alpha}$ as the master public key for $\vec{y}_1 - \vec{y}_0$. Let $\{\vec{e}_1, \ldots, \vec{e}_k\}$ be the canonical basis of $\mathbb{Z}_q^k$. $\mathcal{B}$ finds $c_i, \lambda_{i,j} \in \mathbb{Z}_q$ such that

$$\vec{e}_i = c_i \left(\vec{y}_1 - \vec{y}_0\right) + \sum_{j=1}^{k-1} \lambda_{i,j} \mathbf{z}_j$$

then, sets

$$\gamma_i = \prod_{j=1}^{k-1} e(g,g)^{d_j \lambda_{i,j}} \quad \text{and} \quad e(g,g)^{\alpha s_i} = e(g,g)^{\alpha c_i} \cdot \gamma_i.$$

The simulator now creates the public key PK and gives $\mathcal{A}$ the decryption keys for all identities in $\mathcal{R}$. It will implicitly set b as $a_1 + a_2 + \cdots + a_m$. The simulator first chooses a random $\xi \in \mathbb{Z}_q$.

The public key PK is published as:

$$\left(g, g^b = \prod_{1\leq i\leq m} g^{a_i}, g^{b^2} = \prod_{1\leq i,j\leq m} \left(g^{a_i \cdot a_j}\right), h = \prod_{1\leq i\leq m} \left(g^{a_i}\right)^{-\mathrm{ID}_i} g^{\xi}, e(g,g)^{\alpha}, e(g,g)^{\alpha s_i}\right)$$

We observe that the public parameters are distributed identically to the real system.

Simulate Decryption keys: The simulator $\mathcal{B}$ should answer all decryption keys in the revocation set $\mathcal{R}$ such that the function $\vec{x}$ which is assigned to $\mathsf{ID} \in \mathcal{R}$ must hold

$$\langle \vec{x}, \vec{y}_1 - \vec{y}_0 \rangle = 0.$$

- We have

$$\langle \vec{s}, \vec{x} \rangle = \sum_{j\in[k-1]} \left(\sum_{i\in[k]} x_i \lambda_{i,j} \right) d_j.$$

- To cancel out the term g^{α} which $\mathcal{B}$ doesn't know. $\mathcal{B}$ will chooses randomly an element $z_i \in \mathbb{Z}_q$ and will implicitly set the randomness t_i as

$$t_i = -\alpha \langle \vec{s}, \vec{x} \rangle / a_i^2 + z_i,$$

for each identity ID_i.

- The decryption key $\mathsf{sk}_{\vec{x},\mathsf{ID}_i}$ for ID_i and a function $\vec{x}$ is generated as follows:

$$\begin{aligned}
\mathsf{sk}_0 &= \left(\prod_{\substack{1\le j,k\le n \\ \text{s.t. if } j=k \text{ then } j,k\neq i}} \left(g^{-\langle \vec{s},\vec{x}\rangle \alpha a_j a_k/a_i^2}\right) \right) \prod_{1\le j,k\le n} (g^{a_j a_k})^{z_i} \\
\mathsf{sk}_1 &= \left(\prod_{\substack{1\le j\le n \\ j\neq i}} \left(g^{-\alpha\langle \vec{s},\vec{x}\rangle \cdot a_j/a_i^2}\right)^{(\mathsf{ID}_i-\mathsf{ID}_j)} \left(g^{(\mathsf{ID}_i-\mathsf{ID}_j)\cdot a_j}\right)^{z_i} \right) \left(g^{-\alpha\langle \vec{s},\vec{x}\rangle/a_i^2}\right)^{\xi} g^{\xi z_i} \\
\mathsf{sk}_2 &= g^{\alpha\langle \vec{s},\vec{x}\rangle/a_i^2} g^{-z_i}.
\end{aligned}$$

Simulate Challenge Ciphertext: The simulator receives $\vec{y}_0, \vec{y}_1$ and it picks randomly $\beta \in \{0,1\}$. The simulator then chooses random $r', r'_1, \ldots, r'_m \in \mathbb{Z}_q$ such that $r' = \sum_i r'_i$. This quantity can compute from PK and we set

$$u_i = g^{b^2 \mathsf{ID}_i} h^b.$$

The ciphertext will be encrypted under randomness $\tilde{r} = r + r'$. This randomness will be splited into shares $\tilde{r}_i = a_i r/b + r'_i$. It implies that that $\sum \tilde{r}_i = \tilde{r}$.

The challenge ciphertext will be simulated as follows:

$$\begin{aligned}
\mathsf{CT}_0 &= (\mathsf{C}_1, \mathsf{C}_2, \ldots, \mathsf{C}_k) \\
&= \left(e(g,g)^{y_{\beta,1}} T^{s_1} \cdot e(g,g)^{s_1 \alpha r'}, \ldots, e(g,g)^{y_{\beta,k}} T^{s_k} \cdot e(g,g)^{s_k \alpha r'}\right) \\
\mathsf{CT}_1 &= g^r g^{r'}, \\
\mathsf{CT}_{j,1} &= g^{r a_j} \left(\prod_i g^{a_i}\right)^{r'_i}, \\
\mathsf{CT}_{j,2} &= \left(\prod_{\substack{1\le i\le m \\ i\neq j}} (g^{r a_i a_j})^{\mathsf{ID}_j-\mathsf{ID}_i} \right) (g^{a_j r})^{\xi} u_j^{r'_j}.
\end{aligned}$$

for all $j \in [m]$.

Finally, the adversary $\mathcal{A}$ outputs a guess β' for β. $\mathcal{B}$ outputs 1 to guesses that $T = e(g,g)^{\alpha r}$ if $\beta = \beta'$; otherwise, it outputs 0 to indicate that T is a random in $\mathbb{G}_T$.

When $T = e(g,g)^{\alpha r}$, $\mathcal{B}$ simulates the ciphertext perfectly. We have

$$\Pr\left[\mathcal{B}\left(P, T = e(g,g)^{\alpha r}\right) = 0\right] = \frac{1}{2} + \mathsf{Adv}_{\mathcal{A}}.$$

When $T = R$, we have

$$\Pr[\mathcal{B}(P, T = R) = 0] = \frac{1}{2}.$$

Therefore, $\mathcal{B}$ can break the MEBDH assumption with non-negligible advantage. ■

6.2.2 Construction based on BDDH and DLIN Assumptions

Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ a bilinear map, where $\mathbb{G}, \mathbb{G}_T$ are cyclic groups of order q, written multiplicatively.

$\mathsf{Setup}(1^\lambda, 1^k)$: This algorithm generates a bilinear pairing $(\mathbb{G}, \mathbb{G}_T, q, e)$ for sufficiently large prime order q and g is a generator of the group $\mathbb{G}$. The bilinear map e over $(\mathbb{G}, \mathbb{G})$ can be calculated efficiently.

- We choose random generators $g, v, v_1, v_2, w, h \in G$.
- We pick random exponents $a_1, a_2, b, \alpha_1, \ldots, \alpha_k \in \mathbb{Z}_p$.
- We set $\tau_1 = v v_1^{a_1}, \tau_2 = v v_2^{a_2}$.
- The public key is

  $$\mathsf{PK} = \left(g^b, g^{a_1}, g^{a_2}, g^{ba_1}, g^{ba_2}, \tau_1, \tau_2, \tau_1^b, \tau_2^b, w, h, e(g,g)^{\alpha_1 a_1 b}, \ldots, e(g,g)^{\alpha_k a_1 b}\right).$$

- The master secret key is

  $$\mathsf{MSK} = (g, v, v_1, v_2, \vec{\alpha} = (\alpha_1, \ldots, \alpha_k)).$$

$\mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, \vec{x})$: Takes as input an identity $\mathsf{ID} \in \mathbb{Z}_q$, the master secret key MSK and a characteristic vector $\vec{x} = (x_1, \ldots, x_k) \in \mathbb{Z}_q^k$. For each identity ID, this algorithm chooses random exponents $d_1, d_2, z_1, z_2 \in \mathbb{Z}_p$ and sets $d = d_1 + d_2$. The secret key $\mathsf{sk}_{\vec{x},\mathsf{ID}}$ is the following 8-tuple:

$$D_1 = g^{\langle \vec{\alpha}, \vec{x} \rangle a_1} v^d, D_2 = g^{-\langle \vec{\alpha}, \vec{x} \rangle} v_1^d g^{z_1}, D_3 = \left(g^b\right)^{-z_1}, D_4 = v_2^d g^{z_2},$$
$$D_5 = \left(g^b\right)^{-z_2}, D_6 = g^{d_2 b}, D_7 = g^{d_1}, K = \left(w^{\mathsf{ID}} h\right)^{d_1}.$$

$\mathsf{Encrypt}(\mathcal{R}, \mathsf{PK}, \vec{y})$: Takes as input the set $\mathcal{R}$ of identities $\{\mathsf{ID}_1, \ldots, \mathsf{ID}_r\}$ that secret keys will be revoked, the public key PK, a message $\vec{y} = (y_1, \ldots, y_k) \in \mathbb{Z}_q^k$. The encryption algorithm chooses random exponents $s_1, s_2, t_1, \ldots, t_r$ and sets $s = s_1 + s_2, t = t_1 + \cdots + t_r$ (where $r = |\mathcal{R}|$, the number of revoked users). The ciphertext CT is constructed as:

$$\begin{aligned}
\mathsf{CT}_0 &= (\mathsf{C}_1, \mathsf{C}_2, \ldots, \mathsf{C}_k) \\
&= \left(e(g,g)^{y_1} \left(e(g,g)^{\alpha_1 a_1 b}\right)^{s_2}, \ldots, e(g,g)^{y_k} \left(e(g,g)^{\alpha_k a_1 b}\right)^{s_2}\right) \\
\mathsf{CT}_1 &= \left(g^b\right)^s, \mathsf{CT}_2 = \left(g^{ba_1}\right)^{s_1}, \mathsf{CT}_3 = (g^{a_1})^{s_1}, \mathsf{CT}_4 = \left(g^{ba_2}\right)^{s_2} \\
\mathsf{CT}_5 &= (g^{a_2})^{s_2}, \mathsf{CT}_6 = \tau_1^{s_1} \tau_2^{s_2}, \mathsf{CT}_7 = \left(\tau_1^b\right)^{s_1} \left(\tau_2^b\right)^{s_2} w^{-t} \\
\mathsf{CT}_{j,1} &= g^{t_j} \\
\mathsf{CT}_{j,2} &= \left(w^{\mathsf{ID}_j} h\right)^{t_j},
\end{aligned}$$

for all $j \in [m]$.

$\mathsf{Decrypt}(\mathsf{PK}, \mathsf{sk}_{\vec{x},\mathsf{ID}}, \mathsf{CT})$: In the case $\mathsf{ID} \neq \mathsf{ID}_j$ for all $j \in [m]$, the decryption algorithm will start by computing:

$$\begin{aligned}
A_1 &= e\left(\mathsf{CT}_1, D_1\right) e\left(\mathsf{CT}_2, D_2\right) e\left(\mathsf{CT}_3, D_3\right) e\left(\mathsf{CT}_4, D_4\right) e\left(\mathsf{CT}_5, D_5\right) \\
&= e\left(g^{bs}, g^{\langle \vec{\alpha}, \vec{x} \rangle a_1} v^d\right) e\left(g^{ba_1 s_1}, g^{-\langle \vec{\alpha}, \vec{x} \rangle} v_1^d g^{z_1}\right) e\left(g^{a_1 s_1}, g^{-bz_1}\right) \\
&\qquad e\left(g^{ba_2 s_2}, v_2^d g^{z_2}\right) e\left(g^{a_2 s_2}, g^{-bz_2}\right) \\
&= e(g,g)^{\langle \vec{\alpha}, \vec{x} \rangle a_1 b s_2} e(v,g)^{bsd} e\left(v_1, g\right)^{a_1 b s_1 d} e\left(v_2, g\right)^{a_2 b s_2 d}.
\end{aligned}$$

Next, the algorithm computes:

$$\begin{aligned} A_2 &= e\left(\mathsf{CT}_6, D_6\right) e\left(\mathsf{CT}_7, D_7\right) \\ &= e(v,g)^{bsd} e\left(v_1, g\right)^{a_1 b s_1 d} e\left(v_2, g\right)^{a_2 b s_2 d} e(g,w)^{-d_1 t}. \end{aligned}$$

Now,

$$A_3 = \frac{A_1}{A_2} = e(g,g)^{\langle \vec{\alpha}, \vec{x} \rangle a_1 b s_2} e(g,w)^{d_1 t}.$$

We compute $e(g,w)^{d_1 t}$ as follows:

$$\begin{aligned} A_4 &= \prod_{j=1}^{r} \left(\frac{e\left(\mathsf{CT}_{j,1}, K\right)}{e\left(\mathsf{CT}_{j,2}, D_7\right)} \right)^{\frac{1}{\mathsf{ID}-\mathsf{ID}_j}} \\ &= \prod_{j=1}^{r} \left(e(g,w)^{d_1 t_j (\mathsf{ID}-\mathsf{ID}_j)} \right)^{\frac{1}{\mathsf{ID}-\mathsf{ID}_j}} \\ &= \prod_{j=1}^{r} e(g,w)^{d_1 t_j} = e(g,w)^{d_1 t}. \end{aligned}$$

Finally, we compute

$$\begin{aligned} \frac{\prod_{i=1}^{k} \mathsf{C}_i^{x_i}}{A_3 / A_4} &= \frac{\prod_{i=1}^{k} e(g,g)^{x_i y_i} e(g,g)^{x_i \alpha_i a_1 b s_2}}{e(g,g)^{\langle \vec{\alpha}, \vec{x} \rangle a_1 b s_2}} \\ &= \frac{e(g,g)^{\langle \vec{x}, \vec{y} \rangle} e(g,g)^{\langle \vec{\alpha}, \vec{x} \rangle a_1 b s_2}}{e(g,g)^{\langle \vec{\alpha}, \vec{x} \rangle a_1 b s_2}} \\ &= e(g,g)^{\langle \vec{x}, \vec{y} \rangle}, \end{aligned}$$

and output $\langle \vec{x}, \vec{y} \rangle = \log_{e(g,g)} e(g,g)^{\langle \vec{x}, \vec{y} \rangle}$.

Theorem 6.2

Under the DLIN and BDDH assumption, the above construction is adaptively secure.

Proof. This proof is analogous as in [LSW10] and [AL10]. By using the dual system methodology, we will define (normal, semi-functional) keys and ciphertexts. A normal key and ciphertext are generated by the real scheme and the normal key can decrypt both normal ciphertext and semi-functional ciphertext. A semi-functional key can only decrypt a normal ciphertext and it cannot decrypt a semi-functional ciphertext.
Semi-functional keys: By calling Extract to generate a decryption key for an identity ID. We denote

$$\left(D'_1, D'_2, D'_3, D'_4, D'_5, D'_6, D'_7, K'\right).$$

by the normal key.
Then we set $D_3 = D'_3, D_5 = D'_5, D_6 = D'_6, D_7 = D'_7, K = K'$ (these values are left unchanged). We choose a random $\gamma \in \mathbb{Z}_p$. We set the rest of the key as:

$$D_1 = D'_1 \cdot g^{-a_1 a_2 \gamma}, D_2 = D'_2 \cdot g^{a_2 \gamma}, D_4 = D'_4 \cdot g^{a_1 \gamma}.$$

Semi-Functional Ciphertexts: From a normal ciphertext

$$\mathsf{CT}'_0, \mathsf{CT}'_1, \mathsf{CT}'_2, \mathsf{CT}'_3, \mathsf{CT}'_4, \mathsf{CT}'_5, \mathsf{CT}'_6, \mathsf{CT}'_7, \mathsf{CT}'_{i,1}, \mathsf{CT}'_{i,2}, \forall i \in \mathcal{R}.$$

we pick randomly $\chi \in \mathbb{Z}_q$ and replace $(\mathsf{CT}'_4, \mathsf{CT}'_5, \mathsf{CT}'_6, \mathsf{CT}'_7)$ by

$$\mathsf{CT}_4 = \mathsf{CT}'_4 \cdot g^{ba_2\chi}, \mathsf{CT}_5 = \mathsf{CT}'_5 \cdot g^{a_2\chi}, \mathsf{CT}_6 = \mathsf{CT}'_6 \cdot v_2^{a_2\chi}, \mathsf{CT}_7 = \mathsf{CT}'_7 \cdot v_2^{a_2 b\chi}.$$

We will prove the Theorem by a following sequence of games:
$\mathsf{Game}_{\mathsf{Real}}$ is the real security game. We denote $\mathsf{Game}_{\mathsf{Real}}\mathsf{Adv}_{\mathcal{A}}$ as the advantage of an adversary $\mathcal{A}$ in the game.
Game_0 is identical to $\mathsf{Game}_{\mathsf{Real}}$ but the ciphertext given to the attacker $\mathcal{A}$ is semi-functional.
Game_κ (for $1 \leq \kappa \leq q$) is identical to Game_0 but the first κ decryption key generation queries are answered by returning a semi-functional key. We note that in Game_q the ciphertext and all the keys are semi-functional.
$\mathsf{Game}_{\mathsf{Final}}$: This is the same as Game_q except that the ciphertext is a semi-functional encryption of a random message instead of $\vec{y}_\beta$.

The proof uses the indistinguishability between two consecutive games under some assumptions. The sequence starts from $\mathsf{Game}_{\mathsf{Real}}$ and stops at $\mathsf{Game}_{\mathsf{Final}}$ where the ciphertext is random and the adversary does not have any advantage.

The indistinguishability between $\mathsf{Game}_{\mathsf{Real}}$ and Game_0 is stated and proved as follows:

Lemma 6.1

Under the DLIN assumption, $\mathsf{Game}_{\mathsf{Real}}$ and Game_0 are computationally indistinguishable.

Proof

$\mathcal{B}$ first receives an instance of the DLIN problem

$$(G, g, f, \nu, g^{c_1}, f^{c_2}, T),$$

it needs to distinguish $T = \nu^{c_1+c_2}$ or T is random. When the adversary $\mathcal{A}$ sends $\mathcal{R} = \{\mathsf{ID}_1, \ldots, \mathsf{ID}_r\}$ to $\mathcal{B}$, $\mathcal{B}$ simulates as follows:
$\mathcal{B}$ **simulates** PK: The simulator $\mathcal{B}$ chooses randomly $b, \alpha_1, \ldots, \alpha_k, y_v, y_{v_1}, y_{v_2} \in \mathbb{Z}_q$ and random group elements $w, h \in G$. It then sets

$$g = g, g^{a_1} = f, g^{a_2} = \nu, w = w, h = h.$$

The values a_1, a_2 are unknown to the simulator $\mathcal{B}$. $\mathcal{B}$ sets:

$$g^b, g^{ba_1} = f^b, g^{ba_2} = \nu^b, v = g^{y_v}, v_1 = g^{y_{v_1}}, v_2 = g^{y_{v_2}}.$$

It then can computes

$$\tau_1, \tau_2, \tau_1^b, \tau_2^b, e(g,g)^{\alpha_1 a_1 b} = e(g,f)^{\alpha_1 b}, \ldots, e(g,g)^{\alpha_k a_1 b} = e(g,f)^{\alpha_k b}.$$

We have $\tau_1 = v v_1^{a_1} = v f^{y_{v_1}}$. Finally, $\mathcal{B}$ sends the public parameters to $\mathcal{A}$.

Decryption Key Extraction: $\mathcal{B}$ can generate decryption keys for ID_i for all $\mathsf{ID}_i \in \mathcal{R}$. It can be done because the simulator $\mathcal{B}$ knows

$$\mathsf{MSK} = (g, v, v_1, v_2, \vec{\alpha} = (\alpha_1, \ldots, \alpha_k)) .$$

Simulate Challenge Ciphertext: After obtaining the public key PK and decryption keys for all identities of $\mathcal{R} = \{\mathsf{ID}_1, \ldots, \mathsf{ID}_r\}$, the adversary $\mathcal{A}$ sends $\mathcal{B}$ two messages $\vec{y}_0, \vec{y}_1$. The simulator $\mathcal{B}$ chooses a random value $\beta \in \{0, 1\}$ and will create a semi-functional ciphertext for $\vec{y}_\beta$ and revoked set $\mathcal{R}$ as follows:

1. First, the simulator $\mathcal{B}$ chooses randomly, $s'_1, s'_2, t_1, \ldots, t_r$, and calls the normal encryption algorithm to generate

   $$\mathsf{CT}'_0, \mathsf{CT}'_1, \ldots, \mathsf{CT}'_7, \mathsf{CT}'_{1,1}, \mathsf{CT}'_{1,2}, \ldots, \mathsf{CT}'_{r,1}, \mathsf{CT}'_{r,2},$$

   where

   $$\mathsf{CT}'_0 = (\mathsf{C}'_1, \mathsf{C}'_2, \ldots, \mathsf{C}'_k) .$$

2. $\mathcal{B}$ keeps fix terms $\mathsf{CT}_{i,1} = \mathsf{CT}'_{i,1}, \mathsf{CT}_{i,2} = \mathsf{CT}'_{i,2}$, $i \in [r]$.

It simulates the other components as follows:

$$\mathsf{CT}_0 = (\mathsf{C}_1, \ldots, \mathsf{C}_k); \mathsf{C}_j = \mathsf{C}'_j \left(e\left(g^{c_1}, f\right) e\left(g, f^{c_2}\right)\right)^{b\alpha_j}, j \in [k],$$
$$\mathsf{CT}_1 = \mathsf{CT}'_1 \left(g^{c_1}\right)^b, \mathsf{CT}_2 = \mathsf{CT}'_2 \left(f^{c_2}\right)^{-b},$$
$$\mathsf{CT}_3 = \mathsf{CT}'_3 \left(f^{c_2}\right)^{-1}, \mathsf{CT}_4 = \mathsf{CT}'_4 (T)^b, \mathsf{CT}_5 = \mathsf{CT}'_5 T,$$
$$\mathsf{CT}_6 = \mathsf{CT}'_6 \left(g^{c_1}\right)^{y_v} \left(f^{c_2}\right)^{-y_{v_1}} T^{y_{v_2}}, \mathsf{CT}_7 = \mathsf{CT}'_7 \left(\left(g^{c_1}\right)^{y_v} \left(f^{c_2}\right)^{-y_{v_1}} T^{y_{v_2}}\right)^b .$$

We consider 2 cases:

1. If $T = \nu^{c_1+c_2}$, this will be a normal ciphertext with $s_1 = -c_2 + s'_1, s_2 = c_1 + c_2 + s'_2$, and $s = s_1 + s_2 = c_1 + s'_1 + s'_2$.

2. If T is random, this will be a properly distributed semi-functional ciphertext.

Therefore, whenever $\mathcal{A}$ has some advantage in distinguishing $\mathsf{Game}_{\mathsf{Real}}$ from Game_0, the simulator can distinguish between the random tuple and DLIN tuple with the same advantage. ■

Lemma 6.2

Under the DLIN assumption, Game_κ and $\mathsf{Game}_{\kappa-1}$ are computationally indistinguishable for each $\kappa \in [r]$.

Proof

We assume that there exists an adversary $\mathcal{A}$ can distinguish two game Game_κ and the $\mathsf{Game}_{\kappa-1}$. We will build an algorithm $\mathcal{B}$ to break DLIN assumption.

That is $\mathcal{B}$ first receives an instance of the DLIN problem:

$$(G, g, f, \nu, g^{c_1}, f^{c_2}, T)\,,$$

it needs to distinguish $T = \nu^{c_1+c_2}$ or T is random.

To do this, $\mathcal{B}$ will call on $\mathcal{A}$ by simulating either Game_κ or $\mathsf{Game}_{\kappa-1}$.

$\mathcal{B}$ simulates PK: $\mathcal{B}$ picks randomly $\alpha_1, \ldots, \alpha_k, a_1, a_2, y_{v_1}, y_{v_2}, y_w, y_h \in \mathbb{Z}_q$. It then sets

$$\begin{gathered}
g^b = f, g^{a_1}, g^{a_2}, g^{ba_1} = f^{a_1}, g^{ba_2} = f^{a_2}, v = \nu^{-a_1 a_2}, \\
v_1 = \nu^{a_2} g^{y_{v_1}}, v_2 = \nu^{a_1} g^{y_{v_2}}, \\
e(g,g)^{\alpha_1 a_1 b} = e(f,g)^{\alpha_1 a_1}, \ldots, e(g,g)^{\alpha_k a_1 b} = e(f,g)^{\alpha_k a_1}, \\
\tau_1 = v v_1^{a_1}, \tau_2 = v v_2^{a_2}, \tau_1^b = f^{y_{v_1} a_1}, \tau_2^b = f^{y_{v_2} a_2}, \\
w = f g^{y_w}, h = w^{-\mathsf{ID}_\kappa} g^{y_h}.
\end{gathered}$$

Decryption Key Extraction: The simulator $\mathcal{B}$ generates decryption keys in three cases:

- In case $j > \kappa$, $\mathcal{B}$ generates a normal key for ID_j by calling the usual decryption key generation algorithm because it knows the MSK.
- In case $j < \kappa$, $\mathcal{B}$ generates a semi-functional key for ID_j because it can run the semi-functional key generation algorithm described above and it knows the exponents a_1 and a_2.
- For $j = \kappa$, the simulator will create a decryption key that is normal if $T = \nu^{c_1+c_2}$ and is semi-functional if T is random.

We need to generate the functional key for ID_κ, $\mathcal{B}$ calls the key generation algorithm to extract a normal key $\mathsf{sk}_{\vec{x},\mathsf{ID}} : D'_1, D'_2, \ldots, D'_7, K'$. We let d'_1, d'_2, z'_1, z'_2 denote the random exponents that were chosen. We then set:

$$\begin{gathered}
D_1 = D'_1 T^{-a_1 a_2}, D_2 = D'_2 T^{a_2} \left(g^{c_1}\right)^{y_{v_1}}, D_3 = D'_3 \left(f^{c_2}\right)^{y_{v_1}}, \\
D_4 = D'_4 T^{a_1} \left(g^{c_1}\right)^{y_{v_2}}, D_5 = D'_5 \left(f^{c_2}\right)^{y_{v_2}}, D_6 = D'_6 f^{c_2}, \\
D_7 = D'_7 \left(g^{c_1}\right), K = K' \left(g^{c_1}\right)^{y_h}.
\end{gathered}$$

We consider 2 cases:

1. If $T = \nu^{c_1+c_2}$, then this is a normal key with $d_1 = d'_1 + c_1$ and $d_2 = d'_2 + c_2$.
2. If T is random, we can write T as $T = \nu^{c_1+c_2} g^\gamma$. This is a semi-functional key.

Simulate Challenge Ciphertext: After obtaining the public key PK and decryption keys for all identities of $\mathcal{R} = \{\mathsf{ID}_1, \ldots, \mathsf{ID}_r\}$, the adversary $\mathcal{A}$ sends $\mathcal{B}$ two messages $\vec{y}_0, \vec{y}_1$. The simulator $\mathcal{B}$ chooses a random value $\beta \in \{0, 1\}$ and generates a semi-functional ciphertext for $\vec{y}_\beta$ and revoked set $\mathcal{R}$ as follows:

1. First, $\mathcal{B}$ uses the normal encryption algorithm with randomly chosen exponents s'_1, s'_2, t' to create $\mathsf{CT}'_0, \mathsf{CT}'_1, \ldots, \mathsf{CT}'_7$.
2. We keep fix $\mathsf{CT}_0 = \mathsf{CT}'_0, \mathsf{CT}_1 = \mathsf{CT}'_1, \mathsf{CT}_2 = \mathsf{CT}'_2, C_3 = C'_3$.

3. To add semi-functionality, $\mathcal{B}$ chooses a random exponent $x \in \mathbb{Z}_p$ and sets:

$$\mathsf{CT}_4 = \mathsf{CT}'_4 f^{a_2 x}, \mathsf{CT}_5 = \mathsf{CT}'_5 g^{a_2 x}, \mathsf{CT}_6 = \mathsf{CT}'_6 v_2^{a_2 x}, \mathsf{CT}_7 = \mathsf{CT}'_7 f^{a_2 y_{v_2} x_\nu - a_1 x y_w a_2}.$$

4. For CT_7, we have implicitly set $g^t = g^{t'} \nu^{a_1 x a_2}$. We let y_ν denote the unknown discrete log of ν in base g. Then, we have set $t = t' + y_\nu a_1 a_2 x$, so t is not known to $\mathcal{B}$, but t' is.

5. For $i \neq \kappa, 1 \leq i \leq r, \mathcal{B}$ sets t_i to be a randomly chosen value. We let t'' denote the sum of these values. Then t_κ is defined to be $t' - t'' + y_\nu a_1 a_2 x$. For $i \neq \kappa$, the simulator $\mathcal{B}$ knows the value of t_i, and so can compute:

$$\mathsf{CT}_{i,1} = g^{t_i}, \mathsf{CT}_{i,2} = \left(w^{\mathsf{ID}_i} h\right)^{t_i}.$$

For $i = \kappa, \mathcal{B}$ computes:

$$\mathsf{CT}_{\kappa,1} = g^{t_\kappa} = g^{t'-t''} \nu^{a_1 a_2 x},$$
$$\mathsf{CT}_{\kappa,2} = \left(w^{\mathsf{ID}_\kappa} w^{-\mathsf{ID}_\kappa} g^{y_h}\right)^{y_\nu a_1 a_2 x + t' - t''} = \nu^{y_h a_1 a_2 x} g^{y_h (t'-t'')}.$$

We consider 2 cases:

1. If $T = \nu^{c_1+c_2}$, and $\mathcal{B}$ has guessed correctly, $\mathcal{B}$ has properly simulated $\mathsf{Game}_{\kappa-1}$.

2. When T is random and $\mathcal{B}$ has guessed correctly, $\mathcal{B}$ has properly simulated Game_κ.

■

Lemma 6.3

Under the BDDH assumption, Game_r and $\mathsf{Game}_{\mathsf{Final}}$ are computationally indistinguishable.

Proof

The simulator $\mathcal{B}$ first receives an instance of the BDDH problem: $(g, g^{c_1}, g^{c_2}, g^{c_3}, T)$. $\mathcal{B}$ must decide whether $T = e(g,g)^{c_1 c_2 c_3}$ or is random. To accomplish this, $\mathcal{B}$ will call on $\mathcal{A}$ by simulating either Game_r or $\mathsf{Game}_{\mathsf{Final}}$. $\mathcal{A}$ first sends a set $S = \{\mathsf{ID}_1, \ldots, \mathsf{ID}_r\}$ to $\mathcal{B}$.
$\mathcal{B}$ simulates PK: $\mathcal{B}$ chooses random exponents $b, \alpha_1, \ldots, \alpha_k, y_v, y_{v_1}, y_{v_2}, y_w, y_h \in \mathbb{Z}_q$. It sets:

$$g = g, g^b, g^{a_1} = g^{c_1}, g^{a_2} = g^{c_2}, g^{ba_1}, g^{ba_2} = \left(g^{c_2}\right)^b, v = g^{y_v}, v_1 = g^{y_{v_1}}$$
$$v_2 = g^{y_{v_2}}, w = g^{y_w}, h = g^{y_h},$$
$$e(g,g)^{a_1 \alpha_1 b} = e\left(g^{c_1}, g^{c_2}\right)^{\alpha_1}, \ldots, e(g,g)^{a_1 \alpha_k b} = e\left(g^{c_1}, g^{c_2}\right)^{\alpha_k}.$$

Decryption Key Extraction: $\mathcal{B}$ must now generate semi-functional keys for

$\mathsf{ID}_1, \ldots, \mathsf{ID}_r$. For each ID_i, $\mathcal{B}$ chooses random exponents $d_1, d_2, z_1, z_2, \gamma' \in \mathbb{Z}_q$ and sets $d = d_1 + d_2$. The key elements are computed as:

$$D_1 = (g^{c_2})^{-\gamma' a_1} v^d, D_2 = (g^{c_2})^{\gamma'} v_1^d g^{z_1}, D_3 = \left(g^b\right)^{-z_1}, D_4 = (g^{c_1})^{a_1} g^{a_1 \gamma'} v_2^d g^{z_2}$$
$$D_5 = g^{-bz_2}, D_6 = g^{d_2 b}, D_7 = g^{d_1}, K = \left(w^{\mathsf{ID}_i} h\right)^{d_1}.$$

Simulate Challenge Ciphertext: After obtaining the public key PK and decryption keys for all identities of $\mathcal{R} = \{\mathsf{ID}_1, \ldots, \mathsf{ID}_r\}$, the adversary $\mathcal{A}$ sends $\mathcal{B}$ two messages $\vec{y}_0, \vec{y}_1$. The simulator $\mathcal{B}$ chooses a random value $\beta \in \{0, 1\}$ and will create either a semi-functional ciphertext for $\vec{y}_\beta$ or a semi-functional encryption of a random message as follows:

$\mathcal{B}$ chooses random exponents $s_1, x', t_1, \ldots, t_r$ and sets $t = t_1 + \cdots + t_r$. It forms the ciphertext as:

$$\mathsf{CT}_0 = (\mathsf{C}_1, \ldots, \mathsf{C}_k) = \left(e(g,g)^{y_{\beta,1} T^b}, \ldots, e(g,g)^{y_{\beta,k} T^b}\right)$$
$$\mathsf{CT}_1 = g^{s_1 b} \left(g^{c_3}\right)^b, \mathsf{CT}_2 = g^{b a_1 s_1}, \mathsf{CT}_3 = g^{a_1 s_1}, \mathsf{CT}_4 = (g^{c_2})^{x' b},$$
$$\mathsf{CT}_5 = (g^{c_2})^{x'}, \mathsf{CT}_6 = \tau_1^{s_1} (g^{c_3})^{y_v} (g^{c_2})^{y_{v_2} x'}, \mathsf{CT}_7 = \left(\tau_1^b\right)^{s_1} (g^{c_3})^{y_v b} (g^{c_2})^{y_{v_2} x' b} w^{-t},$$
$$\mathsf{CT}_{1,1} = g^{t_1}, \mathsf{CT}_{1,2} = \left(w^{\mathsf{ID}_1} h\right)^{t_1}, \ldots, \mathsf{CT}_{r,1} = g^{t_r}, \mathsf{CT}_{r,2} = \left(w^{\mathsf{ID}_r} h\right)^{t_r}.$$

These assignments implicitly set $s_2 = c_3$ and $x = -c_3 + x'$.

1. If $T = e(g,g)^{c_1 c_2 c_3}$, then this is a properly distributed semi-functional encryption of M_β.

2. If T is random, then this is a properly distributed semi-functional encryption of a random message.

Therefore $\mathcal{B}$ can use $\mathcal{A}$'s output to distinguish $T = e(g,g)^{c_1 c_2 c_3}$ from random with the same advantage that $\mathcal{A}$ has in distinguishing Game_r from $\mathsf{Game}_{\mathsf{Final}}$. ■

6.3 Revocable Functional Encryption for Inner Product with Constant-size Ciphertext

In this section we present two constructions of revocable inner product functional encryption. These constructions achieve constant-size ciphertext but the size of private keys is longer than constructions in the previous section as we applied the n-equation technique proposed by Attrapadung et al. [AL10]. The first construction is based on MEBDH assumption and it achieves selective security. The second construction is based on simple assumptions such as BDDH and DLIN and it achieves adaptive security as we use Waters 's dual system method [Wat09].

6.3.1 Construction based on q-type Assumption

We will construct a revocable functional encryption for inner product with constant-size ciphertext.

Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ a bilinear map, where $\mathbb{G}, \mathbb{G}_T$ are cyclic groups of order q, written multiplicatively.

$\mathsf{Setup}(1^\lambda, 1^k)$: This algorithm generates a bilinear pairing $(\mathbb{G}, \mathbb{G}_T, q, e)$ for sufficiently large prime order q and g is a generator of the group $\mathbb{G}$. The bilinear map e over $(\mathbb{G}, \mathbb{G})$ can be calculated efficiently.

- For each $i \in \{1, \ldots, k\}$, randomly choose $s_i \xleftarrow{\$} \mathbb{Z}_q$. We set $\vec{s} = (s_1, \ldots, s_k)$.
- We take randomly $\alpha, \alpha_1, \alpha_2, \ldots, \alpha_k \xleftarrow{\$} \mathbb{Z}_q$ and set $\vec{\alpha} = (\alpha_1, \ldots, \alpha_k)$.
- The public key is

$$\mathsf{PK} = \Big(\mathbb{G}, g, g^{\alpha_1}, \ldots, g^{\alpha_k}, g^{\alpha_1^2}, g^{\alpha_1\alpha_2}, \ldots, g^{\alpha_1\alpha_k}, e(g,g)^{\alpha s_1}, \ldots, e(g,g)^{\alpha s_k}\Big).$$

- The master secret key is

$$\mathsf{MSK} = \{\alpha, \alpha_1, \ldots, \alpha_k, s_1, \ldots, s_k\}.$$

$\mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, \vec{x})$: Takes as input an identity ID (an integer in the set $[n]$), the master secret key MSK and a characteristic vector $\vec{x} = (x_1, \ldots, x_k) \in \mathbb{Z}_q^k$. Each identity ID corresponds to a vector

$$\vec{a} = (\mathsf{ID}^0, \mathsf{ID}^1, \ldots, \mathsf{ID}^{k-1}) = (a_1, \ldots, a_k).$$

We assume that the first component of $\vec{a}$ is $a_1 \neq 0 \mod q$. For each vector $\vec{a}$, we take randomly $t \xleftarrow{\$} \mathbb{Z}_q$ and the secret key $\mathsf{sk}_{\vec{x},\vec{a}}$ is a $(k+1)$-tuple:

$$\mathsf{sk}_0 = g^t, \mathsf{sk}_1 = g^{\langle \vec{s}, \vec{x} \rangle \alpha + \alpha_1^2 t},$$

$$\mathsf{sk}_2 = g^{t\left(-\alpha_1 \frac{a_2}{a_1} + \alpha_2\right)}, \mathsf{sk}_3 = g^{t\left(-\alpha_1 \frac{a_3}{a_1} + \alpha_3\right)}, \ldots, \mathsf{sk}_k = g^{t\left(-\alpha_1 \frac{a_k}{a_1} + \alpha_k\right)}.$$

In a matrix form, we have

$$\mathsf{sk}_0 = g^t, \mathsf{sk}_1 = g^{\langle \vec{s}, \vec{x} \rangle \alpha + \alpha_1^2 t}, \mathsf{sk}_2 = g^{t M_{\mathsf{ID}}^\top \vec{\alpha}},$$

where

$$M_{\mathsf{ID}}^T := \begin{pmatrix} -\frac{a_2}{a_1} & 1 & 0 & \ldots & 0 \\ -\frac{a_3}{a_1} & 0 & 1 & \ldots & 0 \\ \vdots & & & \ddots & \\ -\frac{a_k}{a_1} & 0 & 0 & \ldots & 1 \end{pmatrix} \in \mathbb{Z}_q^{(k-1)\times k}$$

$\mathsf{Encrypt}(\mathcal{R}, \mathsf{PK}, \vec{y})$: Takes as input the set $\mathcal{R}$ of identities that secret keys will be revoked, the public key PK, a message $\vec{y} = (y_1, \ldots, y_k) \in \mathbb{Z}_q^k$.

We choose $b_1, \ldots, b_k$ are coefficients of polynomial $f(x)$ with degree $k-1$ that is defined as follows:

$$f(x) = z(x) \prod_{i \in \mathcal{R}} (x - i), \text{ where } z(i) \neq 0, \forall i \in \mathcal{R}.$$

We sample $r \xleftarrow{\$} \mathbb{Z}_q$ and compute

$$\begin{aligned} \mathsf{CT}_0 &= (\mathsf{C}_1, \mathsf{C}_2, \ldots, \mathsf{C}_k) \\ &= (e(g,g)^{y_1} e(g,g)^{s_1 \alpha r}, \ldots, e(g,g)^{y_k} e(g,g)^{s_k \alpha r}) \\ \mathsf{CT}_1 &= g^{r\alpha_1 \langle \vec{b}, \vec{\alpha} \rangle} \\ \mathsf{CT}_2 &= g^r \\ \mathsf{CT}_3 &= g^{r\alpha_1}. \end{aligned}$$

$\mathsf{Decrypt}(\mathsf{PK}, \mathsf{sk}_{\vec{x},\mathsf{ID}}, \mathsf{CT})$: We set

$$M_1^T = \left(-\frac{a_2}{a_1}, \ -\frac{a_3}{a_1}, \cdots, -\frac{a_k}{a_1}\right),$$
$$\vec{b}_1^T = (b_2, \ b_3, \ldots, b_k).$$

We have

$$\begin{aligned} A_1 := \left(\frac{e\left(\mathsf{CT}_1, \mathsf{sk}_0\right)}{e\left(\mathsf{sk}_2^{\vec{b}_1}, \mathsf{CT}_3\right)}\right)^{\frac{1}{M_1\vec{b}_1 - b_1}} &= \left(\frac{e\left(g^{r\alpha_1\langle\vec{b},\vec{\alpha}\rangle}, g^t\right)}{e\left(g^{t\vec{b}_1^T M_{\mathsf{ID}}^T\vec{\alpha}}, g^{\alpha_1 r}\right)}\right)^{\frac{1}{M_1\vec{b}_1 - b_1}} \\ &= \left(\frac{e(g,g)^{rt\alpha_1\langle\vec{b},\vec{\alpha}\rangle}}{e(g,g)^{rt\alpha_1\vec{b}_1^T M_{\mathsf{ID}}^T\vec{\alpha}}}\right)^{\frac{1}{M_1\vec{b}_1 - b_1}} \\ &= \left(\frac{1}{e(g,g)^{rt\alpha_1\langle M_{\mathsf{ID}}\vec{b}_1 - \vec{b},\vec{\alpha}\rangle}}\right)^{\frac{1}{M_1\vec{b}_1 - b_1}} \\ &= e(g,g)^{-rt\alpha_1^2}. \end{aligned}$$

The last equation is true because

$$\langle M_{\mathsf{ID}}\vec{b}_1 - \vec{b}, \vec{\alpha}\rangle = \left(M_1\vec{b}_1 - b_1\right)\alpha_1.$$

We then computes

$$A_2 := e(\mathsf{sk}_1, \mathsf{CT}_2) = e(g^{\langle\vec{s},\vec{x}\rangle\alpha + \alpha_1^2 t}, g^r) = e(g,g)^{r\alpha\langle\vec{s},\vec{x}\rangle} e(g,g)^{\alpha_1^2 tr}.$$

Finally, we compute

$$\begin{aligned} \frac{\prod_{i=1}^k \mathsf{C}_i^{x_i}}{A_1 \cdot A_2} &= \frac{\prod_{i=1}^k e(g,g)^{x_i y_i} e(g,g)^{r\alpha x_i s_i}}{e(g,g)^{r\alpha\langle\vec{s},\vec{x}\rangle}} \\ &= \frac{e(g,g)^{\langle\vec{x},\vec{y}\rangle} e(g,g)^{\langle\vec{s},\vec{x}\rangle r\alpha}}{e(g,g)^{r\alpha\langle\vec{s},\vec{x}\rangle}} \\ &= e(g,g)^{\langle\vec{x},\vec{y}\rangle}, \end{aligned}$$

and output $\langle\vec{x},\vec{y}\rangle = \log_{e(g,g)} e(g,g)^{\langle\vec{x},\vec{y}\rangle}$.

Theorem 6.3

Under the q-MEBDH assumption, the construction is selectively secure.

Proof

Suppose that there exists an adversary $\mathcal{A}$ who breaks the selective security game of our scheme. We will build a simulator $\mathcal{B}$ that breaks the decisional q-MEBDH assumption.

The simulator $\mathcal{B}$ takes as input a q-MEBDH challenge P, T. The simulator then proceeds in the game as follows.

The adversary $\mathcal{A}$ submits a revocation set $\mathcal{R} = \{\mathsf{ID}_1, \ldots, \mathsf{ID}_m\}$, $(m \leq \mathbf{q})$ and two messages $\vec{y}_0, \vec{y}_1 \in \mathbb{Z}_q^k$ to the simulator.

$\mathcal{B}$ finds a basis $\{\mathbf{z}_i\}_{i\in[k-1]}$ of $(\vec{y}_1 - \vec{y}_0)^\perp$. For each vector $\mathbf{z}_i$, chooses a random element $d_i \in \mathbb{Z}_p$ as secret key and set $e(g,g)^{d_i}$ as the master public key for $\mathbf{z}_i$

and also, $\mathcal{B}$ sets $e(g,g)^{\alpha}$ as the master public key for $\vec{y}_1 - \vec{y}_0$. Let $\{\vec{e}_1, \ldots, \vec{e}_k\}$ be the canonical basis of $\mathbb{Z}_q^k$. $\mathcal{B}$ finds $c_i, \lambda_{i,j} \in \mathbb{Z}_q$ such that

$$\vec{e}_i = c_i \left(\vec{y}_1 - \vec{y}_0\right) + \sum_{j=1}^{k-1} \lambda_{i,j} \mathbf{z}_j$$

then, sets

$$\gamma_i := \prod_{j=1}^{k-1} e(g,g)^{d_j \lambda_{i,j}} \quad \text{and} \quad e(g,g)^{\alpha s_i} := e(g,g)^{\alpha c_i} \cdot \gamma_i.$$

Let $\vec{a} = (a_1, \ldots, a_{\mathbf{q}})^T$.

For each $j = 1, \ldots, \mathbf{q}$, we denote $M_{\mathsf{ID}_j}^{(-1)}$ as the matrix obtained by omitting the first row. We have

$$\mathsf{rank}\left(M_{\mathsf{ID}_j}^{(-1)}\right) = k - 1.$$

Therefore the linear equation system with unknowns $(b_{2,j}, \ldots, b_{k,j})$:

$$\left(M_{\mathsf{ID}_j}\right)^T \vec{b}_j := \left(M_{\mathsf{ID}_j}\right)^T (1, b_{2,j}, \cdots, b_{k,j})^T = \overrightarrow{0}$$

have a solution. We define a $k \times \mathbf{q}$ matrix

$$B = \left[\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_{\mathbf{q}}\right],$$

where $\vec{b}_j$ is the j^{th} column of B.

We choose $\vec{\delta} = (\delta_1, \ldots, \delta_{\mathbf{q}})^T \xleftarrow{\$} \mathbb{Z}_q^{\mathbf{q}}$ and implicitly define the vector

$$\vec{\alpha} = B\vec{a} + B\vec{\delta}.$$

We set

$$g^{\alpha_1} = g^{a_1 + \cdots + a_{\mathbf{q}} + \delta_1 + \cdots + \delta_{\mathbf{q}}}.$$

Therefore, we can calculate

$$g^{\alpha_1 \vec{\alpha}} = g^{\alpha_1 B\vec{a}} g^{\alpha_1 B\vec{\delta}} = g^{B(a_1 + \cdots + a_{\mathbf{q}} + \delta_1 + \cdots + \delta_{\mathbf{q}})\vec{a}} g^{\alpha_1 B\vec{\delta}}$$

from terms g^{a_i}, $g^{a_i a_j}$ of MEBDH instance problem. Therefore, we totally simulate the public key.

Simulate Decryption keys: The simulator $\mathcal{B}$ should answer all decryption keys which are asked by the adversary $\mathcal{A}$ such that the function $\vec{x}$ must hold

$$\langle \vec{x}, \vec{y}_1 - \vec{y}_0 \rangle = 0.$$

- We have

$$\langle \vec{s}, \vec{x} \rangle = \sum_{j \in [k-1]} \left(\sum_{i \in [k]} x_i \lambda_{i,j} \right) d_j.$$

- $\mathcal{B}$ simulates the decryption key for j^{th} query as follows. It chooses $t'_j \xleftarrow{\$} \mathbb{Z}_q$ and implicitly defines $t_j = t'_j - \alpha\langle\vec{s},\vec{x}\rangle/a_j^2$ by setting

$$\begin{aligned}
\mathsf{sk}_0 &= g^{t'_j} g^{-\alpha\langle\vec{s},\vec{x}\rangle/a_j^2}, \\
\mathsf{sk}_1 &= g^{\alpha\langle\vec{s},\vec{x}\rangle + t_j\alpha_1^2} = g^{\alpha\langle\vec{s},\vec{x}\rangle + \left(t'_j - \alpha\langle\vec{s},\vec{x}\rangle/a_j^2\right)\alpha_1^2} \\
&= \left(g^{\alpha_1^2}\right)^{t'_j} \cdot g^{\alpha\langle\vec{s},\vec{x}\rangle\left(1-\alpha_1^2\right)/a_j^2}.
\end{aligned}$$

The components $\mathsf{sk}_0, \mathsf{sk}_1$ can be computed from terms $g^{\alpha/a_j^2}, g^{\alpha a_i^2/a_j^2}$ for $1 \le i, j \le k$ from the problem instance.

We have

$$\begin{aligned}
\mathsf{sk}_2 &= g^{t_j\left(M_{\mathsf{ID}_j}\right)^T\vec{\alpha}} \\
&= g^{t_k\left(M_{\mathsf{ID}_j}\right)^T(B\vec{a}+B\vec{\delta})} \\
&= g^{t'_k\left(M_{\mathsf{ID}_j}\right)^\top B\vec{a}} \cdot g^{-\frac{\alpha\langle\vec{s},\vec{x}\rangle\left(M_{\mathsf{ID}_j}\right)^T B\vec{a}}{a_j^2}} \cdot g^{t_j\left(M_{\mathsf{ID}_j}\right)^T B\vec{\delta}}.
\end{aligned}$$

The term $g^{t'_j\left(M_{\mathsf{ID}_j}\right)^\top B\vec{a}}$ can be calculated trivially from g^{a_i} and the term $g^{t_j\left(M_{\mathsf{ID}_j}\right)^T B\vec{\delta}}$ can be calculated from g^{α/a_j^2}.

We consider the term $g^{-\frac{\alpha\langle\vec{s},\vec{x}\rangle\left(M_{\mathsf{ID}_j}\right)^T B\vec{a}}{a_j^2}}$. Since

$$\left(M_{\mathsf{ID}_j}\right)^T\vec{b}_j = \overrightarrow{0},$$

the term a_j does not appear in $\left(M_{\mathsf{ID}_j}\right)^T B\vec{a}$.

Therefore, the term $g^{-\alpha\left(M_{\mathsf{ID}_j}\right)^T B\vec{a}/a_j^2}$ can be computed from the term $g^{\alpha a_i/a_j^2}$ for $j \ne i$ from the problem instance.

Simulate Challenge Ciphertext: The simulator $\mathcal{B}$ receives $\vec{y}_0, \vec{y}_1$ and it picks randomly $\beta \in \{0,1\}$. The simulator then chooses random $r' \in \mathbb{Z}_q$. The adversary $\mathcal{A}$ chooses a set of identity $\mathcal{R}^\star \subset \mathcal{R}$ and sends to $\mathcal{B}$. We call $\vec{b}^\star$ a represention of the set $\mathcal{R}^\star$. Hence, for all $j \in [1, \mathbf{q}]$, there must exist $\vec{w}_j \in \mathbb{Z}_p^{k-1}$ such that $\vec{b}^\star = M_{\mathsf{ID}_j}\vec{w}_j$. The ciphertext will be encrypted under randomness $\widetilde{r} = r + r'$. The challenge ciphertext will be simulated as follows:

$$\begin{aligned}
\mathsf{CT}_0 &= (\mathsf{C}_1, \mathsf{C}_2, \ldots, \mathsf{C}_k) \\
&= \left(e(g,g)^{y_{\beta,1}} T^{s_1} \cdot e(g,g)^{s_1\alpha r'}, \ldots, e(g,g)^{y_{\beta,k}} T^{s_k} \cdot e(g,g)^{s_k\alpha r'}\right) \\
\mathsf{CT}_1 &= g^{r\alpha_1\left\langle\vec{b}^\star,\vec{\alpha}\right\rangle} g^{r'\alpha_1\left\langle\vec{b}^\star,\vec{\alpha}\right\rangle} \\
&= g^{r\alpha_1\left(\vec{b}^\star\right)^T B\vec{a}} g^{r\alpha_1\left(\vec{b}^\star\right)^T B\vec{\delta}} g^{r'\alpha_1\left\langle\vec{b}^\star,\vec{\alpha}\right\rangle} \\
\mathsf{CT}_2 &= g^r g^{r'} \\
\mathsf{CT}_3 &= g^{ra_1} \cdots g^{ra_{\mathbf{q}}} \cdot \left(g^r\right)^{\delta_1+\cdots+\delta_{\mathbf{q}}} \cdot \left(g^{\alpha_1}\right)^{r'}.
\end{aligned}$$

We consider the term $g^{r\alpha_1\left(\vec{b}^\star\right)^T B\vec{a}}$ in the ciphertext CT_1. We have

$$g^{r\alpha_1\left(\vec{b}^\star\right)^T B\vec{a}} = 1.$$

Indeed, for each $j \in [1, q]$, the coefficient of a_j in $\left(\vec{b}^\star\right)^T B\vec{a}$ is 0 since

$$\left(\vec{b}^\star\right)^T \vec{b}_j = \left(M_{\mathsf{ID}_j}\vec{w}_j\right)^T \vec{b}_j = \left(\vec{w}_j\right)^T \left(M_{\mathsf{ID}_j}\right)^T \vec{b}_j = \left(\vec{w}_j\right)^T \overrightarrow{0} = 0.$$

Therefore, there does not have $g^{ra_i^2}$ in the term $g^{r\alpha_1\left(\vec{b}^\star\right)^T B\vec{a}}$.

The other components CT_0, CT_2, CT_3 of the ciphertext can be easily calculated from the problem instance.

Finally, the adversary $\mathcal{A}$ outputs a guess β' for β. The simulator then outputs 1 to guesses that $T = e(g,g)^{\alpha r}$ if $\beta = \beta'$; otherwise, it and outputs 0 to indicate that it believes T is a random group element in $\mathbb{G}_T$.

When $T = e(g,g)^{\alpha r}$ the simulator $\mathcal{B}$ gives a perfect simulation so we have that

$$\Pr\left[\mathcal{B}\left(P, T = e(g,g)^{\alpha r}\right) = 0\right] = \frac{1}{2} + \mathsf{Adv}_{\mathcal{A}}.$$

When $T = R$ is a random group element the message $\vec{y}_\beta$ is completely hidden from the adversary and we have

$$\Pr[\mathcal{B}(P, T = R) = 0] = \frac{1}{2}.$$

Therefore, $\mathcal{B}$ can solve the decisional q-MEBDH game with non-negligible advantage. ■

6.3.2 Construction based on BDDH and DLIN Assumptions

Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ a bilinear map, where $\mathbb{G}, \mathbb{G}_T$ are cyclic groups of order q, written multiplicatively.

$\mathsf{Setup}(1^\lambda, 1^k)$**:** This algorithm generates a bilinear pairing $(\mathbb{G}, \mathbb{G}_T, q, e)$ for sufficiently large prime order q and g is a generator of the group $\mathbb{G}$. The bilinear map e over $(\mathbb{G}, \mathbb{G})$ can be calculated efficiently.

- We choose random generators $g, v, v_1, v_2 \in \mathbb{G}$.
- We pick random exponents $a_1, a_2, b, \alpha_1, \ldots, \alpha_k \in \mathbb{Z}_p$.
- We set $\tau_1 = vv_1^{a_1}, \tau_2 = vv_2^{a_2}, w = g^{\alpha_1}$.
- The public key is

$$\mathsf{PK} = \left(\begin{array}{c} g, w, g^b, g^{a_1}, g^{a_2}, g^{\alpha_1}, \ldots, g^{\alpha_k}, g^{ba_1}, g^{ba_2}, \tau_1, \tau_2, \tau_1^b, \tau_2^b, \\ e(g,g)^{\alpha_1 a_1 b}, \ldots, e(g,g)^{\alpha_k a_1 b} \end{array} \right)$$

- The master secret key is

$$\mathsf{MSK} = \left(v, v_1, v_2, \vec{\alpha} = (\alpha_1, \ldots, \alpha_k)\right).$$

$\mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, \vec{x})$: Takes as input an identity $\mathsf{ID} \in \mathbb{Z}_q$, the master secret key MSK and a characteristic vector $\vec{x} = (x_1, \ldots, x_k) \in \mathbb{Z}_q^k$. Each identity ID corresponds to a vector

$$\vec{\beta} = (\beta_1, \ldots, \beta_k) \in \mathbb{Z}_q^k.$$

We assume that the first component of $\vec{\beta}$ is $\beta_1 \neq 0 \mod q$.

The algorithm chooses random exponents $d_1, d_2, z_1, z_2 \in \mathbb{Z}_p$ and sets $d = d_1 + d_2$. The secret key $\mathsf{sk}_{\vec{x},\mathsf{ID}}$ is the following tuple:

$$D_1 = g^{\langle \vec{\alpha}, \vec{x} \rangle a_1} v^d, D_2 = g^{-\langle \vec{\alpha}, \vec{x} \rangle} v_1^d g^{z_1}, D_3 = \left(g^b\right)^{-z_1}, D_4 = v_2^d g^{z_2},$$
$$D_5 = \left(g^b\right)^{-z_2}, D_6 = g^{d_2 b}, D_7 = g^{d_1}.$$

$$K_2 = g^{d_1\left(-\alpha_1 \frac{\beta_2}{\beta_1} + \alpha_2\right)}, K_3 = g^{d_1\left(-\alpha_1 \frac{\beta_3}{\beta_1} + \alpha_3\right)}, \ldots, K_k = g^{d_1\left(-\alpha_1 \frac{\beta_k}{\beta_1} + \alpha_k\right)}.$$

The term K_i can be rewritten in matrix form as follows:

$$K = g^{d_1 M_{\mathsf{ID}}^T \vec{\alpha}},$$

where

$$M_{\mathsf{ID}}^T := \begin{pmatrix} -\frac{\beta_2}{\beta_1} & 1 & 0 & \ldots & 0 \\ -\frac{\beta_3}{\beta_1} & 0 & 1 & \ldots & 0 \\ \vdots & & & \ddots & \\ -\frac{\beta_k}{\beta_1} & 0 & 0 & \ldots & 1 \end{pmatrix}.$$

$\mathsf{Encrypt}(\mathcal{R}, \mathsf{PK}, \vec{y})$: Takes as input the set $\mathcal{R}$ of identities $\{\mathsf{ID}_1, \ldots, \mathsf{ID}_r\}$ that secret keys will be revoked, the public key PK, a message $\vec{y} = (y_1, \ldots, y_k) \in \mathbb{Z}_q^k$. We choose $b_1, \ldots, b_k$ are coefficients of polynomial $f(x)$ with degree $k-1$ defined as follows:

$$f(x) = z(x) \prod_{i \in \mathcal{R}} (x - i), \text{ where } z(i) \neq 0, \forall i \in \mathcal{R}.$$

The encryption algorithm chooses random exponents s_1, s_2, t and sets $s = s_1 + s_2$. The ciphertext CT is constructed as:

$$\begin{aligned}
\mathsf{CT}_0 &= (\mathsf{C}_1, \mathsf{C}_2, \ldots, \mathsf{C}_k) \\
&= \left(e(g,g)^{y_1} \left(e(g,g)^{\alpha_1 a_1 b}\right)^{s_2}, \ldots, e(g,g)^{y_k} \left(e(g,g)^{\alpha_k a_1 b}\right)^{s_2}\right) \\
\mathsf{CT}_1 &= \left(g^b\right)^s, \mathsf{CT}_2 = \left(g^{b a_1}\right)^{s_1}, \mathsf{CT}_3 = \left(g^{a_1}\right)^{s_1}, \mathsf{CT}_4 = \left(g^{b a_2}\right)^{s_2} \\
\mathsf{CT}_5 &= \left(g^{a_2}\right)^{s_2}, \mathsf{CT}_6 = \tau_1^{s_1} \tau_2^{s_2}, \mathsf{CT}_7 = \left(\tau_1^b\right)^{s_1} \left(\tau_2^b\right)^{s_2} w^{-t} \\
\mathsf{CT}_8 &= \left(w^{b_1} \cdot g^{\alpha_2 b_2} \cdots g^{\alpha_k b_k}\right)^t = g^{t \langle \vec{\alpha}, \vec{b} \rangle}, \mathsf{CT}_9 = g^t.
\end{aligned}$$

$\mathsf{Decrypt}(\mathsf{PK}, \mathsf{sk}_{\vec{x},\mathsf{ID}}, \mathsf{CT})$: The decryption algorithm will start by computing:

$$\begin{aligned}
A_1 &= e\left(\mathsf{CT}_1, D_1\right) e\left(\mathsf{CT}_2, D_2\right) e\left(\mathsf{CT}_3, D_3\right) e\left(\mathsf{CT}_4, D_4\right) e\left(\mathsf{CT}_5, D_5\right) \\
&= e\left(g^{bs}, g^{\langle \vec{\alpha}, \vec{x} \rangle a_1} v^d\right) e\left(g^{b a_1 s_1}, g^{-\langle \vec{\alpha}, \vec{x} \rangle} v_1^d g^{z_1}\right) e\left(g^{a_1 s_1}, g^{-b z_1}\right) \\
&\qquad e\left(g^{b a_2 s_2}, v_2^d g^{z_2}\right) e\left(g^{a_2 s_2}, g^{-b z_2}\right) \\
&= e(g,g)^{\langle \vec{\alpha}, \vec{x} \rangle a_1 b s_2} e(v,g)^{bsd} e\left(v_1, g\right)^{a_1 b s_1 d} e\left(v_2, g\right)^{a_2 b s_2 d}.
\end{aligned}$$

Next, the algorithm computes:

$$\begin{aligned} A_2 &= e\left(\mathsf{CT}_6, D_6\right) e\left(\mathsf{CT}_7, D_7\right) \\ &= e(v,g)^{bsd} e\left(v_1, g\right)^{a_1 b s_1 d} e\left(v_2, g\right)^{a_2 b s_2 d} e(g,w)^{-d_1 t}. \end{aligned}$$

Now,

$$A_3 = \frac{A_1}{A_2} = e(g,g)^{\langle \vec{\alpha}, \vec{x} \rangle a_1 b s_2} e(g,w)^{d_1 t}.$$

We compute $e(g,w)^{d_1 t}$ as follows:

$$\begin{aligned} A_4 &= \left(\frac{e\left(K_2^{b_2} \cdots K_k^{b_k}, \mathsf{CT}_9\right)}{e\left(\mathsf{CT}_8, D_7\right)} \right)^{-\frac{\beta_1}{\vec{\beta} \cdot \vec{b}}} \\ &= \left(\frac{e\left(\prod_{i=2}^{k} \left(g^{-\alpha_1 \frac{\beta_i}{\beta_1}} g^{\alpha_i}\right)^{d_1 b_i}, g^t\right)}{e\left(\left(g^{\alpha_1 b_1 + \cdots + \alpha_k b_k}\right)^t, g^{d_1}\right)} \right)^{-\frac{\beta_1}{\vec{\beta} \cdot \vec{b}}} \\ &= \left(\frac{e\left(\left(w^{-\frac{\beta_2 b_2 + \cdots + \beta_k b_k}{\beta_1}} g^{\alpha_2 b_2 + \cdots + \alpha_n b_k}\right)^{d_1}, g^t\right)}{e\left(\left(w^{b_1} \cdot g^{\alpha_2 b_2 + \cdots + \alpha_k b_k}\right)^t, g^{d_1}\right)} \right)^{-\frac{\beta_1}{\vec{\beta} \cdot \vec{b}}} \\ &= e\left(w^{\frac{\vec{\beta} \cdot \vec{b}}{\beta_1}}, g\right)^{d_1 t \cdot \frac{\beta_1}{\vec{\beta} \cdot \vec{b}}} = e(g,w)^{d_1 t}. \end{aligned}$$

Finally, we compute

$$\begin{aligned} \frac{\prod_{i=1}^{k} \mathsf{C}_i^{x_i}}{A_3 / A_4} &= \frac{\prod_{i=1}^{k} e(g,g)^{x_i y_i} e(g,g)^{x_i \alpha_i a_1 b s_2}}{e(g,g)^{\langle \vec{\alpha}, \vec{x} \rangle a_1 b s_2}} \\ &= \frac{e(g,g)^{\langle \vec{x}, \vec{y} \rangle} e(g,g)^{\langle \vec{\alpha}, \vec{x} \rangle a_1 b s_2}}{e(g,g)^{\langle \vec{\alpha}, \vec{x} \rangle a_1 b s_2}} \\ &= e(g,g)^{\langle \vec{x}, \vec{y} \rangle}, \end{aligned}$$

and output $\langle \vec{x}, \vec{y} \rangle = \log_{e(g,g)} e(g,g)^{\langle \vec{x}, \vec{y} \rangle}$.

Theorem 6.4

Under the DLIN and BDDH assumption, the construction is adaptively secure.

Proof. This proof is analogous as in [LSW10] and [AL10]. By using the dual system methodology, we will define (normal, semi-functional) keys and ciphertexts. A normal key and ciphertext are generated by the real scheme and the normal key can decrypt both normal ciphertext and semi-functional ciphertext. A semi-functional key can only decrypt a normal ciphertext and it cannot decrypt a semi-functional ciphertext.

Semi-functional keys: By calling $\mathsf{Extract}$ to generate a decryption key for an identity ID. We denote

$$(D'_1, D'_2, D'_3, D'_4, D'_5, D'_6, D'_7, K'_2, \ldots, K'_k)\,.$$

by the normal key.
We then generate the semi-functional key as follows

$$(D_1, D_2, D'_3, D_4, D'_5, D'_6, D'_7, K'_2, \ldots, K'_k)\,,$$

where we define

$$D_1 = D'_1 \cdot g^{-a_1 a_2 \gamma}, D_2 = D'_2 \cdot g^{a_2 \gamma}, D_4 = D'_4 \cdot g^{a_1 \gamma},$$

in which $\gamma \in \mathbb{Z}_p$ is picked randomly.
Semi-Functional Ciphertexts: From a normal ciphertext

$$(\mathsf{CT}'_0, \mathsf{CT}'_1, \mathsf{CT}'_2, \mathsf{CT}'_3, \mathsf{CT}'_4, \mathsf{CT}'_5, \mathsf{CT}'_6, \mathsf{CT}'_7, \mathsf{CT}'_8, \mathsf{CT}'_9)\,,$$

we pick randomly $\chi \in \mathbb{Z}_p$ and replace $(\mathsf{CT}'_4, \mathsf{CT}'_5, \mathsf{CT}'_6, \mathsf{CT}'_7)$ by

$$\mathsf{CT}_4 = \mathsf{CT}'_4 \cdot g^{b a_2 \chi}, \mathsf{CT}_5 = \mathsf{CT}'_5 \cdot g^{a_2 \chi}, \mathsf{CT}_6 = \mathsf{CT}'_6 \cdot v_2^{a_2 \chi}, \mathsf{CT}_7 = \mathsf{CT}'_7 \cdot v_2^{a_2 b \chi}.$$

We will prove the Theorem by a following sequence of games:
$\mathsf{Game}_{\mathsf{Real}}$ is the real security game. We denote $\mathsf{Game}_{\mathsf{Real}}\mathsf{Adv}_{\mathcal{A}}$ as the advantage of an adversary $\mathcal{A}$ in the game.
Game_0: This is the same as $\mathsf{Game}_{\mathsf{Real}}$ except that the ciphertext given to the attacker $\mathcal{A}$ is semi-functional.
Game_κ (for $1 \leq \kappa \leq q$) is identical to Game_0 but the first i decryption key generation queries are answered by returning a semi-functional key. We note that in Game_q the ciphertext and all the keys are semi-functional.
$\mathsf{Game}_{\mathsf{Final}}$: This is the same as Game_q except that the ciphertext is a semi-functional encryption of a random message instead of $\vec{y}_\beta$.

The proof uses the indistinguishability between two consecutive games under some assumptions. The sequence starts from $\mathsf{Game}_{\mathsf{Real}}$ and stops at $\mathsf{Game}_{\mathsf{Final}}$ where the ciphertext is random and the adversary does not have any advantage.

The indistinguishability between $\mathsf{Game}_{\mathsf{Real}}$ and Game_0 is stated and proved as follows:

Lemma 6.4

Under the DLIN assumption, $\mathsf{Game}_{\mathsf{Real}}$ and Game_0 are computationally indistinguishable .

Proof

The simulator $\mathcal{B}$ is given an instance of the DLIN problem

$$(G, g, f, \nu, g^{c_1}, f^{c_2}, T)\,,$$

$\mathcal{B}$ must decide whether $T = \nu^{c_1+c_2}$ or is random. To do this, $\mathcal{B}$ will call on $\mathcal{A}$ by simulating either $\mathsf{Game}_{\mathsf{Real}}$ or Game_0. $\mathcal{A}$ first sends a set $S = \{\mathsf{ID}_1, \ldots, \mathsf{ID}_r\}$ to $\mathcal{B}$.
$\mathcal{B}$ **simulate** PK: The simulator $\mathcal{B}$ chooses randomly $b, \alpha_1, \ldots, \alpha_k, y_v, y_{v_1}, y_{v_2} \in$

$\mathbb{Z}_q$ and random group elements $w, h \in G$. It then sets

$$g = g, g^{a_1} = f, g^{a_2} = \nu, w = w, h = h.$$

The values a_1, a_2 are unknown to the simulator $\mathcal{B}$. $\mathcal{B}$ sets:

$$g^b, g^{ba_1} = f^b, g^{ba_2} = \nu^b, v = g^{y_v}, v_1 = g^{y_{v_1}}, v_2 = g^{y_{v_2}}.$$

It then can computes

$$\begin{aligned} \tau_1 &= v v_1^{a_1} = v f^{y_{v_1}} \\ \tau_2 &= v v_2^{a_2} = v g^{y_{v_2}} \\ e(g,g)^{\alpha_1 a_1 b} &= e(g,f)^{\alpha_1 b}, \ldots, e(g,g)^{\alpha_k a_1 b} = e(g,f)^{\alpha_k b}. \end{aligned}$$

Finally, $\mathcal{B}$ sends PK to $\mathcal{A}$.
Decryption Key Extraction: $\mathcal{B}$ can generate decryption keys for ID_i for all $\mathsf{ID}_i \in \mathcal{R}$. It can be done because the simulator $\mathcal{B}$ knows

$$\mathsf{MSK} = (g, v, v_1, v_2, \vec{\alpha} = (\alpha_1, \ldots, \alpha_k)).$$

Simulate Challenge Ciphertext: After obtaining the public key PK and decryption keys for all identities of $\mathcal{R} = \{\mathsf{ID}_1, \ldots, \mathsf{ID}_r\}$, the adversary $\mathcal{A}$ sends $\mathcal{B}$ two messages $\vec{y}_0, \vec{y}_1$. The simulator $\mathcal{B}$ chooses a random value $\beta \in \{0,1\}$ and generate a semi-functional ciphertext for $\vec{y}_\beta$ and revoked set $\mathcal{R}$ as follows:

1. First, the simulator $\mathcal{B}$ chooses randomly, s'_1, s'_2, t' and calculates a normal ciphertext

   $$(\mathsf{CT}'_0, \mathsf{CT}'_1, \mathsf{CT}'_2, \mathsf{CT}'_3, \mathsf{CT}'_4, \mathsf{CT}'_5, \mathsf{CT}'_6, \mathsf{CT}'_7, \mathsf{CT}'_8, \mathsf{CT}'_9),$$

   where

   $$\mathsf{CT}'_0 = (\mathsf{C}'_1, \mathsf{C}'_2, \ldots, \mathsf{C}'_k).$$

2. $\mathcal{B}$ keeps fix terms $\mathsf{CT}_8 = \mathsf{CT}'_8, \mathsf{CT}_9 = \mathsf{CT}'_9$.

It simulates the other components as follows:

$$\begin{gathered} \mathsf{CT}_0 = (\mathsf{C}_1, \ldots, \mathsf{C}_k); \mathsf{C}_j = \mathsf{C}'_j \left(e\left(g^{c_1}, f\right) e\left(g, f^{c_2}\right)\right)^{b\alpha_j}, j \in [k], \\ \mathsf{CT}_1 = \mathsf{CT}'_1 \left(g^{c_1}\right)^b, \mathsf{CT}_2 = \mathsf{CT}'_2 \left(f^{c_2}\right)^{-b}, \\ \mathsf{CT}_3 = \mathsf{CT}'_3 \left(f^{c_2}\right)^{-1}, \mathsf{CT}_4 = \mathsf{CT}'_4 (T)^b, \mathsf{CT}_5 = \mathsf{CT}'_5 T, \\ \mathsf{CT}_6 = \mathsf{CT}'_6 \left(g^{c_1}\right)^{y_v} \left(f^{c_2}\right)^{-y_{v_1}} T^{y_{v_2}}, \mathsf{CT}_7 = \mathsf{CT}'_7 \left(\left(g^{c_1}\right)^{y_v} \left(f^{c_2}\right)^{-y_{v_1}} T^{y_{v_2}}\right)^b. \end{gathered}$$

We consider 2 cases:

1. If $T = \nu^{c_1+c_2}$, ciphertext has the distribution of a normal ciphertext with $s_1 = -c_2 + s'_1, s_2 = c_1 + c_2 + s'_2$, and $s = s_1 + s_2 = c_1 + s'_1 + s'_2$.

2. If T is random, this will be a properly distributed semi-functional ciphertext.

Therefore, whenever $\mathcal{A}$ has some advantage in distinguishing $\mathsf{Game}_{\mathsf{Real}}$ from Game_0, the simulator can distinguish between the random tuple and DLIN tuple with the same advantage. ■

Lemma 6.5

Under the DLIN assumption, Game_j and Game_{j-1} are computationally indistinguishable, for each $j \in [\mathbf{q}]$.

Proof

We assume that there exists an adversary $\mathcal{A}$ who can distinguish Game_j from Game_{j-1}. We need to build an simulator $\mathcal{B}$ to break the DLIN assumption. $\mathcal{B}$ takes as input a DLIN instance $(\mathbb{G}, g, f, \nu, g^{c_1}, f^{c_2}, T)$ and has to decide whether $T = \nu^{c_1+c_2}$ or not.

Init. The adversary $\mathcal{A}$ first sends the vectors $\vec{\beta}_1, \ldots, \vec{\beta}_q$ to the challenger $\mathcal{B}$. We are considering the j^{th} vector $\vec{\beta}_j$ as $(\beta_1, \ldots, \beta_k)$.

Setup. The algorithm $\mathcal{B}$ first randomly chooses $\alpha, \alpha_1, \ldots, \alpha_k, a_1, a_2, y_{v_1}, y_{v_2} \xleftarrow{\$} \mathbb{Z}_p$ and sets $g = g$.

$$\begin{array}{llll} A_1 = g^{a_1}, & A_2 = g^{a_2}, & B = g^b = f, & v_1 = \nu^{a_2} \cdot g^{y_{v_1}} \\ B_1 = g^{ba_1} = f^{a_1}, & B_2 = g^{ba_2} = f^{a_2}, & v = \nu^{-a_1 a_2}, & v_2 = \nu^{a_1} \cdot g^{y_{v_2}} \end{array}$$

We set $e(g,g)^{\alpha_1 a_1 b} = e(f,g)^{\alpha_1 a_1}, \ldots, e(g,g)^{\alpha_k a_1 b} = e(f,g)^{\alpha_k a_1}$, and then define

$$\tau_1 = v v_1^{a_1} = g^{y_{v_1} a_1}, \quad \tau_2 = v v_2^{a_2} = g^{y_{v_2} a_2}, \quad \tau_1^b = f^{y_{v_1} a_1}, \quad \tau_2^b = f^{y_{v_2} a_2}.$$

The simulator $\mathcal{B}$ picks randomly $y_w, \delta_1, \ldots, \delta_n \xleftarrow{\$} \mathbb{Z}_q$ and defines $w = f \cdot g^{y_w}$. For $i = 2, \ldots, k$, we set

$$h_i = w^{\beta_i/\beta_1} \cdot g^{\delta_i}$$

The simulator $\mathcal{B}$ uses the master secret

$$\mathsf{MSK} = (v, v_1, v_2, \vec{\alpha} = (\alpha_1, \ldots, \alpha_k))$$

Key Queries. When $\mathcal{A}$ makes the m^{th} private key query, $\mathcal{B}$ does as follows.
Case $m > j$. It generates a normal key, using the master secret key msk.
Case $m < j$. It creates a semi-functional key using $g^{a_1 a_2}$.
Case $m = j$. In this case, it generates a key by computing

$$\begin{array}{lll} D_1 = D'_1 \cdot T^{-a_1 a_2}, & D_2 = D'_2 \cdot T^{a_2} \cdot (g^{c_1})^{y_{v_1}}, & D_3 = D'_3 \cdot (f^{c_2})^{y_{v_1}} \\ D_4 = D'_4 \cdot T^{a_1} \cdot (g^{c_1})^{y_{v_2}}, & D_5 = D'_5 \cdot (f^{c_2})^{y_{v_2}}, & D_6 = D'_6 \cdot f^{c_2}. \end{array}$$

and $D_7 = D_7' \cdot (g^{c_1})$, as well as elements

$$K_i = K_i' \cdot (g^{c_1})^{\delta_i} \qquad \text{for } i = 2, \ldots, k,$$

which are all computable since we have $w^{\beta_i/\beta_1} \cdot h_i = g^{\delta_i}$ and

$$K_i = K_i' \cdot (g^{c_1})^{\delta_1} = \left(w^{\beta_i/\beta_1} h_i\right)^{r_1'+c_1},$$

with $r_1' = \log_g(D_7')$. We consider 2 cases:

1. If $T = \nu^{c_1+c_2}$, the decryption key $\mathsf{sk}_{\vec{\beta}_j} = (D_1, \ldots, D_7, K_2, \ldots, K_n)$ forms a normal key where

$$r_1 = r_1' + \theta_1, r_2 = r_2' + c_2, z_1 = z_1' - y_{v_1} c_2, z_2 = z_2' - y_{v_2} c_2$$

2. If T is random, it can be expressed as $T = \nu^{c_1+c_2} \cdot g^{\gamma}$ for some $\gamma \xleftarrow{\$} \mathbb{Z}_q$, so that the decryption key $\mathsf{sk}_{\vec{\beta}_j}$ is distributed as a semi-functional decryption key.

Challenge. The adversary $\mathcal{A}$ sends to the challenger $\mathcal{B}$ two messages $\vec{y}_0, \vec{y}_1$ and a vector $\vec{b}^\star = (b_1^\star, \ldots, b_k^\star)$ such that $\vec{\beta}_i \cdot \vec{b}^\star = 0$ for each $i \in \{1, \ldots, q\}$. The simulator $\mathcal{B}$ takes randomly $\beta \xleftarrow{\$} \{0,1\}$ and generates a normal encryption of $\vec{y}_\beta$

$$(\mathsf{CT}_0', \mathsf{CT}_1', \mathsf{CT}_2', \mathsf{CT}_3', \mathsf{CT}_4', \mathsf{CT}_5', \mathsf{CT}_6', \mathsf{CT}_7', \mathsf{CT}_8', \mathsf{CT}_9'),$$

where

$$\mathsf{CT}_0' = (\mathsf{C}_1', \mathsf{C}_2', \ldots, \mathsf{C}_k').$$

$\mathcal{B}$ then picks randomly $\chi \xleftarrow{\$} \mathbb{Z}_q$ and simulates the ciphertext as

$$\mathsf{CT}_4 = \mathsf{CT}_4' \cdot f^{a_2 \cdot \chi}, \quad \mathsf{CT}_5 = \mathsf{CT}_5' \cdot g^{a_2 \cdot \chi}, \quad \mathsf{CT}_6 = \mathsf{CT}_6' \cdot v_2^{a_2 \cdot \chi}$$

$$\mathsf{CT}_7 = \mathsf{CT}_7' \cdot \nu^{-y_w \cdot a_1 \cdot a_2 \cdot \chi} \cdot f^{y_{v_2} \cdot a_2 \cdot \chi} = T_1^{s_1'} \cdot T_2^{s_2'} \cdot w^{-t'} \cdot \nu^{-y_w \cdot a_1 \cdot a_2 \cdot \chi} \cdot f^{y_{v_2} \cdot a_2 \cdot \chi}$$

$$\mathsf{CT}_8 = \mathsf{CT}_8' \cdot \left(\nu^{\sum_{i=2}^n y_i^\star \delta_i}\right)^{a_1 \cdot a_2 \cdot \chi}, \mathsf{CT}_9 = \mathsf{CT}_9' \cdot \nu^{a_1 \cdot a_2 \cdot \chi}.$$

The semi-functional component CT_7 is created by implicitly setting $t = \log_g(\mathsf{CT}_9)$ as $t = t' + \log_g(\nu) a_1 a_2 \chi$. Since $\vec{\beta} \cdot \vec{b}^\star = 0$, we have

$$\begin{aligned}
\mathsf{CT}_8 &= \left(w^{b_1^\star} \cdot h_2^{b_2^\star} \cdots h_k^{b_k^\star}\right)^t = \left(w^{b_1^\star} \cdot \left(w^{\beta_2/\beta_1} g^{\delta_2}\right)^{b_2^\star} \cdots \left(w^{\beta_k/\beta_1} g^{\delta_k}\right)^{b_k^\star}\right)^t \\
&= \left(w^{\vec{\beta} \cdot \vec{b}^\star/\beta_1} \cdot \left(g^{\delta_2}\right)^{y_2^\star} \cdots \left(g^{\delta_k}\right)^{y_k^\star}\right)^t = \left(g^{\sum_{j=2}^n \delta_j y_j^\star}\right)^t \\
&= \mathsf{CT}_8' \cdot \left(\nu^{\sum_{i=2}^k y_i^\star \delta_i}\right)^{a_1 \cdot a_2 \cdot \chi}
\end{aligned}$$

We then can conclude that

$$(\mathsf{CT}_0', \mathsf{CT}_1', \mathsf{CT}_2', \mathsf{CT}_3', \mathsf{CT}_4', \mathsf{CT}_5', \mathsf{CT}_6', \mathsf{CT}_7', \mathsf{CT}_8', \mathsf{CT}_9')$$

is distributed as a semi-functional ciphertext.

We consider 2 cases:

1. If $T = \nu^{c_1+c_2}$, the simulator $\mathcal{B}$ guesses correctly and $\mathcal{B}$ simulated the Game_{j-1}.
2. If T is random, $\mathcal{B}$ simulated the Game_j.

■

Lemma 6.6

Under the BDDH assumption, $\mathsf{Game}_{\mathbf{q}}$ and $\mathsf{Game}_{\mathsf{Final}}$ are computationally indistinguishable.

Proof

We assume that there exists an adversary $\mathcal{A}$ who can distinguish $\mathsf{Game}_{\mathbf{q}}$ from $\mathsf{Game}_{\mathsf{Final}}$. We need to build a simulator $\mathcal{B}$ to break the BDDH assumption. The simulator $\mathcal{B}$ will obtain an instance of the BDDH problem:

$$(g, g^{c_1}, g^{c_2}, g^{c_3}, T)\,.$$

It must decide whether $T = e(g,g)^{c_1 c_2 c_3}$ or T is random.
Init. The adversary $\mathcal{A}$ first sends the vectors $\vec{\beta}_1, \ldots, \vec{\beta}_q$ to the challenger $\mathcal{B}$.
Setup. The algorithm $\mathcal{B}$ first randomly chooses

$$b, \alpha, \alpha_1, \ldots, \alpha_k, a_1, a_2, y_v, y_{v_1}, y_{v_2}, y_w \in \mathbb{Z}_q,$$

and sets $g = g$ and the master public key is defined as follows:

$$g^b, g^{a_1} = g^{c_1}, g^{a_2} = g^{c_2}, g^{ba_1}, g^{ba_2} = (g^{c_2})^b, v = g^{y_v}, v_1 = g^{y_{v_1}}$$
$$v_2 = g^{y_{v_2}}, w = g^{y_w}, h_1 = g^{\alpha_1}, \ldots, h_k = g^{\alpha_k}$$
$$e(g,g)^{a_1\alpha_1 b}, \ldots, e(g,g)^{a_1\alpha_k b}.$$

Key Queries. We will simulate semi-functional keys for $\vec{\beta}_1, \ldots, \vec{\beta}_{\mathbf{q}}$. For each $\vec{\beta}_i = (\beta_1, \ldots, \beta_k)$, we take randomly $d_1, d_2, z_1, z_2, \gamma' \in \mathbb{Z}_q$ and set $d = d_1 + d_2$. The components of decryption key are calculated as follows:

$$D_1 = (g^{c_2})^{-\gamma' a_1} v^d, D_2 = (g^{c_2})^{\gamma'} v_1^d g^{z_1}, D_3 = \left(g^b\right)^{-z_1}, D_4 = (g^{c_1})^{a_1} g^{a_1\gamma'} v_2^d g^{z_2}$$
$$D_5 = g^{-bz_2}, D_6 = g^{d_2 b}, D_7 = g^{d_1},$$
$$K_2 = \left(h_1^{\beta_2/\beta_1} \cdot h_2\right)^{d_1}, \ldots, K_k = \left(h_1^{\beta_k/\beta_1} \cdot h_k\right)^{d_1}.$$

Challenge. The adversary $\mathcal{A}$ sends to the challenger $\mathcal{B}$ two messages $\vec{y}_0, \vec{y}_1$ and a vector $\vec{b}^\star = (b_1^\star, \ldots, b_k^\star)$ such that $\vec{\beta}_i \cdot \vec{b}^\star = 0$ for each $i \in \{1, \ldots, \mathbf{q}\}$. The simulator $\mathcal{B}$ takes randomly $\beta \xleftarrow{\$} \{0,1\}$ and generates either a semi-functional ciphertext for $\vec{y}_\beta$ or a semi-functional encryption of a random message as follows:

$\mathcal{B}$ takes randomly exponents s_1, t, χ'. It forms the ciphertext as:

$$\mathsf{CT}_0 = (\mathsf{C}_1, \ldots, \mathsf{C}_k) = \left(e(g,g)^{y_{\beta,1} T^{\alpha_1 a_1 b}}, \ldots, e(g,g)^{y_{\beta,k} T^{\alpha_k a_1 b}}\right)$$
$$\mathsf{CT}_1 = g^{s_1 b} \left(g^{c_3}\right)^b, \mathsf{CT}_2 = g^{b a_1 s_1}, \mathsf{CT}_3 = g^{a_1 s_1}, \mathsf{CT}_4 = \left(g^{c_2}\right)^{\chi' b},$$
$$\mathsf{CT}_5 = \left(g^{c_2}\right)^{\chi'}, \mathsf{CT}_6 = \tau_1^{s_1} \left(g^{c_3}\right)^{y_v} \left(g^{c_2}\right)^{y_{v_2} \chi'}, \mathsf{CT}_7 = \left(\tau_1^b\right)^{s_1} \left(g^{c_3}\right)^{y_v b} \left(g^{c_2}\right)^{y_{v_2} \chi' b} w^{-t},$$
$$\mathsf{CT}_8 = (h_1^{b_1^\star} \cdots h_k^{b_k^\star})^t, \mathsf{CT}_9 = g^t.$$

We implicitly set $s_2 = c_3$ and $\chi = -c_3 + \chi'$.

1. If $T = e(g,g)^{c_1 c_2 c_3}$, the ciphertext is distributed as a semi-functional encryption of $\vec{y}_\beta$.

2. If T is random, the ciphertext is distributed as a semi-functional encryption of a random message.

Therefore $\mathcal{B}$ can use $\mathcal{A}$'s output to distinguish $T = e(g,g)^{c_1 c_2 c_3}$ from random with the same advantage that $\mathcal{A}$ has in distinguishing $\mathsf{Game}_{\mathbf{q}}$ from $\mathsf{Game}_{\mathsf{Final}}$. ■

6.4 Towards Fine-grained Revocable Functional Encryption for Inner Product

In this section, we propose a new research direction on revocable functional encryption. The motivation is as follows:

From the definition of revocable functional encryption, we can observe that for each of the identity ID appearing in list $\mathcal{R}$, every evaluation of functions F which are assigned to the identity ID will be disable. We consider the following situation (in PayTV context) each subscriber can use the decode functionality over many channels of a content provider. When a user wants to unsubcribe some channels, he will request the provider to stop charging him over these channels. It is clear that the above definition does not cover this situation because once an identity is revoked, the service provider will disable decryption over every channel corresponding to this identity. We will need a fine-grained solution in which the encryption algorithm will allow us to exclude some identities from decryption for one functionality while preserving decryption for the other functionalities. We call it as a functional encryption scheme with fine-grained revocation. The main difference from the previous definition of revocable functional encryption is in the encrypt function: The set $\mathcal{R}$ consists of elements in the form of a pair (ID, F) instead of just ID in the previous case.

Definition 6.3

A *fine-grained revocable functional encryption* scheme frFE consists of four algorithms $(\mathsf{Setup}, \mathsf{Extract}, \mathsf{Encrypt}, \mathsf{Decrypt})$ which is defined as follows:

$\mathsf{Setup}(1^\lambda)$: Takes as input a security parameter λ and outputs a master key pair $(\mathsf{PK}, \mathsf{MSK})$.

$\mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, F)$: Given an identity ID of an user, a circuit $F \in \mathcal{F}_\lambda$ and the master secret key MSK, this algorithm outputs a secret key $\mathsf{sk}_{F,\mathsf{ID}}$.

$\mathsf{Encrypt}(\mathsf{PK}, y, \mathcal{R} = \{(\mathsf{ID}, F)\})$: Takes as input the public key PK, a message $y \in \mathcal{Y}_\lambda$, a list of $\mathcal{R}$, this randomized algorithm will output a ciphertext CT.

$\mathsf{Decrypt}(\mathsf{PK}, \mathsf{sk}_{F,\mathsf{ID}}, \mathsf{CT})$: Given the master public key PK, a secret key $\mathsf{sk}_{F,\mathsf{ID}}$ and a ciphertext CT, this algorithm outputs $F(y) \in \mathcal{S}_\lambda$ or an invalid symbol $\perp$.

For correctness, we require that for all $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$, all $y \in \mathcal{Y}_\lambda$, each $F \in \mathcal{F}_\lambda$ and all identities $(\mathsf{ID}, F) \notin \mathcal{R}$, $\mathsf{sk}_{F,\mathsf{ID}} \leftarrow \mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, F)$, if $\mathsf{CT} \leftarrow \mathsf{Encrypt}(\mathsf{PK}, y, \mathcal{R})$ then one should get $\mathsf{Decrypt}(\mathsf{PK}, \mathsf{sk}_{F,\mathsf{ID}}, \mathsf{CT}) = F(y)$, with overwhelming probability.

The security of revocable functional encryption scheme frFE will be defined as follows:

Definition 6.4

A fine-grained revocable functional encryption scheme frFE for a list $\mathcal{R}$,

$$\mathsf{frFE} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Encrypt}, \mathsf{Decrypt})$$

is semantically secure under chosen-plaintext attacks (or $\mathsf{IND} - \mathsf{CPA}$ security) if no PPT adversary has non-negligible advantage in the following game:

- The challenger $\mathcal{B}$ runs $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$ and the master public key PK is given to the adversary $\mathcal{A}$.
- The adversary adaptively makes secret key queries to the challenger. That is, the adversary $\mathcal{A}$ chooses some pairs of identities ID and functions $F \in \mathcal{F}_\lambda$. $\mathcal{A}$ sends them to $\mathcal{B}$ and then obtains $\mathsf{sk}_{F,\mathsf{ID}} \leftarrow \mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, F)$ from $\mathcal{B}$.
- We denote $\mathcal{C} = \mathcal{ID}_C \times \mathcal{F}_C \subseteq \mathcal{ID} \times \mathcal{F}$ as the set of all elements that have been asked for the decryption key by the adversary $\mathcal{A}$. The adversary $\mathcal{A}$ chooses distinct messages $y_0, y_1 \in \mathcal{Y}_\lambda$, a list $\mathcal{R} = \mathcal{ID}_R \times \mathcal{F}_R$ which each of element has the form $\{(\mathsf{ID}, F)\}$ such that:
  1. $\mathcal{ID}_R \subseteq \mathcal{ID}_C$.

2. For each element $(\mathsf{ID}, F) \in \mathcal{C}$ and if $(\mathsf{ID}, F) \notin \mathcal{R}$, the following conditions need to be satisfied:

$$F(y_0) = F(y_1).$$

This restriction is required in all functional encryption to avoid trivial attacks.

Whenever $\mathcal{B}$ receives the messages, it randomly picks $\beta \hookleftarrow \{0,1\}$ and then transfers to $\mathcal{A}$ a ciphertext $\mathsf{CT}_\beta = \mathsf{Encrypt}(\mathsf{PK}, y_\beta, \mathcal{R})$.

- Adversary $\mathcal{A}$ eventually returns a guess β' for a bit β and wins if $\beta' = \beta$.

Selective security. When the messages y_0, y_1, a list of revoked elements $\mathcal{R}$ for the challenge ciphertext are chosen before the Setup algorithm started, the frFE scheme is said selectively-security against chosen-plaintext attacks, which is denoted by sel−IND−CPA.

A candidate construction for fine-grained revocable inner product functional encryption. We will give a candidate construction for fine-grained revocable inner product functional encryption. The construction is in two steps. In Step 1, by fixing a function, we will construct a scheme in which one will revoke an arbitrary set of users corresponding to the function. In step 2, we extend this construction for many functions.

Step 1. Revocable Functional Encryption for One Function.

Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ a bilinear map, where $\mathbb{G}, \mathbb{G}_T$ are cyclic groups of order q, written multiplicatively.

$\mathsf{Setup}(1^\lambda, 1^k)$**:** This algorithm generates a bilinear pairing $(\mathbb{G}, \mathbb{G}_T, q, e)$ for sufficiently large prime order q and g is a generator of the group $\mathbb{G}$. The bilinear map e over $(\mathbb{G}, \mathbb{G})$ can be calculated efficiently.

- For each $i \in \{1, \ldots, k\}$, randomly choose $s_i \xleftarrow{\$} \mathbb{Z}_q$. We set $\vec{s} = (s_1, \ldots, s_k)$.
- We take randomly $\alpha \xleftarrow{\$} \mathbb{Z}_q$.
- Let $\mathcal{H}_1, \mathcal{H}_2 : \mathbb{Z}_q^k \to \mathbb{G}$ be two cryptographic hash functions.
- The public key is $\mathsf{PK} = \left(\mathbb{G}, g, \mathcal{H}_1, \mathcal{H}_2, e(g,g)^{s_1\alpha}, \ldots, e(g,g)^{s_k\alpha}\right)$.
- The master secret key is $\mathsf{MSK} = \{\alpha, g^{s_1}, \ldots, g^{s_k}\}$.

$\mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, \vec{x})$**:** Takes as input an identity $\mathsf{ID} \in \mathbb{Z}_q$, the master secret key MSK and a characteristic vector $\vec{x} = (x_1, \ldots, x_k) \in \mathbb{Z}_q^k$. For each identity ID, we take randomly $t \xleftarrow{\$} \mathbb{Z}_q$ and the secret key $\mathsf{sk}_{\vec{x},\mathsf{ID}}$ is a tuple:

$$\mathsf{sk}_0 = g^t, \mathsf{sk}_1 = g^{\alpha\langle\vec{s},\vec{x}\rangle} \cdot \mathcal{H}_2(\vec{x})^t, \mathsf{sk}_2 = \left(\mathcal{H}_1(\vec{x})\mathcal{H}_2(\vec{x})^{\mathsf{ID}}\right)^t.$$

$\mathsf{Encrypt}(\mathcal{R}, \mathsf{PK}, \vec{y}, \vec{x})$**:** Takes as input the set $\mathcal{R}$ of m identities $\{\mathsf{ID}_1, \ldots, \mathsf{ID}_m\}$ associating to a function $\vec{x}$, the public key PK, a message $\vec{y} = (y_1, \ldots, y_k) \in \mathbb{Z}_q^k$.

These secret keys will be revoked. We sample $r \xleftarrow{\$} \mathbb{Z}_q$ chooses random $r_1, \ldots, r_m$ such that $r = r_1 + \ldots + r_m$ and compute

$$\begin{aligned}
\mathsf{CT}_0 &= (\mathsf{C}_1, \mathsf{C}_2, \ldots, \mathsf{C}_k) \\
&= (e(g,g)^{y_1} e(g,g)^{s_1 \alpha r}, \ldots, e(g,g)^{y_k} e(g,g)^{s_k \alpha r} \\
\mathsf{CT}_{j,1} &= \left(\mathcal{H}_1(\vec{x}) \mathcal{H}_2(\vec{x})^{\mathsf{ID}_j}\right)^{r_j} \\
\mathsf{CT}_{j,2} &= g^{r_j},
\end{aligned}$$

for all $j \in [m]$.

$\mathsf{Decrypt}(\mathsf{PK}, \mathsf{sk}_{\vec{x},\mathsf{ID}}, \mathsf{CT})$: We assume that CT is a ciphertext which is assigned to a function $\vec{x}'$. In the case $\mathcal{H}_1(\vec{x}) = \mathcal{H}_1(\vec{x}') = g^{\alpha_2}$, $\mathcal{H}_2(\vec{x}) = \mathcal{H}_2(\vec{x}') = g^{\alpha_1}$ and $\mathsf{ID} \neq \mathsf{ID}_j$ for all $j \in [m]$, we first compute

$$\begin{aligned}
& e(\mathsf{sk}_2, \mathsf{CT}_{2,j})^{\frac{1}{\mathsf{ID}-\mathsf{ID}_j}} e(\mathsf{sk}_0, \mathsf{CT}_{1,j})^{-\frac{1}{\mathsf{ID}-\mathsf{ID}_j}} = \\
&= e\left(\left(\mathcal{H}_1(\vec{x}) \mathcal{H}_2(\vec{x})^{\mathsf{ID}}\right)^t, g^{r_j}\right)^{\frac{1}{\mathsf{ID}-\mathsf{ID}_j}} e\left(g^t, \left(\mathcal{H}_1(\vec{x}') \mathcal{H}_2(\vec{x}')^{\mathsf{ID}_j}\right)^{r_j}\right)^{-\frac{1}{\mathsf{ID}-\mathsf{ID}_j}} \\
&= e(g,g)^{\frac{r_j t(\alpha_1 \mathsf{ID} + \alpha_2)}{\mathsf{ID}-\mathsf{ID}_j}} e(g,g)^{\frac{-r_j t(\alpha_1 \mathsf{ID}_j + \alpha_2)}{\mathsf{ID}-\mathsf{ID}_j}} \\
&= e(g,g)^{\alpha_1 t r_j}.
\end{aligned}$$

Since $r = r_1 + \ldots + r_m$, this implies that

$$\prod_{j=1}^{m} \left(e(\mathsf{sk}_2, \mathsf{CT}_{2,j})^{\frac{1}{\mathsf{ID}-\mathsf{ID}_j}} e(\mathsf{sk}_0, \mathsf{CT}_{1,j})^{-\frac{1}{\mathsf{ID}-\mathsf{ID}_j}} \right) = e(g,g)^{\alpha_1 t r}.$$

We then computes

$$\prod_{j=1}^{m} e(\mathsf{sk}_1, \mathsf{CT}_{2,j}) = \prod_{j=1}^{m} e(g^{\alpha \langle \vec{s}, \vec{x} \rangle + \alpha_1 t}, g^{r_j}) = e(g,g)^{r \alpha \langle \vec{s}, \vec{x} \rangle} e(g,g)^{\alpha_1 t r}.$$

Finally, we compute

$$\begin{aligned}
\frac{e(g,g)^{\alpha_1 t r} \prod_{i=1}^{k} \mathsf{C}_i^{x_i}}{\prod_{j=1}^{m} e(\mathsf{sk}_1, \mathsf{CT}_{2,j})} &= \frac{e(g,g)^{\alpha_1 t r} \prod_{i=1}^{k} e(g,g)^{x_i y_i} e(g,g)^{r \alpha x_i s_i}}{e(g,g)^{r \alpha \langle \vec{s}, \vec{x} \rangle} e(g,g)^{\alpha_1 t r}} \\
&= \frac{e(g,g)^{\langle \vec{x}, \vec{y} \rangle} e(g,g)^{r \alpha \langle \vec{s}, \vec{x} \rangle}}{e(g,g)^{r \alpha \langle \vec{s}, \vec{x} \rangle}} \\
&= e(g,g)^{\langle \vec{x}, \vec{y} \rangle},
\end{aligned}$$

and output $\langle \vec{x}, \vec{y} \rangle = \log_{e(g,g)} e(g,g)^{\langle \vec{x}, \vec{y} \rangle}$.

Step 2. Construction of fine-grained revocable inner product functional encryption. We will construct a fine-grained revocable functional encryption for inner product from revocable one function. We assume that the revocable one function is a 4-tuple $(\mathsf{Setup}, \mathsf{Extract}, \mathsf{Encrypt}, \mathsf{Decrypt})$.

$\Gamma.\mathsf{Setup}(1^\lambda, 1^k)$: This algorithm generates a bilinear pairing $(\mathbb{G}, \mathbb{G}_T, q, e)$ for sufficiently large prime order q and g is a generator of the group $\mathbb{G}$. The bilinear map e over $(\mathbb{G}, \mathbb{G})$ can be calculated efficiently.

- For each $i \in \{1, \ldots, k\}$, randomly choose $s_i \xleftarrow{\$} \mathbb{Z}_q$. We set $\vec{s} = (s_1, \ldots, s_k)$.

- We take randomly $\alpha \xleftarrow{\$} \mathbb{Z}_q$.
- Let $\mathcal{H}_1, \mathcal{H}_2 : \mathbb{Z}_q^k \to \mathbb{G}$ be two cryptographic hash functions.
- The public key is $\mathsf{PK} = \left(\mathbb{G}, g, \mathcal{H}_1, \mathcal{H}_2, e(g,g)^{\alpha s_1}, \ldots, e(g,g)^{\alpha s_k}\right)$.
- The master secret key is $\mathsf{MSK} = \{\alpha, g^{s_1}, \ldots, g^{s_k}\}$.

$\Gamma.\mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, \vec{x})$: Takes as input an identity $\mathsf{ID} \in \mathbb{Z}_q$, the master secret key MSK and a characteristic vector $\vec{x} = (x_1, \ldots, x_k) \in \mathbb{Z}_q^k$. For each identity ID, we take randomly $t \xleftarrow{\$} \mathbb{Z}_q$ and the secret key $\mathsf{sk}_{\vec{x},\mathsf{ID}}$ is a tuple:

$$\mathsf{sk}_0 = g^t, \mathsf{sk}_1 = g^{\alpha\langle \vec{s}, \vec{x}\rangle} \cdot \mathcal{H}_2(\vec{x})^t, \mathsf{sk}_2 = \left(\mathcal{H}_1(\vec{x})\mathcal{H}_2(\vec{x})^{\mathsf{ID}}\right)^t.$$

$\Gamma.\mathsf{Encrypt}(\mathcal{R}, \mathsf{PK}, \vec{y}, \vec{X})$: We assume that $\mathcal{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_m\}$ and $\vec{X} = \{\vec{x}_1, \ldots, \vec{x}_m\}$ in which $\mathcal{R}_i$ is a list of revokers corresponding to the function $\vec{x}_i$. This algorithm will call $\mathsf{Encrypt}(\mathcal{R}_i, \mathsf{PK}, \vec{y}, \vec{x}_i)$ on the same message $\vec{y} = (y_1, \ldots, y_k) \in \mathbb{Z}_q^k$. The ciphertext is generated as follows

$$\mathsf{CT} = (\mathsf{Encrypt}(\mathcal{R}_1, \mathsf{PK}, \vec{y}, \vec{x}_1), \ldots, \mathsf{Encrypt}(\mathcal{R}_m, \mathsf{PK}, \vec{y}, \vec{x}_m)).$$

$\Gamma.\mathsf{Decrypt}(\mathsf{PK}, \mathsf{sk}_{\vec{x},\mathsf{ID}}, \mathsf{CT})$: The decryptor calls the procedure $\mathsf{Decrypt}$ to recover $\langle \vec{x}, \vec{y}\rangle$.

Security. It is very challenging to prove the security of our candidate construction. Different from the case of identity revocation as in [NWZ16] and all classical model, we have to revoke functions as well. The proof probably requires new insighful techniques. We leaves this as an open problem for future works.

7 Conclusion & Disscussion

The contributions presented in this thesis focus on the design of Broadcast Encryption, Traitor Tracing, Revocation, and Trace & Revoke schemes. In this chapter, we will summarize our results and discuss further works.

1. In Chapter 3, we constructed an anonymous broadcast encryption scheme (AnoBEB) based on $k-$ LWE assumption. Our scheme is the first one, which is obtained efficiently as PKE LWE. It is secure against chosen-plaintext attacks.

2. In Chapter 4, we propose a Trace & Revoke system as an application of AnoBEB and the robust-IPP code. The security of this system relies on AnoBEB and our construction is the most efficient trace and revoke scheme for standard black-box tracing in the bounded collusion model.

3. In Chapter 5, we introduce the notion of Traceable Functional Encryption, and we also provide an instantiation for the inner product case. The scheme obtained black-box confirmation, and it is secure under DDH assumption.

4. In Chapter 6, we study revocation problem in functional encryption and propose several pairing-based constructions for inner product functional encryption with short ciphertexts or decryption keys. We construct a revocable inner product functional encryption in the multi-client setting. We extend this primitive to a new notion of *fine-grained revocable functional encryption* and propose an efficient construction.

Perspectives. There are a couple of questions that remain open.

- We provided in Chapter 3 a LWE-based construction of AnoBEB which is as efficient as the underlying LWE PKE. We raise an open question of constructing AnoBEB schemes from other standard encryptions, namely ElGamal, RSA, Paillier encryptions, without a significant loss in the efficiency. This seems to us an interesting and a challenging problem, even for the simplest case of a system of $N = 2$ users. The solution will directly give the most efficient Trace & Revoke systems for bounded collusion model (by instantiating our Trace & Revoke scheme in Chapter 4) from DDH, RSA and DCR assumptions respectively.

- In designing Trace & Revoke systems, there are two main approaches to tackle this problem:

- restrict to bounded collusion model (motivated by the fact that this is a practical scenario) and give efficient solutions;
- consider the full collusion setting (all users can become traitors) and improve theoretical results as there are actually no efficient scheme, say the ciphertext size depends on $\mathsf{polylog}(N)$, from the standard assumptions, without relying on iO or multi-linear maps.

Recently, at STOC '18, Goyal, Koppula and Waters [GKW18b], relying on Mixed Functional Encryption with Attribute-Based Encryption, gave a traitor tracing scheme for full collusion from the LWE assumption with $\mathsf{polylog}(N)$ ciphertext size. It is an interesting open question of constructing a polylog size Trace & Revoke scheme for full collusion from LWE or from a standard assumption, as combining tracing and revoking functionalities is always a difficult problem.

- In Chapter 5 of this thesis, we formalized the concept Traceable Functional Encryption and provided an construction of Traceable Functional Encryption for inner product (Traceable IPFE). This is just a selective scheme and achieves one-target tracing level in the sense that an adversary $\mathcal{A}$ is allowed to ask secret keys for one target function only, but many identities, and then produces a pirate decoder for this function. This leads to several open questions in this direction that follows:

  1. Can we build a Traceable IPFE achieve stronger security with more general security in the sense that the adversary $\mathcal{A}$ can ask secret keys for several functions (instead of just one target function)?
  2. Can we build a Traceable IPFE secure from the Learning With Errors (LWE) or Decisional Composite Residuosity (DCR) assumption?
  3. Can we build a Traceable Functional Encryption for more general functions such as quadratic functions, or any circuit?
  4. Can we build a Traceable Functional Encryption in multi-client setting under standard assumptions?

- In Chapter 6, we studied revocation schemes for inner product functional encryption. There are still several interesting questions that remain open.

  1. Can we construct an anonymous revocable inner product functional encryption? That is, how we can hide a set of revoked user at the time of encryption? or in other words, whether we can construct a revocable inner product functional encryption preserving the user's identity?
  2. Can we build a (fine-grained) revocable IPFE from the Learning With Errors (LWE) or Decisional Composite Residuosity (DCR) assumption? Whether these constructions can be extended into multi-input or multi-client schemes.
  3. Can we build a decentralized multi-client revocable Functional Encryption under standard assumptions?
  4. Can we build a (fine-grained) revocable functional encryption for more general functions such as quadratic functions, or any circuit from standard assumption?

Bibliography

[ABDP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 733–751. Springer, Heidelberg, March / April 2015.

[ABP+17] Shweta Agrawal, Sanjay Bhattacherjee, Duong Hieu Phan, Damien Stehlé, and Shota Yamada. Efficient public trace and revoke from standard assumptions: Extended abstract. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 17*, pages 2277–2293. ACM Press, October / November 2017.

[ACGU20] Michel Abdalla, Dario Catalano, Romain Gay, and Bogdan Ursu. Inner-product functional encryption with fine-grained access control. LNCS, pages 467–497. Springer, Heidelberg, December 2020.

[AKPS12] Murat Ak, Aggelos Kiayias, Serdar Pehlivanoglu, and Ali Aydin Selcuk. Generic construction of trace and revoke schemes. Cryptology ePrint Archive, Report 2012/531, 2012. http://eprint.iacr.org/2012/531.

[AL10] Nuttapong Attrapadung and Benoît Libert. Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 384–402. Springer, Heidelberg, May 2010.

[ALS16] Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 333–362. Springer, Heidelberg, August 2016.

[AP11] J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. *Theor. Comput. Science*, 48(3):535–553, 2011.

[AY20] Shweta Agrawal and Shota Yamada. Optimal broadcast encryption from pairings and LWE. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, LNCS, pages 13–43. Springer, Heidelberg, May 2020.

[BBW06] Adam Barth, Dan Boneh, and Brent Waters. Privacy in encrypted content distribution using private broadcast encryption. In Giovanni Di Crescenzo and Avi Rubin, editors, *FC 2006*, volume 4107 of *LNCS*, pages 52–64. Springer, Heidelberg, February / March 2006.

[Ber91] Shimshon Berkovits. How to broadcast a secret (rump session). In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 535–541. Springer, Heidelberg, April 1991.

[BF99] Dan Boneh and Matthew K. Franklin. An efficient public key traitor tracing scheme. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 338–353. Springer, Heidelberg, August 1999.

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 258–275. Springer, Heidelberg, August 2005.

[BK13] Alexander Barg and Grigory Kabatiansky. Robust parent-identifying codes and combinatorial arrays. *IEEE Trans. Information Theory*, 59(2):994–1003, 2013.

[BN08] Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertext. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 08*, pages 501–510. ACM Press, October 2008.

[BP08] Olivier Billet and Duong Hieu Phan. Efficient traitor tracing from collusion secure codes. In Reihaneh Safavi-Naini, editor, *ICITS 08*, volume 5155 of *LNCS*, pages 171–182. Springer, Heidelberg, August 2008.

[BS95] Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data (extended abstract). In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 452–465. Springer, Heidelberg, August 1995.

[BSW06] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 573–592. Springer, Heidelberg, May / June 2006.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011.

[BW06] Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06*, pages 211–220. ACM Press, October / November 2006.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, December 2013.

[BWZ14] Dan Boneh, Brent Waters, and Mark Zhandry. Low overhead broadcast encryption from multilinear maps. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 206–223. Springer, Heidelberg, August 2014.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499. Springer, Heidelberg, August 2014.

[CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 257–270. Springer, Heidelberg, August 1994.

[CPP05] Hervé Chabanne, Duong Hieu Phan, and David Pointcheval. Public traceability in traitor tracing schemes. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 542–558. Springer, Heidelberg, May 2005.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, April / May 2002.

[Del07] Cécile Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 200–215. Springer, Heidelberg, December 2007.

[DF02] Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption for stateless receivers. In *Digital Rights Management Workshop*, volume 2696 of *Lecture Notes in Computer Science*, pages 61–80. Springer, November 2002.

[DPP07] Cécile Delerablée, Pascal Paillier, and David Pointcheval. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, editors, *PAIRING 2007*, volume 4575 of *LNCS*, pages 39–59. Springer, Heidelberg, July 2007.

[DPP20] Xuan Thanh Do, Duong Hieu Phan, and David Pointcheval. Traceable inner product functional encryption. In *CT-RSA 2020*, LNCS, pages 564–585. Springer, Heidelberg, 2020.

[DPY20] Xuan Thanh Do, Duong Hieu Phan, and Moti Yung. A concise bounded anonymous broadcast yielding combinatorial trace-and-revoke schemes. LNCS, pages 145–164. Springer, Heidelberg, 2020.

[FN94] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 480–491. Springer, Heidelberg, August 1994.

[FNP07] Nelly Fazio, Antonio Nicolosi, and Duong Hieu Phan. Traitor tracing with optimal transmission rate. In Juan A. Garay, Arjen K. Lenstra, Masahiro Mambo, and René Peralta, editors, *ISC 2007*, volume 4779 of *LNCS*, pages 71–88. Springer, Heidelberg, October 2007.

[FP12] Nelly Fazio and Irippuge Milinda Perera. Outsider-anonymous broadcast encryption with sublinear ciphertexts. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 225–242. Springer, Heidelberg, May 2012.

[Fre10] David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 44–61. Springer, Heidelberg, May / June 2010.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, Heidelberg, May 2013.

[GKM+19] Rishab Goyal, Sam Kim, Nathan Manohar, Brent Waters, and David J. Wu. Watermarking public-key cryptographic primitives. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2019, Part III*, LNCS, pages 367–398. Springer, Heidelberg, August 2019.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 555–564. ACM Press, June 2013.

[GKSW10] Sanjam Garg, Abishek Kumarasubramanian, Amit Sahai, and Brent Waters. Building efficient fully collusion-resilient traitor tracing and revocation schemes. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 10*, pages 121–130. ACM Press, October 2010.

[GKW18a] Romain Gay, Lucas Kowalczyk, and Hoeteck Wee. Tight adaptively secure broadcast encryption with short ciphertexts and keys. In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 123–139. Springer, Heidelberg, September 2018.

[GKW18b] Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th ACM STOC*, pages 660–670. ACM Press, June 2018.

[GKW19] Rishab Goyal, Venkata Koppula, and Brent Waters. New approaches to traitor tracing with embedded identities. In *TCC 2019, Part II*, LNCS, pages 149–179. Springer, Heidelberg, March 2019.

[GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proc. of STOC*, pages 197–206. ACM, 2008. Full version available at http://eprint.iacr.org/2007/432.pdf.

[GQWW19] Rishab Goyal, Willy Quach, Brent Waters, and Daniel Wichs. Broadcast and trace with N^{ϵ} ciphertext size from standard assumptions. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2019, Part III*, LNCS, pages 826–855. Springer, Heidelberg, August 2019.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, Heidelberg, August 2012.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 545–554. ACM Press, June 2013.

[GVW19] Rishab Goyal, Satyanarayana Vusirikala, and Brent Waters. Collusion resistant broadcast and trace from positional witness encryption. In *PKC 2019, Part II*, LNCS, pages 3–33. Springer, Heidelberg, 2019.

[GW09] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 171–188. Springer, Heidelberg, April 2009.

[HS02] Dani Halevy and Adi Shamir. The LSD broadcast encryption scheme. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 47–60. Springer, Heidelberg, August 2002.

[KHAM08] Masafumi Kusakawa, Harunaga Hiwatari, Tomoyuki Asano, and Seiichi Matsuda. Efficient dynamic broadcast encryption and its extension to authenticated dynamic broadcast encryption. In Matthew K. Franklin, Lucas Chi Kwong Hui, and Duncan S. Wong, editors, *CANS 08*, volume 5339 of *LNCS*, pages 31–48. Springer, Heidelberg, December 2008.

[KS12] Aggelos Kiayias and Katerina Samari. Lower bounds for private broadcast encryption. In *Information Hiding*, volume 7692 of *Lecture Notes in Computer Science*, pages 176–190. Springer, 2012.

[KW20] Sam Kim and David J. Wu. Collusion resistant trace-and-revoke for arbitrary identities from standard assumptions. In *ASIACRYPT 2020, Part II*, LNCS, pages 66–97. Springer, Heidelberg, December 2020.

[KY02] Aggelos Kiayias and Moti Yung. Traitor tracing with constant transmission rate. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 450–465. Springer, Heidelberg, April / May 2002.

[LPQ12] Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Anonymous broadcast encryption: Adaptive security and efficient constructions in the standard model. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 206–224. Springer, Heidelberg, May 2012.

[LPSS14] San Ling, Duong Hieu Phan, Damien Stehlé, and Ron Steinfeld. Hardness of k-LWE and applications in traitor tracing. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 315–334. Springer, Heidelberg, August 2014.

[LSW10] Allison B. Lewko, Amit Sahai, and Brent Waters. Revocation systems with very small private keys. In *2010 IEEE Symposium on Security and Privacy*, pages 273–285. IEEE Computer Society Press, May 2010.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 1219–1234. ACM Press, May 2012.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 735–763. Springer, Heidelberg, May 2016.

[NNL01] Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 41–62. Springer, Heidelberg, August 2001.

[NP01] Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. In Yair Frankel, editor, *FC 2000*, volume 1962 of *LNCS*, pages 1–20. Springer, Heidelberg, February 2001.

[NPP13] Hung Q. Ngo, Duong Hieu Phan, and David Pointcheval. Black-box trace and revoke codes. 67(3):418–448, November 2013.

[NWZ16] Ryo Nishimaki, Daniel Wichs, and Mark Zhandry. Anonymous traitor tracing: How to embed arbitrary information in a key. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 388–419. Springer, Heidelberg, May 2016.

[PPS12] Duong Hieu Phan, David Pointcheval, and Mario Strefler. Decentralized dynamic broadcast encryption. In Ivan Visconti and Roberto De Prisco, editors, *SCN 12*, volume 7485 of *LNCS*, pages 166–183. Springer, Heidelberg, September 2012.

[PPSS12] Duong Hieu Phan, David Pointcheval, Siamak Fayyaz Shahandashti, and Mario Strefler. Adaptive CCA broadcast encryption with constant-size secret keys and ciphertexts. In Willy Susilo, Yi Mu, and Jennifer Seberry, editors, *ACISP 12*, volume 7372 of *LNCS*, pages 308–321. Springer, Heidelberg, July 2012.

[PR08] Ely Porat and Amir Rothschild. Explicit non-adaptive combinatorial group testing schemes. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008, Part I*, volume 5125 of *LNCS*, pages 748–759. Springer, Heidelberg, July 2008.

[PST06] Duong Phan, Reihaneh Safavi-Naini, and Dongvu Tonien. Generic construction of hybrid public key traitor tracing with full-public-traceability. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006, Part II*, volume 4052 of *LNCS*, pages 264–275. Springer, Heidelberg, July 2006.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005.

[Rot06] Ron Roth. *Introduction to Coding Theory*. Cambridge University Press, New York, NY, USA, 2006.

[Tar03] Gábor Tardos. Optimal probabilistic fingerprint codes. In *35th ACM STOC*, pages 116–125. ACM Press, June 2003.

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Heidelberg, August 2009.

[Wee16] Hoeteck Wee. Déjà Q: Encore! Un petit IBE. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 237–258. Springer, Heidelberg, January 2016.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.

[Zha20] Mark Zhandry. New techniques for traitor tracing: Size $N^{1/3}$ and more from pairings. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2020, Part I*, LNCS, pages 652–682. Springer, Heidelberg, August 2020.