PAIRING-BASED CONSTRUCTIONS: EFFICIENT REVOCATION, COMPACT CIPHERTEXTS AND GENERIC TRANSFORMATION

SU LE

School of Physical and Mathematical Sciences

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy
2015

Declaration

This thesis is a presentation of my original research work, carried out by myself, in collaboration with others, whilst enrolled in the Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University (NTU) as a candidate for the degree of Doctor of Philosophy. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgment of collaborative research and discussions.

The work was done under the guidance of my main supervisor Associate Professor Wang Huaxiong from NTU, co-supervisors Dr. Bao Feng and Dr. Vrizlynn L.L.Thing from Agency for Science, Technology and Research (A*STAR), Singapore.

Su Le

27 August 2014

Acknowledgment

The past four years have been thus far the most exciting, challenging, interesting and rewarding part of my life. I am very honored that I have come across many wonderful people who helped me one way or another in my pursuit of the Ph.D degree.

First and foremost, I would like to express my sincere gratitude to my main supervisor Associate Professor Wang Huaxiong for his continuous support of my Ph.D study and research, for his patience, encouragement, enthusiasm, and immense knowledge. His guidance helped me throughout my research and also the process of writing of this thesis. I could not have imagined having a better supervisor for my Ph.D study.

I also would like to sincerely thank my co-supervisors Dr. Bao Feng and Dr. Vrizlynn L.L.Thing for their numerous insightful comments and advices, both from academic and industrial perspective. Their important guidance and encouragement are always great source of energy that inspires me forward.

I am in much debt to Dr. Lim Hoon Wei for the countless many useful and insightful discussions in the cold seminar rooms. His knowledge, passion, persistence on research have greatly influenced me. His brilliant ideas, valuable discussions and constructive feedbacks have guided me through difficult times during my Ph.D study.

I am very grateful to Dr. Chen Jie for his guidance, support and discussions during my Ph.D study. I also would like to thank my cohort and office members for their company and encouragement. It has been wonderful four years to study and work with them, and I wish them excel in their future undertakings.

To my mom and dad, I cannot thank them enough for their unconditional and endless love and support. I wish them always be healthy and happy.

To my wife, Zhang Xuan, I couldn't imagine how difficult it could be for my Ph.D study without her. Her understanding, encouragement, support and love have always been there for me. I wish and congratulate her in advance for the completion of her doctoral degree in the near future.

This Ph.D study is supported by A*STAR Graduate Scholarship, Agency for Science, Technology and Research, Singapore.

Abstract

Over the past decade, the emerging of pairing-based cryptography has attracted not only the attention from academic circle, but also industrial organizations. Starting from key agreement protocols and signature schemes based on elliptic curves, the concept of pairing-based cryptography has been widely used in many cryptographic primitives such as identity-based encryption (IBE), broadcast encryption (BE), attribute-based encryption (ABE), inner-product encryption (IPE), spatial encryption (SE) and so on. These primitives have numerous applications, particularly in the domain of access control, content distribution, mail filtering, data searching, broadcasting, tracing, and biometrics etc. Many of the research in this field are dedicated to improve the scheme efficiency, or to provide generic construction of a primitive. In this thesis we exam the following four aspects from pairing-based cryptography: key update in IBE, ciphertext update in BE, ciphertext compactness in BE and ABE, and last but not least, a generic construction of SE.

Identity-based encryption has been regarded as an attractive alternative to conventional certificate-based public key systems. However, although key revocation is a fundamental requirement to any public key systems, not much work has been done in the identity-based setting. We first continue the study of revocable IBE (RIBE) initiated by Boldyreva, Goyal, and Kumar. Their proposal of a selective secure RIBE scheme, and a subsequent construction by Libert and Vergnaud in a stronger adaptive security model are based on a binary tree approach, such that their key update size is logarithmic in the number of users. We ask the question whether the key update size could be further reduced by using a cryptographic accumulator. We show that, indeed, the key update material can be made constant with some small amount of auxiliary information, through a novel combination of the Lewko and Waters IBE scheme and the Camenisch, Kohlweiss, and Soriente pairing-based dynamic accumulator.

We then turn our focus to a more practical application: encrypted file sharing (EFS) system. EFS systems are widely used in real-life for sharing files, for example a computer-aided design (CAD) drawing shared between architects, consultants and contractors; or sharing scientific data among the researchers from different universities and institutes; or even for a single user who wishes to access her documents from varies devices like PC, mobile and tablet (a file sharing not among people, but devices).

In this work, we first define the criteria for an ideal EFS system, then investigate, analyze and show current approaches have room for improvement. We then further propose a new primitive called updatable broadcast encryption (UBE), which could be used to achieve better efficiency for EFS systems. Through some novel techniques, we provide two concrete UBE constructions based on different broadcast encryption schemes which theoretically outperform existing approaches, and we further prove their security rigorously.

Broadcast encryption and attribute-based encryption can be used for enforcing cryptographic access control at different levels of granularity. Both primitives allow a data owner to encrypt and share information with a set of intended recipients. However, current BE and ABE schemes are largely designed for encryption of single messages. In practice, it is often desirable to encrypt multiple messages simultaneously and share them with potentially different sets of recipients. While this can be done straightforwardly by applying any BE or ABE scheme to each individual message in parallel, the resulting ciphertexts are likely to be large. A fundamental reason for this is that each individual message is typically encrypted using a fresh, unique random value to have an appropriate level of security guarantee. In this work, we investigate the possibility of reusing random values across multiple messages targeting for different recipient sets during encryption, such that significant saving can be gained in the size of the resulting ciphertexts. We propose two new primitives called multi-message broadcast encryption (MM-BE) and multi-message key-policy attribute-based encryption (MM-KP-ABE), and provide two concrete constructions. Our MM-BE scheme reduces the ciphertext size of the existing straightforward approach by almost half; while our MM-KP-ABE scheme cuts down the ciphertext size from quadratic to linear complexity.

Last but not least, we investigate a variant of spatial encryption (SE) we call ciphertext-policy SE (CP-SE), which combines the properties of SE and those from ciphertext-policy attribute-based encryption (CP-ABE). The resulting primitive supports non-monotone access structure. In CP-SE, the decryptability of a ciphertext depends on whether the required attribute vectors are in the same affine space that corresponding to the decryption key. This primitive gives rise to many new applications, for example, SE supporting negation, hierarchical ABE (HABE) and forward-secure ABE. In this part, we present techniques for generic construction of CP-SE from ciphertext-policy inner product encryp-

tion (CP-IPE). Our techniques are property-preserving: if the CP-IPE scheme, from which our CP-SE scheme is derived, is fully secure, then so is the resulting CP-SE scheme. Moreover, interestingly, we show that it is possible to perform transformation from the opposite direction, which is to construct a CP-IPE scheme from a given a CP-SE scheme.

Contents

List of Abbreviations

ABE	Attribute-Based Encryption
BE	Broadcast Encryption
CAD	Computer-Aided Design
CCA	Chosen Ciphertext Attack
CPA	Chosen Plaintext Attack
CP-ABE	Ciphertext-Policy Attribute-Based Encryption
CP-IPE	Ciphertext-Policy Inner-Product Encryption
CP-SE	Ciphertext-Policy Spatial Encryption
CRL	Certificate Revocation List
CP-ABE	Ciphertext-Policy Attribute-Based Encryption
DBDH	Decisional Bilinear Diffie-Hellman
DBDHE	Decisional Bilinear Diffie-Hellman Exponent
EFS	Encrypted File Sharing
EHR	Electronic Health Record
FE	Functional Encryption
FIBE	Fuzzy Identity-Based Encryption
GBDHE	Generalised Bilinear Diffie-Hellman Exponent
HABE	Hierarchical Attribute-Based Encryption
HIBE	Hierarchical Identity-Based Encryption
HIBS	Hierarchical Identity-Based Signature
HIPE	Hierarchical Inner-Product Encryption
IBBE	Identity-Based Broadcast Encryption
IBE	Identity-Based Encryption
IBS	Identity-Based Signature
IPE	Inner-Product Encryption

KEM	Key Encapsulation Mechanism
KP-ABE	Key-Policy Attribute-Based Encryption
LSSS	Linear Secret Sharing Scheme
MM-BE	Multi-Message Broadcast Encryption
MM-KP-ABE	Multi-Message Key-Policy Attribute-Based Encryption
MRES	Multi Recipient Encryption Scheme
OBDHE	Oracle Bilinear Diffie-Hellman Exponent
OCSP	Online Certificate Status Protocol
PKG	Private Key Genetator
PKI	Public-Key Infrastructure
PK-PRE	Public-Key Proxy Re-Encryption
PKC	Public-Key Cryptography
PKE	Public-Key Encryption
PPT	Probabilistic Polynomial Time
PRE	Proxy Re-Encryption
RABE	Revocable Attribute-Based Encryption
RIBE	Revocable Identity-Based Encryption
RPE	Revocable Predicate Encryption
RS-ABE	Revocable Storage Attribute-Based Encryption
SE	Spatial Encryption
SKE	Symmetric-Key Encryption
SK-PRE	Symmetric-Key Proxy Re-Encryption
UBE	Updatable Broadcast Encryption

List of Tables

Chapter 1

Introduction

Contents

This chapter gives an overview of the thesis. We provide the motivation for our research and describe the contributions of this thesis. In this chapter, we also present the overall structure of the thesis.

1.1 Motivation

Early in the 2000's, the construction of a feasible and secure identity-based encryption (IBE) scheme have ignited the enthusiasm of cryptographic schemes based on pairings on elliptic curves. Pioneered by the novel work of Sakai et al. [105] on pairing-based key agreement protocols and signature schemes, and subsequent three-party key agreement protocol by Joux [78], Boneh and Franklin [27] then presented the first practical and secure IBE scheme based on the Weil pairings. These three key contributions have stimulated the development of a wide range of pairing-based cryptographic schemes and protocols. Subsequently, there have been proposals on identity-based signature (IBS) schemes [38, 70], hierarchical identity-based encryption (HIBE) and signature (HIBS) schemes [60, 75]. Moving beyond, pairing-based cryptography sees its flourish in more new primitives, such as broadcast encryption (BE)

[28] and its identity-based version IBBE [51], attribute-based encryption [64, 22], inner-product encryption (IPE) [95, 84, 82], spatial encryption (SE) [68]. These new primitives have later been categorized and unified as functional encryption [32].

While the research of pairing-based primitives focuses on improving the scheme efficiency and security level, it is also important to empower them with new functionalities. In the case of private key compromise, or when a user left the system, the problem of user revocation is ubiquitous for every cryptosystem. It is always attempting to keep the key update information as small as possible, and the private key update operation occurred at user side are minimized or even not required. As one of the most studied pairing-based systems, the problem of revocation in IBE, although unquestionably important, attracts less attention. As one of the major tools for file sharing, BE itself has certain flavor of user revocation embedded in the primitive: the encryptor explicitly specifies users with or without access to the encrypted file during encryption. However, in this one-to-many message encryption mechanism, it is meaningful to minimize the impact on other legitimate users when a single user has been compromised.

The expressiveness of certain functional encryption schemes, such as broadcast encryption and attribute-based encryption, makes them very attractive to be deployed into real-life applications for the purpose of file sharing and access control. They are even more appealing than the traditional public key cryptography primitives such as RSA: the user public keys are no longer randomly generated numbers, but meaningful strings such as email addresses, personal credentials, etc. However, this expressiveness comes at some trade-offs. One of such trade-offs occurs at the size of the ciphertext: as the primitive becomes complicated, the ciphertext tends to be large. For example, the ciphertexts for a single message of most ABE schemes are at linear size w.r.t the number of attributes in the system. This downside quickly becomes the bottleneck when multiple encrypted files are stored at a pay-per-storage cloud-based file sharing system. Reduction of total ciphertext size in multiple message encryption is undoubtedly one of the most interesting areas worth further investigation.

It is also important to study the relation between varies primitives under the functional encryption category. Although, at a high level, all these primitives allow a legitimate private key to decrypt the ciphertext successfully as long as a certain *function* is satisfied, the underlying different mathematical structures make certain desirable cryptographic properties harder to achieve in one primitive than an-

other. For example, some hierarchal inner-product encryption (HIPE) schemes are fully secure (the highest level of security) and attribute-hiding (anonymous) but do not have short ciphertexts. On the other hand, certain spatial encryption (SE) schemes have short ciphertexts but are non-anonymous, and is secure only in a selective security model (a weaker model). It is thus natural to investigate whether generic transformation between primitives are possible, such that the desired properties of one primitive A could be preserved after the transformation to primitive B, while the original properties of B are kept intact.

1.2 Contributions

Motivated by the above reasons, in this thesis we exam the following four aspects from pairing-based cryptography area: key update in IBE, ciphertext update in BE, ciphertext compactness in BE and ABE, and last but not least, a generic construction of SE. The contributions of this thesis could be summarized as follows.

1. Firstly, we present an efficient RIBE scheme based on Lewko and Waters' composite order IBE scheme [86]. Combined with Camenisch et al.'s accumulator [34] in a particular way, our scheme achieves almost constant size key update with only small amount of auxiliary information. The Waters' dual system encryption methodology is adopted in the security proof, where we consider two types of adversary and achieve adaptive security for both types. We further show that our scheme could be extended to achieve forward-security, in the sense that compromization of a decryption key at one time would not leak any useful information about other decryption keys for other times. Moreover, we describe how our technique can be used to construct a revocable ABE (RABE) scheme.

2. Secondly, the focus are turned to scheme design of more practical real-life applications. By studying the file sharing system, we propose a new primitive called updatable broadcast encryption (UBE) which allows an owner to share files with potentially many users through a cloud server provider. After investigating the current solutions and explaining the challenges we are facing, we provide two constructions based on different types of BE schemes, namely the exclusive-BE

scheme by Lewko et al. [85], and the inclusive-BE scheme by Boneh et al. [28]. The scheme based on exclusive-BE enjoys theoretically advantages over all current solutions, where the file header size and revocation cost occurred at server, owner and recipients are constant. The second scheme based on inclusive-BE achieves better practical performances.

3. Thirdly, we study the problem of how to minimize the ciphertext storage requirement for encryption of multiple messages under different access policies. This is particularly useful when cryptographic access control needs to be enforced on encrypted data, which in turn are stored on the cloud. We are particularly interested in two primitives, namely broadcast encryption and attribute-based encryption. We first propose the concept of multi-message broadcast encryption (MM-BE) and multi-message key-policy attribute-based encryption (MM-KP-ABE), and their respective security models. Moreover, each new primitive is provided with a concrete construction and rigorous security proofs. We show that significant ciphertext saving is achieved with reasonable trade-offs compared to a naive approach.

4. Lastly, our works is extended to the study of spatial encryption (SE). Particularly, we inject some flavor of cryptographic access control into SE by enriching it with a non-monotone access structure [97] to ciphertexts. In this way, a user's secret key is associated with an access structure over some attributes, reflecting the access policy ascribed to the user. Our access-structure endowed SE is called ciphertext-policy SE (CP-SE), analogous to ciphertext-policy ABE (CP-ABE) [22, 117]. As with the case of CP-ABE, our CP-SE is capable to handle any access structure that can be represented by a boolean formula involving `AND`, `OR`, `NOT`, and threshold operations. We present techniques for generic construction of CP-SE from ciphertext-policy inner product encryption (CP-IPE). Our techniques are property-preserving in the sense that if the CP-IPE scheme, from which our CP-SE scheme is derived, is fully secure, then so is the resulting CP-SE scheme. Moreover, interestingly, we show that it is possible to perform transformation from the opposite direction, that is how to construct a CP-IPE scheme from a given CP-SE scheme.

Some results in this thesis have been published by Springer-Verlag in Lecture Notes in Computer Science (LNCS) series [40, 110]. The Ph.D candidate of this thesis takes the leading role in the first

three works presented in this thesis (Chapter 3, 4, 5), and contributes part of the work in the fourth work (Chapter 6).

1.3 Organization of Thesis

The remainder of this thesis is organized as follows:

Preliminaries. In Chapter 2, we first provide the background information on pairing, its mathematical form, and the various hard assumptions we used during our constructions given out in this thesis. Subsequently we define those main cryptographic primitives we used throughout the thesis and their respective security models, namely IBE, BE and ABE, which are crucial for the understanding of the whole thesis. The definition of SE and IPE are separately given in Chapter 6 as they are independent from previous topics and for the ease of reading of Chapter 6.

Revocable IBE Systems with Almost Constant-size Key Update. In Chapter 3, we present our new construction of RIBE scheme based on composite order bilinear groups. It achieves almost constant size key update with only small size of auxiliary information. We start from providing the motivation for our RIBE work, followed by our analysis of the approach of existing schemes, then explain the difficulties we are facing, and the technical approaches under taken. In the subsequent section we first define the RIBE primitive and its security model, followed by the concrete construction and security proof. we further discuss on scheme extensions: supporting forward secrecy and adaptation to revocable attribute-based encryption (RABE). We provide some concluding remarks on this study at the end of the chapter.

Near-Ideal Encrypted File Sharing At Scale Through Updatable Broadcast Encryption. In Chapter 4, we first define a set of properties for an ideal encrypted file sharing (EFS) system, then investigate, analyze and show potential improvement in current approaches. We then further propose a new primitive called updatable broadcast encryption (UBE), which could be used to achieve better efficiency for EFS systems. Through some novel techniques, we provide two concrete UBE constructions based on different broadcast encryption schemes which theoretically outperform existing approaches, and we further prove their security rigorously.

Multi-Message BE and Multi-Message ABE with Compact Ciphertexts. In Chapter 5, we present

our new primitives MM-BE and MM-ABE, and their respective concrete constructions based on BE and ABE schemes. It starts from the motivation behind this work, the importance of randomness reuse to achieve the goal, the difficulties encountered and the approaches we have take. Subsequently we formally define our new primitives and their security models. In the following two sections, we present the two concrete constructions: one from BE, and the other from KP-ABE, respectively. We also provide efficiency comparison of our approaches with current solutions, and summarize this work at the end of this Chapter.

Spatial Encryption Supporting Non-Monotone Access Structure. In this Chapter, we study the generic transformation between CP-SE and CP-IPE. As usual, we first provide the motivation for this study, the difficulties we encountered and our approach to solve them. Subsequently we provide the definitions and security models of CP-SE and CP-IPE, followed by the generic construction of CP-SE from CP-IPE first, then the generic construction of CP-IPE from CP-SE. Some applications of CP-SE are given before we discuss and compare our scheme with existing schemes.

Conclusion. In the final chapter of this thesis, Chapter 7, we give concluding remarks about our proposals in Chapters 3, 4, 5 and 6, including the problems that we have studied, the importance of these problems, and a summary of our research findings. We also provide some suggestions for future work related to our proposals.

Chapter 2

Preliminaries

Contents

This chapter provides the preliminaries required for this thesis. We first provide a brief background on pairings and the underlying complexity assumptions used for security proofs, followed by definitions of primitives such as revocable identity-based encryption, broadcast encryption and attribute-based encryption.

2.1 Background on Pairings

Prime order bilinear groups are defined by a group generator $\mathcal{G}$ which takes as input a security parameter λ and outputs the description of a bilinear map G. This includes (p, G, G_T, e), where p is a prime number, G and G_T are cyclic groups of order p, and $e : G \times G \rightarrow G_T$ is a bilinear map such that:

- (Bilinear) $\forall g, h \in G, a, b, \in \mathbb{Z}_p, e(g^a, h^b) = e(g, h)^{ab}$,
- (Non-degenerate) $\exists g \in G$ such that $e(g, g)$ has order p in G_T.

In addition to the above properties, we also require that the group operations in G and G_T, together with the bilinear map e, are polynomial-time computable with respect to the security parameter λ. We also assume that the group descriptions of G and G_T include the group generators.

On the other hand, composite order bilinear groups are different from the prime order groups in the sense that, now the group order is $N = p_1p_2p_3$, where p_1, p_2, p_3 are three distinct primes [29]. Composite order groups still maintain the "bilinear" and "non-degenerate" properties as prime order groups, as well as the computational efficiency requirement. For ease of exposition, we let $G_{p_1}, G_{p_2}, G_{p_3}$ denote the subgroups of order p_1, p_2 and p_3 in G respectively. We also note the orthogonality property of the composite order bilinear map: that is, $e(h_i, h_j)$ is the identity element in G_T whenever $h_i \in G_{p_i}$ and $h_j \in G_{p_j}$ for $i \neq j$. This property of the three subgroups will be a principal tool in some of our constructions and their security proofs.

2.2 Complexity Assumptions

The security of a scheme is always based on the intractability of certain mathematical problems. The underlying hard problems, on which the security of the cryptographic schemes provably rely, are called complexity assumptions. It is desirable that the underlying assumptions in the cryptographic schemes are well-established or standard. We now state the complexity assumptions we used to prove the security of our schemes. The first three assumptions are the same those used in [86]. They are static (not dependent on the number of queries made by an adversary) and can be proved using the theorem introduced by Katz, Sakai and Waters [82]. Moreover, they are based on composite order bilinear groups. The remaining assumptions are all based on prime order bilinear groups. Assumption 4, Bilinear Diffie-Hellman Exponent assumption is used in [28], and as an extension of assumption 4, the subsequent assumption 5, Oracle Bilinear Diffie Hellman Exponent (OBDHE) assumption, is used in [98] to prove the security of a broadcast encryption scheme. Assumption 6 and assumption 7 are used in, for example [27, 85], as the respective hard problems.

In the subsequent assumption description and throughout the whole thesis, $a \xleftarrow{R} A$ denotes that a is randomly sampled from A. Adv stands for the adversary's advantage, $Pr[\cdot]$ represent the probability of an event. A negligible function $f(\cdot)$ is a function such that for every positive integer c there exists an

integer λ_0 such that for all $\lambda > \lambda_0$, we have $|f(\lambda)| < \frac{1}{\lambda^c}$.

In the assumptions 1-3 below, we let $G_{p_i p_j}$ denote the subgroup of order $p_i p_j$ in composite order group G.

Assumption 1. ***(Subgroup decisional problem for 3 primes)*** *Given a group generator $\mathcal{G}$, we define the following distribution:*

$$\mathbb{G} = (N = p_1 p_2 p_3,\ G,\ G_T,\ e) \xleftarrow{R} \mathcal{G},$$
$$g \xleftarrow{R} G_{p_1},\ X_3 \xleftarrow{R} G_{p_3},$$
$$D = (\mathbb{G},\ g,\ X_3),$$
$$T_1 \xleftarrow{R} G_{p_1 p_2},\ T_2 \xleftarrow{R} G_{p_1}.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 1 to be:

$$Adv1_{\mathcal{G},\mathcal{A}}(\lambda) := |Pr[\mathcal{A}(D,\ T_1) = 1] - Pr[\mathcal{A}(D,\ T_2) = 1]|.$$

Definition 1. *We say that $\mathcal{G}$ satisfies Assumption 1 if $Adv1_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

Assumption 2. *Given a group generator $\mathcal{G}$, we define the following distribution:*

$$\mathbb{G} = (N = p_1 p_2 p_3,\ G,\ G_T,\ e) \xleftarrow{R} \mathcal{G},$$
$$g,\ X_1 \xleftarrow{R} G_{p_1},\ X_2,\ Y_2 \xleftarrow{R} G_{p_2},\ X_3,\ Y_3 \xleftarrow{R} G_{p_3},$$
$$D = (\mathbb{G},\ g,\ X_1 X_2,\ X_3,\ Y_2 Y_3),$$
$$T_1 \xleftarrow{R} G,\ T_2 \xleftarrow{R} G_{p_1 p_3}.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 2 to be:

$$Adv2_{\mathcal{G},\mathcal{A}}(\lambda) := |Pr[\mathcal{A}(D,\ T_1) = 1] - Pr[\mathcal{A}(D,\ T_2) = 1]|.$$

Definition 2. *We say that $\mathcal{G}$ satisfies Assumption 2 if $Adv2_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

Assumption 3. *Given a group generator $\mathcal{G}$, we define the following distribution:*

$$\mathbb{G} = (N = p_1 p_2 p_3,\ G,\ G_T,\ e) \xleftarrow{R} \mathcal{G},$$
$$\alpha,\ s \xleftarrow{R} \mathbb{Z}_N,\ g \xleftarrow{R} G_{p_1},\ X_2,\ Y_2,\ Z_2 \xleftarrow{R} G_{p_2},\ X_3 \xleftarrow{R} G_{p_3},$$
$$D = (\mathbb{G},\ g,\ g^{\alpha} X_2,\ X_3,\ g^{s} Y_2,\ Z_2),$$
$$T_1 = e(g,g)^{\alpha s},\ T_2 \xleftarrow{R} G_T.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 3 to be:

$$Adv3_{\mathcal{G},\mathcal{A}}(\lambda) := |Pr[\mathcal{A}(D,\ T_1) = 1] - Pr[\mathcal{A}(D,\ T_2) = 1]|.$$

Definition 3. *We say that $\mathcal{G}$ satisfies Assumption 3 if $Adv3_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

Assumption 4. ***(Bilinear Diffie-Hellman Exponent)*** *Given a group generator $\mathcal{G}$, we define the following distribution:*

$$\mathbb{G} = (G,\ G_T,\ e) \xleftarrow{R} \mathcal{G},$$
$$\alpha \xleftarrow{R} \mathbb{Z}_N,\ g,\ f \xleftarrow{R} G,$$
$$D = (\mathbb{G},\ f,\ g,\ g^{\alpha},\ g^{\alpha^2},\ \ldots,\ g^{\alpha^{\ell}},\ g^{\alpha^{\ell+2}},\ \ldots, g^{\alpha^{2\ell}})$$
$$T_1 = e(g^{\alpha^{\ell+1}},\ f),\ T_2 \xleftarrow{R} G_T.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 4 to be:

$$Adv4_{\mathcal{G},\mathcal{A}}(\lambda) := |Pr[\mathcal{A}(D,\ T_1) = 1] - Pr[\mathcal{A}(D,\ T_2) = 1]|.$$

Definition 4. *We say that $\mathcal{G}$ satisfies Assumption 4 if $Adv4_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial-time algorithm $\mathcal{A}$.*

We now define the Diffie-Hellman computation oracle required for the OBDHE assumption.

Definition 5. *The Diffie Hellman computation oracle $\mathcal{O}^{DH}_{g,e}$ takes as inputs $u, v \in \mathbb{G}$ and outputs $w \in \mathbb{G}$ such that $e(u,v) = e(g,w)$.*

We let prime p_1 be the group order of G and define the OBDHE assumption as follow:

Assumption 5. ***(Oracle Bilinear Diffie-Hellman Exponent)*** *Given a group generator $\mathcal{G}$, we define the following distribution:*

$$\mathbb{G} = (G,\ G_T,\ e) \xleftarrow{R} \mathcal{G},$$
$$\alpha \xleftarrow{R} \mathbb{Z}_N,\ g,\ f \xleftarrow{R} G,$$
$$D = (\mathbb{G},\ f,\ g,\ g^{\alpha},\ g^{\alpha^2},\ \ldots,\ g^{\alpha^{\ell}},\ g^{\alpha^{\ell+2}},\ \ldots, g^{\alpha^{2\ell}})$$
$$T_1 = e(g^{\alpha^{\ell+1}},\ f),\ T_2 \xleftarrow{R} G_T.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 5 to be:

$$Adv5_{\mathcal{G},\mathcal{A}}(\lambda) := |Pr[\mathcal{A}(D,\ T_1) = 1] - Pr[\mathcal{A}(D,\ T_2) = 1]|$$

given that $\mathcal{A}$ has access to the $\mathcal{O}^{DH}_{g,e}$ oracle.

Definition 6. *We say that $\mathcal{G}$ satisfies Assumption 5 if $Adv5_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

Assumption 6. ***(Decisional Bilinear Diffie-Hellman)*** *Given a group generator $\mathcal{G}$, we define the following distribution:*

$$\mathbb{G} = (G,\ G_T,\ e) \xleftarrow{R} \mathcal{G},$$
$$a,\ b,\ c \xleftarrow{R} \mathbb{Z}_N,\ g \xleftarrow{R} G,$$
$$D = (\mathbb{G},\ g,\ g^a,\ g^b,\ g^c),$$
$$T_1 = e(g,g)^{abc},\ T_2 \xleftarrow{R} G_T.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 6 to be:

$$Adv6_{\mathcal{G},\mathcal{A}}(\lambda) := |Pr[\mathcal{A}(D,\ T_1) = 1] - Pr[\mathcal{A}(D,\ T_2) = 1]|.$$

Definition 7. *We say that $\mathcal{G}$ satisfies Assumption 6 if $Adv6_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial-time algorithm $\mathcal{A}$.*

Assumption 7. ***(q-Decisional Multi-Exponent Bilinear Diffie-Hellman Assumption)*** *Given a group generator $\mathcal{G}$, we define the following distribution:*

$$\mathbb{G} = (G,\ G_T,\ e) \xleftarrow{R} \mathcal{G},$$
$$s,\ \alpha,\ a_1, \ldots, a_q \xleftarrow{R} \mathbb{Z}_N,\ g \xleftarrow{R} G,$$
$$D_1 = (\mathbb{G},\ g,\ g^s,\ e(g,g)^\alpha),$$
$$D_2 = (g^{a_i},\ g^{a_i s},\ g^{a_i a_j},\ g^{\alpha/a_i^2}) \qquad \forall 1 \leq i,j \leq q,$$
$$D_3 = (g^{a_i a_j s},\ g^{\alpha a_j/a_i^2},\ g^{\alpha a_i a_j/a_k^2},\ g^{\alpha a_i^2/a_j^2}) \qquad \forall 1 \leq i,j,k \leq q, i \neq j,$$
$$T_1 = e(g,g)^{\alpha s},\ T_2 \xleftarrow{R} G_T.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 7 to be:

$$Adv7_{\mathcal{G},\mathcal{A}}(\lambda) := |Pr[\mathcal{A}(D_1, D_2, D_3, T_1) = 1] - Pr[\mathcal{A}(D_1, D_2, D_3, T_2) = 1]|.$$

Definition 8. *We say that $\mathcal{G}$ satisfies Assumption 7 if $Adv7_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial-time algorithm $\mathcal{A}$.*

2.3 Revocable Identity-Based Encryption

Revocation in IBE was first studied in Boneh and Franklin's IBE scheme [27]. It was later formalized by Boldyreva et al [24]. We adapt the same definition of RIBE from [24].

Let $\mathcal{M}$ denote a message space, $\mathcal{I}$ denote an identity space, and $\mathcal{T}$ denote a time space. Assume that the sizes of $\mathcal{M}, \mathcal{I}, \mathcal{T}$ are all polynomial in the security parameter. Each algorithm within RIBE is run by either one of three types of parties—key authority, sender or receiver. The key authority maintains a revocation list RL and state ST. An algorithm is called stateful if it updates either RL or ST. We treat time as discrete as opposed to continuous.

An identity-based encryption scheme with efficient revocation or simply revocable IBE (RIBE) scheme has seven PPT algorithms defined as follows:

- $\mathsf{Setup}(1^k, n) \rightarrow (\mathsf{PP}, \mathsf{MK}, \mathsf{RL}, \mathsf{ST})$. The setup algorithm takes as input a security parameter k and a maximal number of users n. It outputs a public parameters PP, a master key MK, a revocation list RL (initially empty), and a state ST. RL abd ST are public information. (This is run by the key authority.)

- $\mathsf{PriKeyGen}(\mathsf{PP}, \mathsf{MK}, id, \mathsf{ST}) \to (\mathsf{SK}_{id}, \mathsf{ST})$. The private key generation algorithm takes as input the public parameters PP, the master key MK, an identity $id \in \mathcal{I}$, and the state ST. It outputs a private key SK_{id} and an updated state ST. (This is stateful and run by the key authority.)

- $\mathsf{KeyUpd}(\mathsf{PP}, \mathsf{MK}, t, \mathsf{RL}, \mathsf{ST}) \to \mathsf{KU}_t$. The key update algorithm takes as input the public parameters PP, the master key MK, a key update time $t \in \mathcal{T}$, the revocation list RL, and the state ST. It outputs a publicly available key update KU_t. (This is run by the key authority.)

- $\mathsf{DecKeyGen}(\mathsf{SK}_{id}, \mathsf{KU}_t) \to \mathsf{DK}_{id,t}$. The decryption key generation takes as input a private key SK_{id} and key update KU_t. It outputs a decryption key $\mathsf{DK}_{id,t}$ or a special symbol $\perp$ indicating that id was revoked. The $\mathsf{DK}_{id,t}$ should be kept secret by the individual receiver. (This is run by the receiver.)

- $\mathsf{Enc}(\mathsf{PP}, id, t, M) \to \mathsf{CT}_{id,t}$. The encryption algorithm takes as input the public parameters PP, an identity id, an encryption time t, and a message $M \in \mathcal{M}$. It outputs a ciphertext $\mathsf{CT}_{id,t}$. (This is run by the sender. For simplicity and without loss of generality, we assume that id, t are efficiently computable from $\mathsf{CT}_{id,t}$.)

- $\mathsf{Dec}(\mathsf{PP}, \mathsf{DK}_{id,t}, \mathsf{CT}_{id,t}) \to M$. The decryption algorithm takes as input the public parameters PP, a decryption key $\mathsf{DK}_{id,t}$, and a ciphertext $\mathsf{CT}_{id,t}$. It outputs a message M. (This is deterministic and run by the receiver.)

- $\mathsf{KeyRev}(id, t, \mathsf{RL}, \mathsf{ST}) \to \mathsf{RL}$. The key revocation algorithm takes as input an identity to be revoked id, a revocation time t, the revocation list RL, and the state ST. It outputs an updated revocation list RL. (This is stateful and run by the key authority.)

The consistency condition requires that for all $k \in \mathbb{N}$ and polynomials (in k) n, all PP and MK output by setup algorithm Setup, all $M \in \mathcal{M}, id \in \mathcal{I}, t \in \mathcal{T}$ and all possible valid states ST and revocation lists RL, we then have

$$\mathsf{Dec}(\mathsf{PP}, \mathsf{DK}_{id,t}, \mathsf{CT}_{id,t}) = M$$

with probability 1 if identity id was not revoked before or at time t.

Next, we define the security of RIBE in the form of a security game played between an adversary $\mathcal{A}$ and a challenger.

- Setup: It is run by the challenger to generate some public parameters PP, a master key MK, a revocation list RL (initially empty), and a state ST. Then $\mathsf{PP}, \mathsf{RL}, \mathsf{ST}$ are given to $\mathcal{A}$.
- Query: $\mathcal{A}$ may adaptively make a polynomial number of queries of the following oracles (which share state information):
  - The private key generation oracle $\mathsf{PriKeyGen}(\cdot)$ takes as input an identity id and runs $\mathsf{PriKeyGen}(\mathsf{PP}, \mathsf{MK}, id, \mathsf{ST})$ to return a private key SK_{id}.
  - The key update generation oracle $\mathsf{KeyUpd}(\cdot)$ takes as input time t and runs $\mathsf{KeyUpd}(\mathsf{PP}, \mathsf{MK}, t, \mathsf{RL}, \mathsf{ST})$ to return key update KU_t.
  - The revocation oracle $\mathsf{KeyRev}(\cdot, \cdot)$ takes as input an identity id and time t, and runs $\mathsf{KeyRev}(id, t, \mathsf{RL}, \mathsf{ST})$ to update RL.
- Challenge: $\mathcal{A}$ outputs the target ID-time pair (id^*, t^*) and two messages M_0, M_1. The challenger flips a random bit d and returns the output of $\mathsf{Enc}(\mathsf{PP}, id^*, t^*, M_d)$ to $\mathcal{A}$. After that, the adversary may continue to make queries to the oracles as with in the Query phase.
- Guess: At the end of the game, the adversary outputs a bit d', and succeeds if $d' = d$.

The following restrictions must always hold:

1. $M_0, M_1 \in \mathcal{M}$ and $|M_0| = |M_1|$.
2. $\mathsf{KeyUpd}(\cdot)$ and $\mathsf{KeyRev}(\cdot, \cdot)$ can be queried on a time which is greater than or equal to all the previously queried times, i.e., the adversary is allowed to query only in non-decreasing order of time. Also, the oracle $\mathsf{KeyRev}(\cdot, \cdot)$ cannot be queried at time t if $\mathsf{KeyUpd}(\cdot)$ was queried on t.
3. If $\mathsf{PriKeyGen}(\cdot)$ was queried on identity id^*, then $\mathsf{KeyRev}(\cdot, \cdot)$ must be queried on (id^*, t) for some $t \leq t^*$.

If the adversary's output d' equals to d, we set $\mathsf{return} = 1$, otherwise $\mathsf{return} = 0$. We define the adversary's advantage as

$$Adv_{\mathcal{A}}^{RIBE}(\lambda) := |Pr[\mathsf{return} = 1] - \frac{1}{2}|.$$

An RIBE scheme is adaptive-ID secure if for all PPT adversaries $\mathcal{A}$ the function $Adv_{\mathcal{A}}^{RIBE}(\lambda)$ is negligible.

2.4 Broadcast Encryption

Broadcast encryption was introduced by Fiat and Naor [56], it allows a message to be encrypted and sent to potentially many recipients by performing the encryption algorithm only once. There are two types of BE schemes, namely inclusive-BE and exclusive-BE. The former involves a specific legitimate recipient set during encryption, while the later encrypts the message under a set of revoked user. In exclusive-BE scheme, anyone with a legitimate private key outside the revocation set could decrypt the message successfully.

We first define an inclusive-BE scheme. It is made up of three randomized algorithms [28]:

- $\mathsf{Setup}(1^k, n) \rightarrow (\mathsf{PP}, d_i)$. The setup algorithm takes as input the security parameter k and the number of receivers n. It outputs n private keys[1] $d_1, \ldots, d_n$ and the public parameter PP.

- $\mathsf{Enc}(S, \mathsf{PP}) \rightarrow (\mathit{Hdr}, K)$. The encryption algorithm takes as input a receiver set S and the public parameter PP. It outputs a pair (Hdr, K) where *Hdr* is called the header and $K \in \mathcal{K}$ is a message encryption key. *Hdr* is often referred as the broadcast ciphertext, and K should be kept secret.

  Let M be a message to be broadcast to the set S and let C_M be the encryption of M under the symmetric key K. The broadcast to users in S consists of (S, Hdr, C_M). The pair (S, Hdr) is often called the full header and C_M is often called the broadcast body.

- $\mathsf{Dec}(S, i, d_i, \mathit{Hdr}, \mathsf{PP}) \rightarrow M$. The decryption algorithm takes as input a subset $S \subseteq \{1, \ldots, n\}$, a user id $i \in \{1, \ldots, n\}$ and the private key d_i for user i, a header *Hdr*, and the public parameter

[1] Some papers define a separate private key generation algorithm.

PP. If $i \in S$, then the algorithm outputs the message encryption key $K \in \mathcal{K}$. The key K can then be used to decrypt the broadcast body C_M and obtain the message M.

Exclusive-BE is defined in a similar way except the set S contains the revoked users who has no access to the message M.

We now give the full security definition for inclusive-BE systems. This is described by a security game between a challenger and an adversary. The game proceeds as follows:

- Setup. The challenger runs setup algorithm to obtain the public parameter PP. It gives PP to the adversary.
- Query Phase 1. The adversary adaptively issues private key queries of user index i. The challenger generates d_i and passes it to the adversary.
- Challenge. The adversary outputs a target receiver set S^* with the restriction that all previously queried user indices i do not appear in S^*. The challenger runs the encryption algorithm under (S^*, PP) to obtain (Hdr^*, K). Next, the challenger picks a random $\beta \in \{0, 1\}$. It sets $K_\beta = K$ and picks a random $K_{1-\beta}$. It then gives $(\mathit{Hdr}^*, K_0, K_1)$ to the adversary.
- Query Phase 2. The adversary continues to adaptively issue private key queries, with the restriction that the queried user indices could not appear in S^*. The challenger responses as in Phase 1.
- Challenge. The adversary outputs its guess β' for β.

The advantage of the adversary is defined as $Adv_{\mathcal{A}}(\lambda) := |Pr[\beta = \beta'] - \frac{1}{2}|$. We say a key-policy attribute-based encryption system is fully secure if all polynomial time attackers have at most a negligible advantage in this security game.

The security game for exclusive-BE is defined in a similar way.

2.5 Attribute-Based Encryption

Attribute-based encryption uses the concept of access structure and linear secret-sharing schemes (LSSS) from [18]. Before we provide the definition of ABE, we define the concept of access structure and LSSS

first.

Definition 9. *(Access Structure) Let $\{P_1, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,\ldots,P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, \ldots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1,\ldots,P_n\}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the* authorized sets*, and the sets not in $\mathbb{A}$ are called the* unauthorized sets*.*

Definition 10. *(Linear Secret-Sharing Schemes (LSSS)) A secret sharing scheme Π over a set of parties $\mathcal{P}$ is called* linear *(over $\mathbb{Z}_p$) if*

1. *The shares for each party form a vector over $\mathbb{Z}_p$.*

2. *There exists a matrix A called the share-generating matrix for Π. The matrix A has ℓ rows and n columns. For all $i = 1, \ldots, \ell$, the i^{th} row of A is labeled by a party $\rho(i)$ (ρ is a function from $\{1, \ldots, \ell\}$ to $\mathcal{P}$). When we consider the column vector $v = (s, r_2, \ldots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared and $r_2, \ldots, r_n \in \mathbb{Z}_p$ are randomly chosen, then Av is the vector of ℓ shares of the secret s according to Π. The share $(Av)_i$ belongs to party $\rho(i)$.*

The above defined scheme has the *linear reconstruction* property. Suppose that Π is an LSSS for access structure $\mathbb{A}$. Let S denote an authorized set, and define $I \subseteq \{1, \ldots, \ell\}$ as $I = \{i | \rho(i) \in S\}$. Then the vector $(1, 0, \ldots, 0)$ is in the span of rows of A indexed by I, and there exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that, for any valid shares $\{\lambda_i\}$ of a secret s according to Π, we have $\sum_{i \in I} \omega_i \lambda_i = s$. These constants $\{\omega_i\}$ can be found in polynomial-time in the size of the share-generating matrix A [18]. For unauthorized sets, no such constants $\{\omega_i\}$ exists.

Attribute-based encryption was introduced by Sahai and Waters [104]. Subsequently Goyal, Pandey, Sahai, and Waters [64] formulated two complimentary forms of ABE: ciphertext-policy attribute-based encryption (CP-ABE) and key-policy attribute-based encryption (KP-ABE). In a CP-ABE system, keys are associated with sets of attributes and ciphertexts are associated with access policies. For example, a Ph.D student in the math department may have his private key associated with attribute set {Ph.D, math, male}. He won't be able to decrypt a ciphertext with access policy {professor `AND` math}. However he could decrypt a ciphertext with access policy {(math `AND` Ph.D) `OR` professor}. In a KP-ABE system,

the situation is reversed: keys are associated with access policies and ciphertexts are associated with sets of attributes.

Next, we define a key-policy attribute-based encryption system. A KP-ABE scheme consists of four algorithms [84]:

- $\mathsf{Setup}(1^k, U) \rightarrow (\mathsf{PP}, \mathsf{MK})$. The setup algorithm takes as input the security parameter k and the attribute universe description U. It outputs the public parameters PP and a master secret key MK.
- $\mathsf{Enc}(M, \mathsf{PP}, S) \rightarrow \mathsf{CT}$. The encryption algorithm takes as input a message M, the public parameters, and a set S of attributes. It outputs a ciphertext CT.
- $\mathsf{KeyGen}(\mathbb{A}, \mathsf{MK}, \mathsf{PP}) \rightarrow \mathsf{SK}$. The private key generation algorithm takes as input an access structure $\mathbb{A}$, the master secret key MK, and the public parameters PP. It outputs a secret key SK.
- $\mathsf{Dec}(\mathsf{CT}, \mathsf{SK}, \mathsf{PP}) \rightarrow M$. The decryption algorithm takes as input a ciphertext encrypted under a set S of attributes, a secret key SK for an access structure $\mathbb{A}$, and the public parameters. It outputs the message M if S satisfies $\mathbb{A}$.

We now give the full security definition for KP-ABE systems. This is described by a security game between a challenger and an adversary. The game proceeds as follows:

- Setup: The challenger runs the setup algorithm and gives the public parameters PP to the adversary.
- Query Phase 1: The adversary queries the challenger for private keys corresponding to sets of access policies $\mathbb{A}_1, \ldots, \mathbb{A}_{q+1}$.
- Challenge: The adversary declares two equal length messages M_0 and M_1 and a target attribute set $S*$. This attribute set cannot be satisfied by any of the queried access policies $\mathbb{A}_1, \ldots, \mathbb{A}_{q+1}$. The challenger flips a random coin $\beta \in \{0, 1\}$, and encrypts M_β under S^*, producing CT^*. It gives CT^* to the adversary.
- Query Phase 2: The adversary queries the challenger for private keys corresponding to sets of access policies $\mathbb{A}_{q+1}, \ldots, \mathbb{A}_n$, with the added restriction that S^* does not satisfy any of these $\mathbb{A}_{q+1}, \ldots, \mathbb{A}_n$.

- Guess: The adversary outputs a guess β' for β.

The advantage of the adversary is defined as $Adv_{\mathcal{A}}(\lambda) := |Pr[\beta = \beta'] - \frac{1}{2}|$. We say a key-policy attribute-based encryption system is fully secure if all polynomial time attackers have at most a negligible advantage in this security game.

We note that the model can easily be extended to handle chosen-ciphertext attacks by allowing for decryption queries in Query Phase 1 and Query Phase 2.

Ciphertext-policy attribute-based encryption is defined in a similar way as KP-ABE, except the roles of access policy and attribute set are reversed. This also applies to the security model.

Chapter 3

Revocable IBE Systems with Almost Constant-size Key Update

Contents

In this chapter, we continue the study of revocable *IBE (RIBE) initiated by Boldyreva, Goyal, and Kumar. Their proposal of a selective secure RIBE scheme, and a subsequent construction by Libert and Vergnaud in a stronger adaptive security model are based on a binary tree approach, such that their key update size is* logarithmic *in the number of users. We ask the question of whether or not the key update size could be further reduced by using a cryptographic accumulator. We show that, indeed, the key update material can be made constant with some small amount of auxiliary information, through a novel combination of the Lewko and Waters IBE scheme and the Camenisch, Kohlweiss, and Soriente pairing-based dynamic accumulator.*

3.1 Introduction

It is sometimes necessary to remove keying material from use prior to the end of its normal key lifetime for reasons that include key compromise, removal of an entity from an organization, and so on. This process is known as *key revocation* and is used to explicitly revoke a symmetric key or the public key of a key pair, although the private key associated with the public key is also revoked [17]. Public key revocation in a conventional, certificate-based public key infrastructure (PKI) has been well studied and understood. A widely deployed revocation mechanism is through the use of a certificate revocation list (CRL) [76]. Alternatively, an Internet protocol called the online certificate status protocol (OCSP) is used to check if a certificate has been revoked [93].

In this chapter, we study public key revocation in an *identity-based encryption* (IBE) system. The idea of using an identity (or identifier) as a public key was originally conceived by Shamir [108], and subsequently realized by Cocks [48] using quadratic residues, and Boneh and Franklin [27] using pairings on elliptic curves. One very appealing property of IBE is that it alleviates cumbersome certificate management in a traditional PKI. To securely send a message to an intended receiver, the sender no longer needs to look up for the public key certificate associated with the receiver, but simply encrypts the message directly using a common set of public system parameters and the receiver's identifier, such as email address. Over the past decade, pairing-based IBE has not only received considerable attention

from academic researchers, but also attracted commercial interest from Mitsubishi, Noretech, Trend Micro, Voltage Security, and Gemplus [57], for example. Moreover, identity-based cryptographic techniques using pairings are currently undergoing standardization through the IEEE 1363.3 and the IETF S/MIME working groups. However, very few studies, for example [24, 42, 89, 107], have been devoted to key revocation thus far.

3.1.1 Motivation

Unlike a certificate-based public key, which is simply a random-looking string, a public key in the IBE setting is a user's identity. This hinders "explicit" revocation of an identity-based public key using conventional revocation mechanisms. Instead, one typically adopts a more "implicit" approach by periodically updating the corresponding private key after a pre-defined validity period, while letting the old private key expire automatically and keeping the public key (identity) unchanged. One trivial way of achieving this is by encoding a current time period into an identity during encryption. This forces a decryptor to regularly obtain her private key (corresponding to the current time period) from a key authority [27]. However, such an approach does not scale well because the key authority has to generate new keys for all the remaining non-revoked users at the beginning of each time period. Further, distribution of private keys requires establishment of secure channels between the key authority and the users. This may not be always feasible for every user.

A more desirable approach is to let the key authority broadcast some *public* information, from which the users can perform key update themselves without interacting with the key authority. Clearly, we must ensure that the broadcast information is useful only to non-revoked users, but meaningless to those who have been revoked. Hanaoka et al. [69] proposed one of the first IBE schemes that supports a *non-interactive* key revocation approach. However, their scheme requires each user to possess a special tamper-resistant hardware device that stores a secret helper key used for key update—a requirement that is likely to hinder practical deployment of the scheme.

Subsequently, Boldyreva et al. [24] proposed a scheme that obviates the need for special devices and significantly reduces the complexity of key update information from linear to *logarithmic* in the number users. They cleverly combined fuzzy IBE (FIBE) [104] with binary tree structure, which has previously

been used to improve the efficiency of certificate revocation in a traditional PKI [94, 2]. By making use of the concept of FIBE, Boldyreva et al. gave a construction they called revocable IBE (RIBE), in which a message is encrypted under two attributes, namely identity id and time t. Correspondingly, the associated decryption key comprises two components, of which the identity part is fixed (also called a long-term private key), while the time part is updated after each time period (or epoch). In order to revoke a user, the key authority simply stops issuing key update for that user in the next time period. Without the latest key update, a revoked user will no longer be able to decrypt any ciphertext generated beyond the current (expiring) time period. As with [94, 2], a binary tree can then be used to more efficiently (logarithmically) represent all the remaining non-revoked users than simply listing all the revoked or non-revoked users. In Boldyreva et al.'s RIBE scheme, each user's id is assigned to a leaf node in the binary tree and her long-term private key is generated according to the key material on each node along the path from the user's leaf node to the root. To decrypt a message encrypted under id and t, the user needs an updated decryption key (associated with t) that can be derived from the key material associated with any one of the nodes along the path from her id leaf node to the root. Hence, if a user has been revoked, such key material will not be made available in the key update broadcast by the key authority.

However, Boldyreva et al.'s scheme was proven secure in a *selective* security model, which is widely accepted as a weaker model in comparison with an *adaptive* security model. The former requires the adversary to announce the target identity and time at the beginning of a security game simulated in the model, while the latter has no such restriction. Nevertheless, Libert and Vergnaud [89] showed that adaptive security is possible. They proposed an RIBE scheme which has key update size that is also logarithmic using a similar binary tree technique, while proving their scheme to be adaptively secure. However, they achieved this at the expense of increasing the size of public parameters from constant to linear in the number of users.

The goal of this chapter is then to study whether or not we could further reduce the key update size while retaining the adaptive security requirement. We give an affirmative answer and provide a concrete construction which relies on only constant-size of key update material along with some auxiliary information, through a novel approach that combines IBE with the concept of a cryptographic

accumulator.

3.1.2 Our Approach

The key component in our approach that enables efficient key revocation and update is a pairing-based cryptographic accumulator by Camenisch et al. [34]. An accumulator, originally introduced by Benaloh and de Mare [21] as an alternative to digital signatures for secure decentralized and distributed protocols, is an algorithm that "compresses" a large set of values into a single, short value with the following two basic properties:

- For each accumulated value, it is possible to compute a *witness* that can be used to prove that a given value was indeed incorporated into the accumulator;
- Whenever a value is added or removed from the accumulator, all witnesses correspond to all the remaining values in the updated accumulator need to be re-computed as well.

The accumulator proposed by Camenisch et al. [34] is designed to address the problem of revocation of *anonymous credentials*—to efficiently prove that a hidden value has been accumulated. By making use of techniques from broadcast encryption developed by Boneh et al. [28], Camenisch et al.'s accumulator has a very nice property that allows update of witnesses to be performed very efficiently. Only one multiplication is required for addition or deletion of a value from the accumulator. Further, update of witnesses can be delegated to an untrusted entities without compromising the security of an anonymous credential system.

In this work, we take a different approach from that of [34] when considering public key revocation in the IBE setting. We combine Lewko and Waters' IBE scheme [86] with Camenisch et al.'s accumulator [34] in a particular way. Conventionally, an accumulator is used for revocation of credentials or keys in an anonymous authentication system that is based on concepts such as, group signatures or anonymous e-cash. Also, a witness is independent of the authentication system, in the sense that it is not directly used for authentication, but rather is typically used to convince a verifier that a user has or has not been revoked through a zero-knowledge proof. On the other hand, in our approach, we integrate an accumulator with a public key encryption scheme, such that the accumulator is associated with ciphertexts, while witnesses are associated with decryption keys. Particularly, our encryption algorithm

takes as input a message, an identity, and an up-to-date accumulator for current time period t, such that a target recipient is able to decrypt the resulting ciphertext using an up-to-date witness. That is, the decryption would succeed only if the recipient has not already been revoked at time t. Here, a user's decryption key comprises an identity-based key and a witness. During decryption, both the witness and the ciphertext component containing the accumulator are required to cancel out a blinding factor of the message.

Our approach of combining the IBE scheme of [86] and the accumulator of [34] requires a careful treatment. As described, one of the basic properties of an accumulator is that a witness can be used to prove that an associated value has been accumulated. Translating this into our design of RIBE, it turns out that a *collusion attack* is possible if a decryption key comprising a witness and an identity-based key is formed in a naïve manner. This is because a revoked user can collude with a non-revoked user, such that a valid witness (of the non-revoked user) can be used by the revoked user (together with her own identity-based key) to decrypt ciphertexts that she no longer has authorized access. To address this, for each user, we introduce a new secret component[1] that is associated with the accumulator and a witness, such that the secret component must be used to cancel out the blinding factor. We then bind the secret component to the user's identity-based key. Since the identity-based key is randomized for each user, two users will no longer be able to collude to share one of their witnesses to perform decryption.

Our key update method (to be performed by the key authority) through an accumulator differs in two aspects from that of existing RIBE schemes [24, 89] which make use of a binary tree. First, we simply update an accumulator according to the updated revocation list at a new time period t', generate a new witness associated with t', and create a signature over the updated accumulator. (We note that we also add t' into the accumulator to ensure that the accumulated value for each time period is always distinct even if there is no change in the revocation list between two successive time periods.) However, in the binary tree method, they first need to identify the minimal set of nodes (in the tree) for which key update needs to be published so that only non-revoked users are able to decrypt ciphertexts generated at time t'. For each node in the identified set, they then generate some key material required to update a decryption key. Second, in terms of communication overhead, our key update material comprises just a (short)

[1] Coincidentally, the secret component we adopt here is also used as a user private key in Boneh et al.'s broadcast encryption scheme [28].

accumulator, a witness, and a signature. Hence, the complexity of the size of our key update improves significantly from $O(\log(n))$ in the binary tree approach for n users to $O(1)$ with some relatively small amount of bookkeeping information by using accumulator. (We provide further details on the efficiency of our scheme in Section 3.2.5.)

In our security analysis, we adopt the Waters dual system encryption methodology [116]. As with that of [24, 89], we consider two types of adversaries: Type I adversaries that are *not* allowed to request for the private key of a target identity throughout the entire security game; and Type II adversaries that are allowed to make a query on the private key of a target identity, provided that the queried identity must subsequently be revoked before the challenge time. However, we show that our accumulator-based approach has simpler and tighter security proofs than those of a binary-tree method. To simulate a security game in the latter setting, extra care needs to be taken in order to appropriately answer any private key query that is associated with a node in the tree. Particularly, to achieve adaptive security, the simulator has to guess the position of the target identity-time pair in the tree beforehand. This causes some loss of reduction in their security proofs. Such concerns do not exist in our proofs.

In this chapter, we also show how our accumulator-based revocation technique can be improved and extended in several ways. First, we show how our RIBE system can be extended to handle more than n users. Second, we provide an RIBE scheme that achieves forward-security, in the sense that compromise of a decryption key at time t would not leak any useful information about other decryption keys for other times $t'_i \neq t$. Moreover, we describe how our technique can be used to construct a revocable ABE (RABE) scheme. These are elaborated in Section 3.3.

3.1.3 Other Related Work

After the work by Boldyreva et al. [24] and Libert and Vergnaud [89], there have been proposals on various instances of functional encryption (generalization of IBE) schemes that support revocation, such as *revocable attribute-based encryption* (RABE) and *revocable predicate encryption* (RPE) [8, 9, 63]. We note that the revocation method in the schemes of [9, 63] is different from that of existing RIBE schemes. In the RABE schemes, the users themselves are the ones who enforce key revocation instead of the key authority. This is known as *sender-local revocation* and is achieved by taking as input a

revocation list during encryption. A receiver's private key can decrypt a ciphertext only if her identity has not been included in the revocation list. This way, the users are not required to perform any private key update as with that in [24]. In [8], a key revocation system combining both approaches from [24] and [9] was proposed.

Moreover, there exist proposals on revocable IBE schemes with *mediators* [15, 26, 53, 88]. Here, a mediator is a semi-trusted authority that helps users to decrypt ciphertexts. If a user has been revoked, the mediator simply stops decrypting for the user. Such an approach, while interesting, does not seem to be satisfactory as it requires interactions between the mediator and the users for decryption of each ciphertext.

Recently, Chen et al. [42] proposed an RIBE scheme based on *lattices* under a similar security model as [24]. Also, Seo and Emura [107] gave a pairing-based RIBE construction and proved that it is secure under a new security model, which considers not only exposure of long-term private keys, but also *exposure of decryption keys*[2] (associated to each time period).

3.1.4 Outline

The rest of this chapter is organized as follow: in Section 3.2, we present our RIBE construction and its security proofs. In Section 3.3, we discuss some extensions to our scheme. We summarize and highlight some open problems in Section 3.4.

3.2 Construction

3.2.1 Intuition

Our RIBE scheme is based on the Lewko and Waters IBE scheme [86]. In our scheme, however, the decryption key of each user has two components: one is fixed (long-term) and is associated with her identity id; while the other is updated at the beginning of each time period (epoch) and corresponds to t. Particularly, the key component associated with id is essentially a normal identity-based key (in the IBE

[2] In the security model of [107], an adversary is allowed to make decryption key queries, in addition to the conventional private key queries allowed in [24].

setting) combined with a secret value[3] that is associated with an accumulator, while the key component associated with t is a witness in the context of an accumulator.

All non-revoked users' identities are captured through an accumulator. At the beginning of each time period t, the key authority adds t to the accumulator and generates the corresponding witness (i.e., the values contained in the up-to-date accumulator are current time period and the identities of all legitimate users under this period). The key authority then broadcasts the updated accumulator and witness (with respect to t) to all users, who will then update their respective existing witnesses. We note here that the integrity of the accumulator is protected through a standard signature. To encrypt a message intended for id at time t, the encryptor makes use of the updated accumulator as part of the ciphertext. To successfully decrypt a ciphertext, on the other hand, the decryptor must possess the correct identity-based key associated with id and updated witness corresponds to t. To revoke a user, the key authority simply removes the identity of the user from the accumulator. A revoked user would not be able to update his witness and therefore, would not be able to decrypt any ciphertext generated beyond the current epoch.

3.2.2 Construction

In addition to the Lewko and Waters IBE scheme [86], our RIBE construction makes use of two other building blocks: Camenisch et al.'s accumulator [34], and any standard public key signatures scheme PKS with three algorithms: the key generation algorithm PKSGen, the signing algorithm PKSSig and the verification algorithm PKSVer.

Let ϕ denote a one-to-one map from a string (id or t) to an index i. Our RIBE construction is described as follows:

- $\mathsf{Setup}(1^k, n) \rightarrow (\mathsf{PP}, \mathsf{MK}, \mathsf{RL}, \mathsf{ST}_\emptyset)$ Performed by the key authority, the setup algorithm first chooses a bilinear group G of order $N = p_1p_2p_3$ (with 3 distinct primes), random exponents $\alpha, \gamma \in \mathbb{Z}_N$, and random group elements $u, g, h \in G_{p_1}$. It also computes $e(g,g)^\alpha$, where $e : G \times G \rightarrow G_T$ is a bilinear map.

  From the parameters $\langle N, G, G_T, e, g\rangle$, the algorithm performs the following steps:

[3] As described in Section 3.1.2, this is needed to circumvent a possible collusion attack.

1. run PKSGen to generate a private-public key pair (sk, pk);
2. calculate $\mathsf{z} = e(g,g)^{(\gamma^{n+1})} \in G_T$ and $P_i = g^{(\gamma^i)} \in G_{p_1}$ for $i = 1, 2, \ldots, n, n+2, \ldots, 2n$, where γ is randomly chosen from $\mathbb{Z}_N$;
3. choose a random $\beta \in \mathbb{Z}_N$ and compute $g^\beta \in G_{p_1}$.

Let U be the bookkeeping information of all the elements that have ever been added into the accumulator (but not necessarily contained in the current accumulator), and at the point of system setup, $U = \emptyset$. The Setup algorithm then sets the accumulator $\mathsf{AC}_\emptyset = 1$ and state

$$\mathsf{ST}_\emptyset = \{U, P_1, \ldots, P_n, P_{n+2}, \ldots, P_{2n}\}.$$

The revocation list RL is initially empty.

The public parameters PP are

$$\langle N, u, g, h, g^\beta, e(g,g)^\alpha, \mathsf{z}, pk, \mathsf{AC}_\emptyset, P_1, \ldots, P_n, P_{n+2}, \ldots, P_{2n}\rangle.$$

The master secrete key MK is $\langle \alpha, \beta, \gamma, sk\rangle$ and a generator of G_{p_3}.

- $\mathsf{PriKeyGen}(\mathsf{PP}, \mathsf{MK}, id, \mathsf{ST}_U) \rightarrow (\mathsf{SK}_{id}, \mathsf{ST}_{U\cup\{i\}})$ This step is also performed by the key authority. Let V denotes the bookkeeping information of the values that have *currently* been accumulated (so V is a subset of U). Given $i = \phi(id) \in [n]$, the private key generation algorithm performs the following steps:

  1. compute $w_i = \prod_{j\in V, j\neq i} P_{n+1-j+i}$;
  2. update the accumulator and state such that

$$\mathsf{AC}_{V\cup\{i\}} = \mathsf{AC}_V \cdot P_{n+1-i} \text{ and}$$
$$\mathsf{ST}_{U\cup\{i\}} = \{U \cup \{i\}, P_1, \ldots, P_n, P_{n+2}, \ldots, P_{2n}\}.$$

The $\mathsf{PriKeyGen}$ algorithm then chooses a random $r \in \mathbb{Z}_N$, and random elements $R_3, R_3' \in G_{p_3}$.

The private key SK_{id} is then:

$$\langle K_1 = g^r R_3,\ K_2 = g^{\alpha}(u^{id}h)^r P_i^{\beta} R_3',\ K_3 = w_i \rangle.$$

The PriKeyGen algorithm also prepares a set V_w, which denotes the values contained in the accumulator when a witness w_i was created (so V_w is fixed for each user and it is also a subset of U). This set V_w is given to the user along with his private key SK_{id}. (We give a simple example in Appendix 3.2.3 illustrating how sets V and V_w are derived and updated.)

- $\mathsf{KeyUpd}(\mathsf{PP}, \mathsf{MK}, t, \mathsf{RL}, \mathsf{ST}_U) \rightarrow \mathsf{KU}_t$ At the start of each new time period t, the key update authority first updates the accumulator by performing the following steps:

  1. remove $l' = \phi(t')$ associated with the just expired time period t' from V;
  2. remove all $i = \phi(id)$ that corresponds to t' in RL from V;
  3. update the accumulator, that is $\mathsf{AC}_V = \prod_{i' \in V} P_{n+1-i'}$ for all i' in the updated V.

  The KeyUpd algorithm then adds the new time period $l = \phi(t) \in [n]$ following the same steps as before (in PriKeyGen) to obtain the latest accumulator $\mathsf{AC}_{V \cup \{l\}}$. It then generates a signature σ_l on $\mathsf{AC}_{V \cup \{l\}}$. The algorithm also prepares a set ΔV, which contains a list of recently joined and revoked users' identities within the last (just expired) epoch. Then the KeyUpd algorithm broadcasts $\mathsf{KU}_t = \langle \mathsf{AC}_{V \cup \{l\}},\ \sigma_l,\ w_l \rangle$, together with the set ΔV, to all users.

- $\mathsf{DecKeyGen}(\mathsf{SK}_{id}, \mathsf{KU}_t) \rightarrow \mathsf{DK}_{id,t}$ This is run by the receiver. The decryption key generation algorithm first checks if:

  1. $i = \phi(id),\ l = \phi(t) \in V$;
  2. σ_l is a valid signature associated with AC_V using the PKSVer algorithm and pk;
  3. $e(P_l, \mathsf{AC}_V)/e(g, w_l) = \mathsf{z}$ to ensure the correctness of AC_V.

  We set a Boolean flag denoted by DecKeyChk to 0 if *any* of the above three checks fails. If all the three conditions are satisfied, we set $\mathsf{DecKeyChk} = 1$. If $\mathsf{DecKeyChk} = 0$, then the

$\mathsf{DecKeyGen}$ algorithm outputs a special symbol $\perp$. Otherwise, $\mathsf{DecKeyGen}$ replaces the existing accumulator with an up-to-date one. It then updates the witness and computes the decryption key as follows:

1. if $i \in V$ and $V \cup V_w \subset U$, compute

$$w_i' = w_i \cdot \frac{\prod_{j \in V \setminus V_w} P_{n+1-j+i}}{\prod_{j \in V_w \setminus V} P_{n+1-j+i}};$$

2. otherwise, output $\perp$.

Set the decryption key $\mathsf{DK}_{id,t}$ to be

$$\langle K_1 = g^r R_3,\ K_2 = g^\alpha (u^{id} h)^r P_i^\beta R_3',\ K_3 = w_i' \rangle.$$

- $\mathsf{Enc}(\mathsf{PP}, M, id, \mathsf{AC}_V) \rightarrow \mathsf{CT}_{id,t}$ Given a message M and an up-to-date accumulator AC_V containing current time t, the message encryptor chooses $s \in \mathbb{Z}_N$ randomly, and set the ciphertext $\mathsf{CT}_{id,t}$ to be

$$C = M \frac{e(g,g)^{\alpha s}}{\mathsf{z}^s},\ C_0 = g^s,\ C_1 = (u^{id} h)^s,\ C_2 = (g^\beta \mathsf{AC}_V)^s.$$

- $\mathsf{Dec}(\mathsf{PP}, \mathsf{DK}_{id,t}, \mathsf{CT}_{id,t}) \rightarrow M$ The decryption algorithm computes

$$\frac{e(C_0, K_2 K_3)}{e(C_1, K_1) e(P_{\phi(id)}, C_2)} = \frac{e(g,g)^{\alpha s}}{\mathsf{z}^s}.$$

The message M can be recovered by dividing C by the computed term.

- $\mathsf{KeyRev}(id, t, \mathsf{RL}, \mathsf{ST}_U) \rightarrow \mathsf{RL}$ The key revocation algorithm adds (id, t) to the revocation list RL if $i = \phi(id) \in \mathsf{ST}_U$. This step is performed by the key authority.

CORRECTNESS. We now verify that the decryption algorithm works correctly. First notice a correct accumulator is always in the form of $\mathsf{AC}_V = \prod_{j \in V} P_{n+1-j}$, and the witness w_i for each $i \in V$ always

has a value $w_i = \prod_{j \in V, j \neq i} P_{n+1-j+i}$. Hence, the following equation always holds:

$$\frac{e(P_i, \mathsf{AC}_V)}{e(g, w_i)} = \frac{e(g,g)^{\sum_{j \in V}(\gamma^{n+1-j+i})}}{e(g,g)^{\sum_{j \in V, j \neq i}(\gamma^{n+1-j+i})}} = e(g,g)^{(\gamma^{n+1})} = \mathsf{z}.$$

Thus we have

$$\begin{aligned}
&\frac{e(C_0, K_2 K_3)}{e(C_1, K_1) e(P_{\phi(id)}, C_2)} \\
&= \frac{e(g^s, g^{\alpha}(u^{id}h)^r P_{\phi(id)}^{\beta} R_3' \cdot w_{\phi(id)})}{e((u^{id}h)^s, g^r R_3) e(P_{\phi(id)}, (g^{\beta}\mathsf{AC}_V)^s)} \\
&= \frac{e(g^s, g^{\alpha}(u^{id}h)^r R_3')}{e((u^{id}h)^s, g^r R_3)} \cdot \frac{e(g^s, P_{\phi(id)}^{\beta} w_{\phi(id)})}{e(P_{\phi(id)}, (g^{\beta}\mathsf{AC}_V)^s)} \\
&= \frac{e(g,g)^{\alpha s} e(g, u^{id}h)^{rs}}{e(u^{id}h, g)^{rs}} \cdot \frac{e(g, P_{\phi(id)})^{\beta s} e(g, w_{\phi(id)})^s}{e(P_{\phi(id)}, g)^{\beta s} e(P_{\phi(id)}, \mathsf{AC}_V)^s} \\
&= \frac{e(g,g)^{\alpha s}}{\mathsf{z}^s}.
\end{aligned}$$

REMARK. In comparison with the Lewko and Waters IBE scheme, our identity-based private key component K_2 has an additional secret value P_i^{β}. Moreover, our ciphertexts are different in two aspects: the blinding factor of our ciphertext component C has an additional term z^{-s}, and we have an additional ciphertext component C_2.

It is also worth stressing again that our scheme enforces key revocation through decryption (at the recipient), that is, a ciphertext recipient can decrypt properly only if he uses an updated decryption key. An encryptor may or may not know the set V (which is associated with a revocation list),[4] and hence, may not always know if a target recipient has been revoked.

3.2.3 Example

In this section, we provide a simple but illustrative example to show how the sets V, V_w, U are constructed, how a user witness is generated, and how to perform key revocation and update. For ease of exposition, we omit the private key generation process and abuse some of the notations.

[4]Recall that in IBE, anyone can encrypt to an identity using the appropriate public parameters (even without having a decryption key).

Scenario: Let's assume that we currently have id_i, id_2, id_3 as the legitimate users in the system under current time period t. For simplicity and without loss of generality, let $\phi(id_i) = i$ for $i = 1, 2, 3$ and $\phi(t) = 4$. We then have $V = \{1, 2, 3, 4\}$ and $\mathsf{AC}_V = \prod_{j \in V} P_{n+1-j} = P_n P_{n-1} P_{n-2} P_{n-3}$. Also, we have $U = V$.

Add user: Now assume that user id_5, where $\phi(id_5) = 5$, joins the system within the same time period t. The key authority (KA) first updates the set V as $V = \{1, 2, 3, 4\} \cup \{5\}$, then issues a witness $w_5 = \prod_{j \in V, j \neq 5} P_{n+1-j+5} = P_{n+5} P_{n+4} P_{n+3} P_{n+2}$ and a unique set $V_w = \{1, 2, 3, 4, 5\}$ to user id_5. Moreover, the KA updates the accumulator $\mathsf{AC}_V = P_n P_{n-1} P_{n-2} P_{n-3} P_{n-4}$ according to the most recent set V. Also it updates U as $U = \{1, 2, 3, 4\} \cup \{5\}$. Note that no key update information will be broadcast to other system users since the time period t has not elapsed yet.

Key update by KA: Assume that user id_2 has been revoked before the expiry of time period t. At the beginning of a subsequent new time period t', where $\phi(t') = 6$, the KA performs the following steps:

1. Remove $\{2, 4\}$ from V (recall 2 corresponds to user id_2 and 4 corresponds to time period t) and add $\{6\}$ into V, obtaining $V = \{1, 3, 5, 6\}$; then compute a new accumulator $\mathsf{AC}_V = P_n P_{n-2} P_{n-4} P_{n-5}$, and update the set U as $U = \{1, 2, 3, 4, 5\} \cup \{6\}$. (Recall U is the set containing all elements that have ever been added into the accumulator.) The KA also sets ΔV to be $\{2, 4, 5, 6\}$.
2. Calculate the witness for time period t' as $w_6 = P_{n+6} P_{n+4} P_{n+2}$.
3. Generate a signature $\sigma_{t'}$ on $\mathsf{AC}_{V''}$, broadcast $\mathsf{KU}_{t'} = \langle \mathsf{AC}_{V''}, \sigma_{t'}, w_6 \rangle$ together with the set ΔV.

Key update by user: During key update, each user first verifies the integrity and authenticity of the accumulator, as described in the scheme. Then, the user derives the latest set V and update his witness. For example, for user id_5, he first obtains the set $V = \{1, 3, 5, 6\}$ (this could easily be calculated from the broadcast set ΔV and his own set V_w); using his own set $V_w = \{1, 2, 3, 4, 5\}$, he then derives $V \backslash V_w = \{6\}$ and $V_w \backslash V = \{2, 4\}$, and updates w_5 as described in the scheme to obtain the latest witness used for decryption.

Encrypt & decrypt: The encryption and decryption algorithms are straightforward.

From the above example, one can easily verify that an accumulator is always in the form of $\mathsf{AC}_V = \prod_{j \in V} P_{n+1-j}$ and a witness for value i has the value $w_i = \prod_{j \in V, j \neq i} P_{n+1-j+i}$.

3.2.4 Security Analysis

Overview

In our security analysis, we consider the following two types of adversaries:

- Type I adversaries that never make a private key query on the target identity id^* at any time throughout the game.
- Type II adversaries that are allowed to make a private key query on the target identity id^* at some point of the game, provided that the queried identity must subsequently be revoked before the challenge time t^*.

We then prove the security of our RIBE scheme by adopting the dual system encryption technique by Waters [116]. In our proofs, private keys and ciphertexts take two forms: normal or semi-functional. A normal private key could decrypt a ciphertext, which in turn, is either normal or semi-functional; while a semi-functional private key can only decrypt a normal ciphertext. When using a semi-functional key to decrypt a semi-functional ciphertext, the decryption will fail. We then use a hybrid argument through a sequence of games to prove the security of our scheme. We first change the challenge ciphertext to semi-functional, then gradually change the private keys into semi-functional one by one. At the very last step, we change the semi-functional ciphertext into an encryption of a random message, in which the adversary has no advantage at all. Particularly, we prove that neither Type I nor Type II adversary learns any useful information about the chosen message from the challenge ciphertext, even when they are provided some information on the associated blinding factor.

Security proofs

We first define the semi-functional ciphertexts and semi-functional keys that will be used in our proofs.

Semi-functional Ciphertext Let g_2 denote a generator of the subgroup G_{p_2}. A semi-functional ciphertext is then created as follows: first, generate a normal ciphertext C', C'_0, C'_1, C'_2 using the encryption algorithm; then choose random exponents $x, z_c, z_d \in \mathbb{Z}_N$ and set C to be C', C_0 to be $C'_0 g_2^x$, C_1 to be $C'_1 g_2^{x z_c}$, and C_2 to be $C'_2 g_2^{z_d}$. The resulting tuple (C, C_0, C_1, C_2) is a semi-functional ciphertext.

Semi-functional Key We create semi-functional key as follows: first generate a set of normal key K'_1, K'_2, K'_3 with the key generation algorithm; then choose random exponents $\delta, z_k \in \mathbb{Z}_N$ and set K_1 to be $K'_1 g_2^{\delta}$, K_2 to be $K'_2 g_2^{\delta z_k}$, and $K_3 = K'_3$ is kept unchanged. The resulting tuple (K_1, K_2, K_3) is a semi-functional key.

If a semi-functional key is used to decrypt a semi-functional ciphertext, an additional factor $e(g_2, g_2)^{x\delta(z_k - z_c)}$ will hinder the decryption. If $z_c = z_k$, the decryption will still work, in this case we call the key *nominally* semi-functional: it is in the semi-functional form but still allows decryption.

Our proofs rely on Assumptions 1, 2, 3, and 5 defined in Section 2.2, and a hybrid argument through a sequence of games: $\mathbf{Game}_{real}$, $\mathbf{Game}_{restricted}$, $\mathbf{Game}_k$, and $\mathbf{Game}_{final}$ as defined in [86], for $0 \leq k \leq q$ where q denotes the number of key queries the attacker makes. We recall them here: The first game, $\mathbf{Game}_{real}$, will be the real security game. The next game, $\mathbf{Game}_{restricted}$, will be like the real security game except that the attacker cannot ask for keys for identities which are equal to the challenge identity modulo p_2. This is a stronger restriction than the real security game, where the identities must be unequal modulo N. We will retain this stronger restriction throughout the subsequent games. We let q denote the number of key queries the attacker makes. For k from 0 to q, we denote $\mathbf{Game}_k$ as:

$\mathbf{Game}_k$: This is like the restricted security game, except that the ciphertext given to the attacker is semi-functional and the first k keys are semi-functional. The rest of the keys are normal.

However, different from [86], we split $\mathbf{Game}_{final}$ into two games, corresponding to two types of adversaries, as follows:

$\mathbf{Game}_{final\_1}$ is the same as $Game_q$ except that the challenge ciphertext is a semi-functional encryption of a random message, embedded with an instance of the hard problem defined by Assumption 3.

$\mathbf{Game}_{final\_2}$ is the same as $Game_{final\_1}$, except that here, we embed a instance of the hard problem defined by Assumption 5.

In $\mathbf{Game}_0$, all the keys are normal and the ciphertext is semi-functional. In $\mathbf{Game}_q$, the ciphertext and all of the keys are semi-functional. Our last game is either $\mathbf{Game}_{final\_1}$ or $\mathbf{Game}_{final\_2}$, which is the same as $\mathbf{Game}_q$ except that the ciphertext is a semi-functional encryption of a random message, not one of the two messages requested by the attacker. We will prove that each of these games is indistinguishable through the following lemmas.

At the outset of the game, the simulator $\mathcal{B}$ flips a coin $coin \xleftarrow{R} \{0,1\}$ as the guess for the type of adversary it will face: 0 for Type I and 1 for Type II. The game then proceed from $Game_{real}$ to $Game_q$. After $Game_q$, if $\mathcal{B}$ faces a Type I adversary, it will proceed to $Game_{final\_1}$, otherwise, it will proceed to $Game_{final\_2}$. In what follows, we prove that these games are indistinguishable from the attacker's viewpoint.

Lemma 1. *Suppose there exists an algorithm $\mathcal{A}$ such that $Game_{real}Adv_{\mathcal{A}} - Game_{restricted}Adv_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage $\frac{\epsilon}{2}$ in breaking Assumption 2.*

Proof. Given $g, X_1X_2, X_3, Y_2Y_3, \mathcal{B}$ can simulate $Game_{real}$ with $\mathcal{A}$. With probability $\epsilon, \mathcal{A}$ produces identities id and id^* such that $id \neq id^*$ modulo N and p_2 divides $id - id^*$. If $\mathcal{A}$ fails to do this, $\mathcal{B}$ will simply guess randomly. $\mathcal{B}$ uses these identities to produce a nontrivial factor of N by computing $a =$gcd$(id - id^*, N)$. We let $b = \frac{N}{a}$. Consider the following three cases:

1. one of a, b is p_1, and the other is p_2p_3;
2. one of a, b is p_2, and the other is p_1p_3;
3. one of a, b is p_3, and the other is p_1p_2.

$\mathcal{B}$ can determine if case 1 has occurred by testing if either of $(Y_2Y_3)^a$ or $(Y_2Y_3)^b$ is the identity element. If this happens, we will suppose that $a = p_1$ and $b = p_2p_3$ without loss of generality. $\mathcal{B}$ can then learn whether T has a G_{p_2} component or not by testing if $e(T^a, X_1X_2)$ is the identity element. If it is not, then T has a G_{p_2} component.

$\mathcal{B}$ can determine if case 2 has occurred by testing if either of $(X_1X_2)^a$ or $(X_1X_2)^b$ is the identity

element. Assuming that $\mathcal{B}$ has already ruled out case 1 and neither of these is the identity element, then case 2 has occurred. $\mathcal{B}$ can learn which of a, b is equal to $p_1 p_3$ by testing which of g^a, g^b is the identity. We assume without loss of generality that $a = p_2$ and $b = p_1 p_3$. Then $\mathcal{B}$ can learn whether T has a G_{p_2} component or not by testing if T^b is the identity element. If it is not, then T has a G_{p_2} component.

$\mathcal{B}$ can determine that case 3 has occurred when the tests for case 1 and 2 fail. It can learn which of a, b is equal to p_3 by testing which of X_3^a, X_3^b is the identity. We assume without loss of generality that $a = p_3$. $\mathcal{B}$ can learn whether T has a G_{p_2} component or not by testing whether $e(T^a, Y_2 Y_3)$ is the identity. If it is not, then T has a G_{p_2} component. □

Lemma 2. *Suppose there exists an algorithm $\mathcal{A}$ such that $Game_{restricted} Adv_{\mathcal{A}} - Game_0 Adv_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 1.*

Proof. $\mathcal{B}$ receives g, X_3, T and simulates either $Game_{restricted}$ or $Game_0$ with $\mathcal{A}$. $\mathcal{B}$ sets the public parameters as follows: it chooses random exponents $a, b, \alpha, \beta, \gamma \in \mathbb{Z}_N$ and sets $g = g, u = g^a, h = g^b$; computes $\mathsf{z} = e(g,g)^{\gamma^{n+1}}$, $P_i = g^{\gamma^i}$ for $i \in [2n] \backslash \{n+1\}$, and sets $\mathsf{AC}_\emptyset = 1$; and generates a public-private key pair (sk, pk) using PKSGen. $\mathcal{B}$ then forwards the public parameters $\langle N, u, g, h, g^\beta, e(g,g)^\alpha, \mathsf{z}, pk, \mathsf{AC}_\emptyset, P_i \rangle$ to $\mathcal{A}$.

Whenever $\mathcal{B}$ is asked to provide a private key for an identity id_i, it chooses random exponents $r_i, t_i, y_i \in \mathbb{Z}_N$. Using the set V which contains the current accumulated values, $\mathcal{B}$ sets $K_1 = g^{r_i} X_3^{t_i}$, $K_2 = g^\alpha (u^{id_i} h)^{r_i} P_i^\beta X_3^{y_i}$, and $K_3 = \prod_{j \in V, j \neq i} P_{n+1-j+i}$.

If $\mathcal{B}$ is asked for a key update, it calculates the new accumulator (with the current sets V and U) as $\mathsf{AC} = \prod_{j \in V} P_{n+1-j}$ and creates a signature σ_j for the accumulator using key sk. $\mathcal{B}$ then publishes $\langle \mathsf{AC}_V, \sigma_j, w_j \rangle$.

At the challenge phase, $\mathcal{A}$ sends $\mathcal{B}$ two messages, M_0 and M_1, along with a challenge identity-time pair (id^*, t^*). $\mathcal{B}$ first checks the Boolean flag $\mathsf{DecKeyChk}$ using an up-to-date accumulator associated with t^* and the given target identity-time pair. If $\mathsf{DecKeyChk} = 0$, then $\mathcal{B}$ just guesses randomly which group the value T belongs to; otherwise $\mathcal{B}$ chooses $d \in \{0, 1\}$ randomly and sets the challenge ciphertext as:

$$C = M_d \frac{e(g,T)^\alpha}{e(g,T)^{\gamma^{n+1}}},\ C_0 = T,\ C_1 = T^{a \cdot id^* + b},\ C_2 = T^{\beta + \sum_{j \in V} \gamma^{n+1-j}}.$$

First note this implicitly sets g^s to be equal to the G_{p_1} part of T, and hence the value z^s is computed as $e(g,T)^{\gamma^{n+1}}$. We note that a normal C_2 component is in the form of $(g^{\beta}\mathsf{AC}_V)^s = (g^{\beta}\prod_{j\in V} g^{\gamma^{n+1-j}})^s = (g^{\beta} g^{\sum_{j\in V}\gamma^{n+1-j}})^s = (g^s)^{\beta+\sum_{j\in V}\gamma^{n+1-j}}$. Thus, if $T \in G_{p_1p_2}$, then the ciphertext is semi-functional with $z_c = a\cdot id^* + b$ and $z_d = \beta + \sum_{j\in V}\gamma^{n+1-j}$. Here the values z_c and z_d modulo p_2 are independent of the values of a, b, β, γ modulo p_1, so they are properly distributed. If $T \in G_{p_1}$, this is a normal ciphertext. Hence $\mathcal{B}$ can use the output of $\mathcal{A}$ to distinguish T. □

Lemma 3. *Suppose there exists an algorithm $\mathcal{A}$ such that $Game_{k-1}Adv_{\mathcal{A}} - Game_kAdv_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 2.*

Proof. $\mathcal{B}$ first receives $g, X_1X_2, X_3, Y_2Y_3, T$. It sets the public parameters as with those of Lemma 2.

If $\mathcal{A}$ requests for the i^{th} key for id_i where $i < k$, $\mathcal{B}$ creates a semi-functional key by choosing random exponents $r_i, t_i, y_i \in \mathbb{Z}_N$ and setting $K_1 = g^{r_i}(Y_2Y_3)^{t_i}$, $K_2 = g^{\alpha}(u^{id_i}h)^{r_i}P_i^{\beta}(Y_2Y_3)^{y_i}$, $K_3 = \prod_{j\in V, j\neq i} P_{n+1-j+i}$. Note that this is a properly distributed semi-functional key with $g_2^{\delta} = Y_2^{t_i}$. As with Lemma 2, the values of t_i and y_i modulo p_2 and modulo p_3 are uncorrelated by the Chinese Reminder Theorem. To handle private key queries for $i > k$, $\mathcal{B}$ generates normal keys using random exponents $r_i, t_i, y_i \in \mathbb{Z}_N$ and setting: $K_1 = g^{r_i}X_3^{t_i}$, $K_2 = g^{\alpha}(u^{id_i}h)^{r_i}P_i^{\beta}X_3^{y_i}$, $K_3 = \prod_{j\in V, j\neq i} P_{n+1-j+i}$. To answer the k^{th} private key query, $\mathcal{B}$ simply sets $z_k = a\cdot id_k + b$, chooses a random exponent $t_k \in \mathbb{Z}_N$, and sets: $K_1 = T$, $K_2 = g^{\alpha}T^{z_k}P_i^{\beta}X_3^{t_k}$, $K_3 = \prod_{j\in V, j\neq i} P_{n+1-j+i}$.

At the challenge phase, $\mathcal{A}$ sends $\mathcal{B}$ two messages, M_0 and M_1, together a challenge identity-time pair (id^*, t^*). $\mathcal{B}$ first uses the current accumulator associated with t^* and the challenge pair to check the value of $\mathsf{DecKeyChk}$. If $\mathsf{DecKeyChk} = 0$ then $\mathcal{B}$ just guesses randomly, otherwise $\mathcal{B}$ chooses $d \in \{0,1\}$ randomly and sets the ciphertext as:

$$C = M_d\frac{e(g,X_1X_2)^{\alpha}}{e(g,X_1X_2)^{\gamma^{n+1}}},\ C_0 = X_1X_2,$$

$$C_1 = (X_1X_2)^{a\cdot id^*+b},\ C_2 = (X_1X_2)^{\beta+\sum_{j\in V}\gamma^{n+1-j}}.$$

In this lemma, the argument that $\mathcal{B}$ could only make a nominally semi-functional key k is similar to that of [86, Lemma 7]. If $T \in G_{p_1p_3}$, then $\mathcal{B}$ has properly simulated $Game_{k-1}$. If $T \in G$, then $\mathcal{B}$ has properly simulated $Game_k$. From the output of $\mathcal{A}$, $\mathcal{B}$ could distinguish the possibilities for T. □

Lemma 4. *Suppose there exists an algorithm $\mathcal{A}$ such that $Game_q Adv_{\mathcal{A}} - Game_{final\_1} Adv_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 3.*

Proof. $\mathcal{B}$ first receives $g, g^{\alpha}, X_3, g^s Y_2, Z_2, T$. It sets public parameters as with those of Lemma 2 except that it sets $e(g,g)^{\alpha}$ as $e(g^{\alpha} X_2, g)$ instead of choosing α randomly. Additionally, $\mathcal{B}$ computes $\mathsf{z}^s = e(g, g^s Y_2)^{\gamma^{n+1}}$ and passes it to $\mathcal{A}$. When $\mathcal{A}$ requests for the private key for id_i, $\mathcal{B}$ generates a semi-functional key, chooses random exponents $c_i, r_i, t_i, x_i, y_i \in \mathbb{Z}_N$, and sets: $K_1 = g^{r_i} Z_2^{x_i} X_3^{t_i}$, $K_2 = g^{\alpha} X_2 (u^{id_i} h)^{r_i} P_i^{\beta} Z_2^{c_i} X_3^{y_i}$, $K_3 = \prod_{j \in V, j \neq i} P_{n+1-j+i}$. At the challenge phase, $\mathcal{A}$ sends $\mathcal{B}$ two messages, M_0 and M_1, together with a challenge identity-time pair (id^*, t^*). $\mathcal{B}$ first checks the value of DecKeyChk. If DecKeyChk $= 0$, then $\mathcal{B}$ just guesses randomly, otherwise $\mathcal{B}$ chooses $d \in \{0,1\}$ randomly and sets the ciphertext as:

$$C = M_d \frac{T}{e(g, g^s Y_2)^{\gamma^{n+1}}},\ C_0 = g^s Y_2,$$

$$C_1 = (g^s Y_2)^{a \cdot id^* + b},\ C_2 = (g^s Y_2)^{\beta + \sum_{j \in V} \gamma^{n+1-j}}.$$

This implicitly sets $z_c = a \cdot id^* + b$. Note that although $u = g^a, h = g^b$ and $z_c = a \cdot id^* + b$, since u, h are elements of G_{p_1} (meaning modulo p_1) and z_c is modulo p_2, their values are independent from each other.

Given the value $\mathsf{z}^s = e(g, g^s Y_2)^{\gamma^{n+1}}$, the remaining task of $\mathcal{A}$ is simply to distinguish the possibilities of T. If $T = e(g,g)^{\alpha s}$, then this is a properly generated semi-functional ciphertext with message M_d. If T is a random element of G_T, then the ciphertext is a semi-functional one with a random message. Thus $\mathcal{B}$ could use the output of $\mathcal{A}$ to distinguish between the possibilities of T. □

Lemma 5. *Suppose there exists an algorithm $\mathcal{A}$ such that $Game_q Adv_{\mathcal{A}} - Game_{final\_2} Adv_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking the OBDHE assumption.*

Proof. $\mathcal{B}$ first receives an instance of the OBDHE problem:

$$g, f, \{P_i\}_{i \in \{1,\ldots,2n\} \setminus \{n+1\}}, v, T$$

where $f = g^s$ for some random $s \in \mathbb{Z}_N$, $P_i = g^{\gamma^i}, v = g^{\beta}$, together with oracle access to $\mathcal{O}_{g,e}^{DH}$

as specified by the assumption. $\mathcal{B}$ is required to distinguish if $T = e(g^{\gamma^{n+1}}, f)$ or T is a random element of G_T. $\mathcal{B}$ then chooses random exponents $a, b, \alpha \in \mathbb{Z}_N$ and sets $g = g, u = g^a, h = g^b$. It can compute $\mathsf{z} = e(g,g)^{\gamma^{n+1}}$ using a pair (P_{n+1-i}, P_i) for any i and calculate $e(P_{n+1-i}, P_i)$. It sets $\mathsf{AC}_\emptyset = 1$. Moreover, $\mathcal{B}$ generates a public-private key pair (sk, pk) using PKSGen and forwards the public parameters $\langle N, g, u, h, g^\beta, e(g,g)^\alpha, \mathsf{z}, pk, \mathsf{AC}_\emptyset, P_i\rangle$ to $\mathcal{A}$. Additionally, $\mathcal{B}$ computes and returns $e(g,g)^{\alpha s} = e(g,f)^\alpha$ to $\mathcal{A}$.

When $\mathcal{A}$ requests for the private key for id_i, $\mathcal{B}$ generates a semi-functional key by choosing random exponents $c_i, r_i, t_i, x_i, y_i \in \mathbb{Z}_N$, random $Z_2 \in G_{p_2}$, $X_3 \in G_{p_3}$. $\mathcal{B}$ first computes $K_1 = g^{r_i} Z_2^{x_i} X_3^{t_i}$ and $K_3 = \prod_{j\in V, j\neq i} P_{n+1-j+i}$. To provide the key component K_2 for id_i, $\mathcal{B}$ queries the oracle $\mathcal{O}^{DH}_{g,e}(P_i, v)$. Since $v = g^\beta$, the oracle's output equals to P_i^β. Then $\mathcal{B}$ sets $K_2 = g^\alpha (u^{id_i} h)^{r_i} P_i^\beta Z_2^{c_i} X_3^{y_i}$.

At the challenge phase, $\mathcal{A}$ sends $\mathcal{B}$ two messages, M_0 and M_1, together with a challenge identity-time pair (id^*, t^*). $\mathcal{B}$ first checks the value of $\mathsf{DecKeyChk}$. If $\mathsf{DecKeyChk} = 0$ then $\mathcal{B}$ just guesses randomly, otherwise $\mathcal{B}$ chooses $d \in \{0,1\}$ and a group G_{p_2} element Y_2 randomly, and sets parts of the ciphertext as:

$$C = M_d \frac{e(g,f)^\alpha}{T},\ C_0 = fY_2,\ C_1 = (fY_2)^{a\cdot id^* + b}.$$

To transform the ciphertext component C_2 into semi-functional, $\mathcal{B}$ computes $\omega = v \prod_{j\in V} P_{n+1-j}$ and makes an oracle query $\mathcal{O}^{DH}_{g,e}(\omega, f)$. Upon receiving the output $h' = (g^\beta \prod_{j\in V} P_{n+1-j})^s = (g^\beta \mathsf{AC}_V)^s$, $\mathcal{B}$ chooses a random $u_i \in \mathbb{Z}_N$ and sets $C_2 = h' Y_2^{u_i}$.

Given the value $e(g,g)^{\alpha s} = e(g,f)^\alpha$, the remaining task of $\mathcal{A}$ is simply to distinguish the possibilities of T. If $T = e(g^{\gamma^{n+1}}, h) = e(g,g)^{\gamma^{n+1} s} = \mathsf{z}^s$, then this is a properly generated semi-functional ciphertext with message M_d. If T is a random element of G_T, then the ciphertext is a semi-functional one with a random message. Thus $\mathcal{B}$ could use the output of $\mathcal{A}$ to distinguish between the possibilities of T. □

Theorem 1. *If Assumptions 1, 2, 3, and 5 hold, then our RIBE scheme is secure.*

The proof for Theorem 1 follows from Lemmas 1 to 5.

3.2.5 Efficiency

We now compare the efficiency of our construction against existing pairing-based RIBE schemes. This is illustrated in Table 3.1. We let $\tilde{n}$ denote the number of users in the system, $\hat{n}$ denote the size of identity space representing $\tilde{n}$ many users, r denote the number of revoked users, and $r' = |\Delta V|$, where ΔV is as defined in Section 3.2.2. Also, we let PP denote public parameters, DK denote decryption key, CT denote ciphertext, KUp denote key update, Dec denote decryption, SM denote security model, and Group denote the underlying bilinear group. The sizes for PP, DK, CT, and KUp are measured in the number of group elements; Dec is measured as the number of pairing operations; SM is either selective or adaptive and Group is either prime or composite.

Table 3.1: A comparison between existing and our RIBE schemes.

	PP size	DK size	CT size	KUp size	Dec	SM	Group
BGK[24]	$O(1)$	4	4	$O(\log(\hat{n}))$	4	select.	prime
LV[89]	$O(\hat{n})$	4	5	$O(\log(\hat{n}))$	3	adapt.	prime
SE[107]	$O(\hat{n})$	3	4	$O(\log(\hat{n}))$	3	adapt.	prime
Ours	$O(\tilde{n})$	3	4	$O(1)$	3	adapt.	composite

From Table 3.1, we see that all adaptively secure RIBE schemes, including ours, have comparable DK and CT sizes, and the computational overhead of Dec. However, our scheme has a clear advantage of having constant size KUp.

We note that during key update, the key authority of all the above considered schemes also broadcasts some auxiliary information with respect to non-revoked/revoked user identities. As shown in the analysis of the key update algorithm in [24], the binary tree approach is most advantageous when $r \leq \frac{\tilde{n}}{2}$, in which case the complexity of key update is $O(r \log(\frac{\tilde{n}}{r}))$. (For the case of $r > \frac{\tilde{n}}{2}$, we simply assume that the scheme can be "reset" to retain the efficiency of key update.) By considering only the case of $r \leq \frac{\tilde{n}}{2}$, we have $\frac{\tilde{n}}{r} \geq 2$ and $\log(\frac{\tilde{n}}{r}) \geq 1$, and thus we have $O(r \log(\frac{\tilde{n}}{r})) \geq O(r)$. In the BGK, LV, SE schemes, therefore, the auxiliary information required during key update has complexity of $O(r)$, while ours has complexity of $O(r')$. In reality, we have $r' < r$, or even $r' \ll r$. Consider a concrete example by letting $r' = |\Delta V| = 100$ and assuming a user identity is of 32-bits, our bookkeeping information during key update consumes only 400 bytes.

3.3 Extensions

3.3.1 Supporting More Than n Users

Our scheme presented in Section 3.2.2 supports up to only n users. However, as shown by Phan et al. [98] in their dynamic broadcast encryption scheme, our scheme similarly can handle polynomially many more than n users (but still bounded) and remains secure under a generalization [98, 52] of the decisional bilinear Diffie-Hellman Exponent (BDHE) assumption [25] defined as follow:

Assumption 8. *Given the input* $g^{\mathsf{P}(x_1)}, \ldots, g^{\mathsf{P}(x_n)} \in G_{p_1}$ *and* $e(g,g)^{\mathsf{Q}(x_1)}, \ldots, e(g,g)^{\mathsf{Q}(x_n)} \in G_T$ *for random choices of* $x_1, \ldots, x_n \in \mathbb{Z}_N$*, the generalized decisional BDHE (GBDHE) assumption says that it is hard to decide between* $e(g,g)^{f(x_1)}, \ldots, e(g,g)^{f(x_n)} \in G_T$ *and a random* $T' \in G_T$ *if polynomial* f *is independent of polynomials* P *and* Q*.*

Recall that the decisional BDHE assumption, typically parameterized by n and denoted by n-BDHE, is an instance of the GBDHE assumption defined as follows:

Assumption 9. *Given the input* $g, h, \{g_k = g^{\alpha^k}\}_{k \in \{1,\ldots,2n\} \setminus \{n+1\}}$ *for random* $g, h \in G_{p_1}$ *and* $\alpha \in \mathbb{Z}_N$*, it is hard to decide between* $e(g_{n+1}, h)$ *and a random* $T' \in G_T$*.*

Hence, let m be the new upper bound of the number of supported users, the security of our RIBE supporting more than n users (where the next user is numbered $n+2$) can be proved by considering the following assumption:

Assumption 10. *Given the input* h *and* $\{g_k = g^{\alpha^k}\}$ *for* $k \in \{n+1-m, \ldots, n+1+m\} \setminus \{n+1\}$ *for random* $g, h \in G_{p_1}$ *and* $\alpha \in \mathbb{Z}_N$*, it is hard to decide between* $e(g_{n+1}, h)$ *and a random* $T' \in G_T$*.*

Note that Assumption 10 is equivalent to the following assumption:

Assumption 11. *Given the input* h *and* $\{g_k = g^{\alpha^k}\}$ *for* $k \in \{1, \ldots, 2m\} \setminus \{m\}$ *for random* $g, h \in G_{p_1}$ *and* $\alpha \in \mathbb{Z}_N$*, it is hard to decide between* $e(g_m, h)$ *and a random* $T' \in G_T$*.*

We have, therefore, $m \geq n+2$ being the new upper bound, since Assumption 11 is comparable to the m-BDHE assumption, which in fact, is also an instance of the GBDHE assumption.

We stress that our system setup and ciphertext size are independent of the upper bound of the number

of supported users. Moreover, when $m \geq n+2$, the private keys of new users can be generated without requiring any update to other existing users' private keys. We assume that the additional new public parameters g_k (for $k \geq n+2$) can be distributed (one-off) to all users as part of key update information broadcast by the key authority or system update imposed by the system owner.

3.3.2 Forward-secure Decryption Keys

Our security model, as with that of Boldyreva et al.'s [24], considers exposure of only long-term private keys, but not decryption keys. Given a private key SK_{id} for a user with identity id, an adversary attacking our RIBE scheme can easily compute the corresponding decryption key $\mathsf{DK}_{id,t}$ associated with time t from SK_{id} and key update KU_t. If a private key query is made on a challenge identity id^*, we simply revoke id^* such that DK_{id^*,t^*} for challenge time t^* is no longer useful to the adversary. However, if the adversary is also allowed access to the decryption key of any user, such as that in a security model recently considered by Seo and Emura [107], we then require the decryption keys to be forward-secure. Otherwise, given a decryption key $\mathsf{DK}_{id^*,t}$ for $t \neq t^*$ and without making a private key query on the challenge id^* (implying id^* has still not been revoked at t^*), the adversary may be able to deduce the decryption key DK_{id^*,t^*} from $\mathsf{DK}_{id^*,t}$ and KU_{t^*}. Consequently, it will be trivial for the adversary to win the security game.

Recall that in our current RIBE scheme, a decryption key comprises

$$\langle K_1 = g^r R_3,\ K_2 = g^{\alpha}(u^{id}h)^r P_{\phi(id)}^{\beta} R'_3,\ K_3 = w_{\phi(id)} \rangle$$

where K_1, K_2 are fixed and K_3 is periodically updated. However, since K_3 can, in principle, be computed by anyone who has access to the public parameters, exposure of a decryption key $\mathsf{DK}_{id^*,t}$ for time $t \neq t^*$ can also lead to exposure of a decryption key $\mathsf{DK}_{id^*,t'}$ for any t', including $t' = t^*$. Clearly, the adversary can then trivially learn the challenge message.

Nevertheless, our RIBE scheme can naturally be extended to achieve forward-secure decryption keys using a 2-level Lewko and Waters HIBE scheme [86]. Particularly, we let level 1 keys be users' long-term private keys (associated with identities), and let level 2 keys be decryption keys (associated with times). For each time period, a user "delegates" a new, fully randomized decryption key with her

long-term private key. As shown in [107], re-randomization of decryption keys is sufficient to achieve the forward-secure property. In what follows, we sketch a modified version of our RIBE scheme that achieves forward-secure decryption keys.

- $\mathsf{Setup}(1^k, n) \to (\mathsf{PP}, \mathsf{MK}, \mathsf{RL}, \mathsf{ST}_\emptyset)$ As before. However, we require two random group elements $u_1, u_2 \in G_{p_1}$, instead of just u.

- $\mathsf{PriKeyGen}(\mathsf{PP}, \mathsf{MK}, id, \mathsf{ST}_U) \to (\mathsf{SK}_{id}, \mathsf{ST}_{U\cup\{i\}})$ As before. However, the private key SK_{id} now has an additional E_2 component:

$$K_1 = g^r R_3,\ K_2 = g^\alpha (u_1^{id} h)^r P_{\phi(id)}^\beta R_3',\ K_3 = w_{\phi(id)},\ E_2 = u_2^r R_3''$$

- $\mathsf{KeyUpd}(\mathsf{PP}, \mathsf{MK}, t, \mathsf{RL}, \mathsf{ST}_U) \to \mathsf{KU}_t$ As before.

- $\mathsf{DecKeyGen}(\mathsf{SK}_{id}, \mathsf{KU}_t) \to \mathsf{DK}_{id,t}$ As before. In addition, the algorithm chooses a random $r' \in \mathbb{Z}_N$ and random elements $\hat{R}_3, \hat{R}_3'$ of G_{p_3}. The decryption key is then set to be:

$$\begin{aligned}
K_1' &= K_1 g^{r'} \hat{R}_3 = g^{r+r'} \tilde{R}_3, \\
K_2' &= K_2 (u_1^{id} h)^{r'} (E_2)^t u_2^{r't} \hat{R}_3' = g^\alpha (u_1^{id} u_2^t h)^{r+r'} P_{\phi(id)}^\beta \tilde{R}_3', \\
K_3 &= w'_{\phi(id)}.
\end{aligned}$$

- $\mathsf{Enc}(\mathsf{PP}, M, id, \mathsf{AC}_V) \to \mathsf{CT}_{id,t}$ As before, except a small modification to C_1:

$$C = M \frac{e(g,g)^{\alpha s}}{\mathsf{z}^s},\ C_0 = g^s,\ C_1 = (u_1^{id} u_2^t h)^s,\ C_2 = (g^\beta \mathsf{AC}_V)^s.$$

- $\mathsf{Dec}(\mathsf{PP}, \mathsf{DK}_{id,t}, \mathsf{CT}_{id,t}) \to M$ As before.

- $\mathsf{KeyRev}(id, t, \mathsf{RL}, \mathsf{ST}_U) \to \mathsf{RL}$ As before.

The security proof of our modified RIBE scheme can be easily obtained by combining techniques used for our original scheme and those of [86]. The major difference in the security proof of this

extended scheme compare to the one in [86] is to inject the accumulator into the proof, and the technique is the same as the one we used in the basic scheme security proof presented in Section 3.2.4.

3.3.3 Revocable Attribute-Based Encryption

Attribute-based encryption (ABE) is a generalization of IBE and fuzzy IBE. In this subsection, we consider a variant of ABE called key-policy ABE (KP-ABE) [64], in which a message is encrypted under a set of descriptive attributes (as compared to just a single identity in the normal IBE setting), and a private key is associated with an access policy that specifies which kind of ciphertexts this particular private key is able to decrypt. A private key could decrypt a ciphertext that is associated with an attribute set only if the attribute set satisfies the access policy associated with the key.

Boldyreva et al. [24] sketched a construction of revocable KP-ABE using the binary tree method. Subsequently Sahai et al. [103] extended their idea and gave a complete construction with a security proof. Similarly, our accumulator-based revocation technique can be extended to the KP-ABE setting. Intuitively, we rely on an accumulator to capture all valid (non-revoked) attributes such that they can be represented with a single group element. Without loss of generality, we assume that an attribute can be a user identity. This way, we can revoke not only a common attribute shared among multiple users, but also a unique identity attribute to revoke a user. We now sketch a construction of revocable KP-ABE that is based on Lewko et al.'s adaptively secure KP-ABE scheme [84].

Let U, V, V_w be sets as defined before. Let n be the total number of attributes in the system. For simplicity, we assume that all attributes i are elements in $\mathbb{Z}_N$ and we let S denote the set of attributes under which a message will be encrypted.

- $\mathsf{Setup}(1^k, n) \rightarrow (\mathsf{PP}, \mathsf{MK}, \mathsf{RL}, \mathsf{ST}_\emptyset)$ As with that of our RIBE, except that it runs the setup algorithm of the scheme of [84]. The public parameters PP are $\langle N, g, g^\beta, e(g,g)^\alpha, \mathsf{z}, pk, T_i, \mathsf{AC}_\emptyset, P_1, \ldots, P_n, P_{n+2}, \ldots, P_{2n} \rangle$. The master secret key MK is $\langle \alpha, \beta, \gamma, sk \rangle$ and a generator X_3 of G_{p_3}.

- $\mathsf{Enc}(\mathsf{PP}, M, S, \mathsf{AC}_V) \rightarrow \mathsf{CT}$ As with that of our RIBE, except that given an attribute set S, the

ciphertext CT is set to be

$$C = M\frac{e(g,g)^{\alpha s}}{\mathsf{z}^s},\ C_0 = g^s,\ C_1 = (g^{\beta}\mathsf{AC}_V)^s,\ C_i = T_i^s$$

for all $i \in S$.

- $\mathsf{PriKeyGen}(\mathsf{PP}, \mathsf{MK}, (A, \rho)) \rightarrow \mathsf{SK}$ For each row x appears in an access matrix A, there is a corresponding attribute i such that $i = \rho(x)$.[5] The algorithm generates the witness for each $\rho(x) \in A$ and updates the accumulator and state as with that of our RIBE. To generate a private key, the algorithm chooses a random vector $\mathbf{u}$ such that $\mathbf{1} \cdot \mathbf{u} = \alpha$ (here, $\mathbf{1}$ denotes the vector with the first entry equals to 1 and the rest are all 0's), and a vector $\mathbf{v}$ such that the first entry is β while the other entries are all 0's. For each row A_x of A, it chooses a random $r_x \in \mathbb{Z}_N$, and random elements $W_x, V_x \in G_{p_3}$. The private key SK is then

$$K_x^1 = g^{A_x \cdot \mathbf{u}} T_{\rho(x)}^{r_x} P_{\rho(x)}^{A_x \cdot \mathbf{v}} W_x,\ K_x^2 = g^{r_x} V_x,\ K_x^3 = w_{\rho(x)}.$$

- $\mathsf{KeyUpd}(\mathsf{PP}, \mathsf{MK}, \mathsf{RL}, \mathsf{ST}_U) \rightarrow \mathsf{KU}$ As before, except that instead of adding/removing identities and times, the algorithm adds/removes attributes into/from the accumulator.

- $\mathsf{DecKeyGen}(\mathsf{SK}, \mathsf{KU}) \rightarrow \mathsf{DK}$ As before, but instead of updating the witness for just an identity, the algorithm updates the witness corresponding to each row of A.

- $\mathsf{Dec}(\mathsf{PP}, \mathsf{DK}, \mathsf{CT}) \rightarrow M$ If the attribute set S satisfies the access matrix A, the decryption algorithm computes constants ω_x such that $\sum_{\rho(x) \in S} \omega_x A_x = \mathbf{1}$. It then computes the blinding factor as

$$\prod_{\rho(x) \in S} \left(\frac{e(C_0, K_x^1)^{\omega_x}}{e(C_{\rho(x)}, K_x^2)^{\omega_x}} \cdot \frac{e(C_0, K_x^3)^{\omega_x A_x \cdot \mathbf{1}}}{e(P_{\rho(x)}, C_1)^{\omega_x A_x \cdot \mathbf{1}}} \right) = \frac{e(g,g)^{\alpha s}}{\mathsf{z}^s}.$$

- $\mathsf{KeyRev}(i, \mathsf{RL}, \mathsf{ST}_U) \rightarrow \mathsf{RL}$ As before.

We can prove the security of the above revocable KP-ABE using similar techniques as those for our RIBE scheme and those used in [84].

[5] Here ρ is a map from the row A_x of A to an index i.

We believe that similar techniques can be applied to obtain a revocable ciphertext-policy ABE (CP-ABE) scheme, another variant of ABE that reverses the properties of KP-ABE. That is, it encrypts a message with an access policy and generates a private key according to an attribute set.

3.4 Summary

In this chapter, we proposed a very efficient and adaptively secure RIBE scheme based on an accumulator. Our scheme enjoys constant-size key update, a major improvement from all previous RIBE schemes.

One immediate open problem would be to achieve adaptive security under more standard assumptions. Also, it would be interesting to investigate if our accumulator-based key update technique can be applied to revocable storage ABE proposed by Sahai et al. [103] and other variants of functional encryption.

Chapter 4

Near-Ideal Encrypted File Sharing At Scale Through Updatable Broadcast Encryption

Contents

In this chapter, we define the criteria for an ideal encrypted file sharing (EFS) system, then investigate, analyze and show current approaches have room for improvement. We then further propose a new primitive called updatable broadcast encryption (UBE), which could be used to achieve better efficiency for EFS systems. Through some novel techniques, we provide two concrete UBE constructions based on different broadcast encryption schemes which theoretically outperform existing approaches, and we further prove their security rigorously.

4.1 Introduction

4.1.1 Motivation

There has recently been proliferation of commercial encrypted file sharing tools, such as Boxcryptor [33], Cloudfogger [47], Viivo [114], DataLocker [50], and SkyCrypt [109]. They are typically designed to "piggyback" on major cloud storage services, such as Box, Dropbox, Google Drive, and Microsoft OneDrive. These tools are used to protect the confidentiality and integrity of files stored on the cloud (presumably untrusted), while providing access transparency to the user (in the sense that a cloud-based file system looks and feels like a local file system). Despite the commercial interest and the fact that encrypted file systems is a well-studied area, numerous challenges and open problems remain.

One of the important unaddressed challenges is supporting encrypted file sharing at a *large scale*. Here, by scale we mean three things—the number of users, the number of files, and the size of each file. The demand for such scalable, private sharing of files is imminent. For example, home furnishing retailers work primarily with computer-aided design (CAD) drawings. These files can be much larger than simple word documents. They need to be shared among architects, consultants, contractors, and engineers potentially spread across different geographic locations, and sometimes across organizations, in a frequent and timely manner [55]. Sharing of increasingly large files is also a common practice across other industries, such as: digital imaging and communications in medicine (DICOM) images in the healthcare industry; raw video files in media and advertising; and transaction reports and comparative performance data in financial services and banking [71, 72]. Within universities, sharing of (potentially large volume of) scientific data is often an integral part of collaborative research projects [102].

However, the majority of existing solutions do not cope well with large-scale, cloud-based enterprise file sharing. (We discuss the insufficiency of existing solutions in Section 4.1.2.)

Table 4.1: Properties achieved by existing cryptographic file systems.

	Plutus [80]	SiRiUS [62]	BE [28]	PRE [6]	ABE&PRE [118]	Boxcryptor [33]
File header size	$O(n)$	$O(n)$	$O(1)$	$O(1)^{\dagger}$	$O(\tilde{n})$	$O(n)$
Revocation cost:						
– file owner	$O(n)$	$O(n)$	$O(1)$	$O(1)$	$O(\tilde{n})$	$O(n)$
– server	–	–	$O(1)$	$O(n)$	$O(n)$	–
– recipient	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(\tilde{n})$	$O(1)$
– no re-upload of content	✗	✗	✗	✗	✗	✗
Cross-organization sharing	✓	✓	✗	✓	✓	✓
Key escrow-free	✓	✓	✗	✓	✓	✓
Cloud-based	✗	✗	✗	✗	✓	✓

Notation: n denotes number of recipients, $\tilde{n}$ denotes number of attributes; † amortized cost.

In this chapter, we first define a set of *ideal properties* associated with the efficiency, usability, and security of an encrypted cloud-base file-sharing system. First, we seek an encrypted file-sharing system that requires minimal effort in sharing a file, i.e., upload cost of the file header (containing the corresponding file key) is independent of the number of recipients. Second, it should be simple and cheap to revoke arbitrary (and large) subsets of recipients in terms of computational and communication overhead. For example, solutions that require data exchange proportional to the number of non-revoked users for revoking a single user are deemed unacceptable. Third, an ideal solution should allow cross-organization sharing between users from different organizations or departments, without relying on a global or centralized trusted key authority. To this end, we deem a solution that requires nothing beyond possessing a long-term, self-generated public-private key pair as acceptable.[1] We elaborate on several other desirable properties in Section 4.1.2. We observe that, despite the well-established importance of the problem, current commercial solutions and research proposals fail to simultaneously achieve our defined set of ideal properties. Table 4.1 explains the properties achieved by existing solutions at a glance; we refer the reader to Section 4.1.2 & Section 4.1.3 for further explanation.

[1]Such systems are already in use (e.g., PGP) and if combined with identity-providing mechanisms, such as OpenID [49], in the future, public key sharing could be made easier.

4.1.2 Problem & Challenges

We investigate the possibility of achieving an ideal encrypted file system (EFS) that satisfies a set of desirable properties. We show why, in practice, it seems infeasible to realize an ideal EFS with current technology.

Problem Definition. We consider a scenario where a file owner encrypts and stores her files on a remote, untrusted platform operated by a cloud storage service provider. Moreover, the file owner may share some of her encrypted files with a group of intended recipients. Encryption files are shared with intended users but must never be made accessible to a third-party or the cloud provider. We assume that a file comprises two components: (i) *header*, containing one or multiple encryptions of a file key, and (ii) *body*, containing the content of the file encrypted with the file key. We also assume that all system users, i.e., both the file owner and the recipients, own a self-generated long-term public-private key pair. The public keys can be made available to all users via a public key server. We then focus on devising an encrypted cloud-based file system that satisfies the following properties:

P1–Confidentiality & integrity: The confidentiality and integrity of any encrypted content should be protected against any party who is not an intended recipient, including the cloud provider.

P2–Minimal upload cost: The size of the file header containing key material required by each recipient to read the corresponding content should be independent of the total number of recipients, i.e., with complexity of $O(1)$. However, clearly, the overall initial upload cost for the entire file would be proportionate to the size of the file body.

P3–Minimal download cost: Similarly, the initial download cost is inevitably proportionate to the size of the entire encrypted file. However, any subsequent access to a modified/updated version of the same file should not require re-downloading of the whole file but only the changes.

P4–Minimal revocation cost: The computational and communication overhead incurred by the file owner for revoking an existing recipient should have complexity of $O(1)$ only, i.e., independent of the total number of recipients. Moreover, the file owner is not required to re-upload the content (encrypted under a new file key) to the cloud.

P5–Key escrow-freeness: The file key used to protect the content of a shared file should be known only to the file owner and the intended recipients.

P6–Cross-organization sharing: The file owner should be able to share a file with any user from within the same or outside her organization, simply by relying on any existing identity provider (IdP).

P7–Minimal key storage: All system users should locally store and manage only a pair of self-generated, long-term public-private key. All other ephemeral or intermediate cryptographic keys used during file sharing should be transparent to the users.

Insufficiency of Existing Solutions. The majority of current commercial tools, such as Boxcryptor and Cloudfogger, rely on a simple combination of symmetric and asymmetric encryption techniques (henceforth, we refer to this as the basic approach). That is, one encrypts a file with a symmetric file key k (e.g., AES) and then places an asymmetric encryption (e.g., RSA) of k, under the recipient's public key, in the file header. The file header and the encrypted content are uploaded and stored in a storage server. If the file is shared among n recipients, then k needs to be encrypted under the public keys of all the n recipients. Hence, this results $n + 1$ ciphertexts in the file header. Clearly, such a solution does not scale for large numbers of recipients. Moreover, to revoke any of the recipient, i.e., preventing the recipient from reading any future update to the file, the file owner essentially has to repeat the same steps as before. That is, she has to generate a new file key k', re-encrypt it under all the public keys of the remaining non-revoked recipients, and re-upload the entire file to the storage server. In fact, earlier notable encrypted file systems, such as SiRiUS [62], Plutus [80], and Windows EFS [111], also employ the described basic approach.

Why is the Ideal EFS Infeasible? More recent academic research largely focuses on designing and applying advanced cryptographic primitives, such as broadcast encryption (BE) [28, 85], proxy re-encryption (PRE) [6], and attribute-based encryption (ABE) [64, 100], to improve various aspects of the basic approach. Among these, reducing the file header size and simplifying key management, particularly key distribution and revocation, are two important directions. Nevertheless, a large part of the recent advancement in cryptographic techniques cannot be easily translated into practice. This is because there exist fundamental tension and trade-offs between efficiency (**P2–P4**), key escrow-freeness

(**P5**), and usability (**P7**).

While the development in BE techniques is very promising, they are not the panacea for a large-scale file sharing system. The seminal work by Boneh et al. [28] enables "compression" of ciphertext components for multiple recipients into a compact ciphertext. Hence, BE becomes useful in our context for distributing a file key K_F to a large number of recipients, while keeping the file header size very small. However, BE requires a trusted key authority possessing a master secret key to derive and distribute private keys to all system users. This implies key escrow, which may not be desired for some application scenarios. One standard way to mitigate the risk of a single trusted key authority is to use *threshold key generation*.[2] Hence, one can avoid fully trusting a single authority, assuming at least one of the authorities does not collude with the others. However, key distribution may yet be another challenge in terms of usability and practicality. This is because each user must now securely obtain multiple keys from different authorities. This, in turn, implies the need for global trusted key generation authorities—something known to be very difficult to realize in practice [54, 101]. This clearly shows the tension between efficiency, usability, and key escrow-freeness.

We note that, in principle, it may be possible to derive an escrow-free BE scheme from the combination of escrow-free identity-based encryption (IBE) [44] and multi-receiver IBE [14], for example. However, the resulting scheme will always have ciphertext size that is linear (or sublinear) to the number of recipients, even if randomness reuse techniques (see for example [19]) are employed.

There exists other alternatives based on ABE and PRE (or combination of both). However, as shown in Table 4.1, these solutions come with different trade-offs and cannot simultaneously satisfy all the ideal properties. (We discuss these alternatives in Section 4.1.3.)

We therefore infer that it is infeasible to design the ideal EFS thus far.

4.1.3 Related Work

Early Cryptographic File Systems. Many cryptographic file systems have been proposed in the past, for example, cryptographic file system (CFS) [23], Cryptfs [119], transparent cryptographic file system [37], and secure network attached disks (SNAD) [91], Plutus [80], and SiRiUS [62]. Most of these

[2] Briefly speaking, a master key (or one of its components) is split into multiple shares, which in turn, are distributed to multiple key authorities. See [28] for further details.

are Unix-based distributed file systems (or known as network file systems) which allow a system user access files over a network much like local storage is accessed via a network protocol. On the other hand, our work focuses on web-based file sharing systems which interoperate with existing cloud storage services.

Broadcast Encryption. The emergence of BE that produces ciphertexts of constant size (and uses small private keys), see for example [28, 51, 85, 98], is seen as an appealing solution to encrypted file sharing. However, a BE-based solution typically assumes the existence of a trusted key authority, as discussed in Section 4.1.2. Hence, conventional BE seems to be more suitable for file sharing within an organization and for an environment where key escrow is acceptable.

Proxy Re-Encryption. As described in Section 4.2.1, PRE [77, 6] allows an untrusted proxy, when given a re-encryption key, to transform a ciphertext generated by the file owner into one that is decryptable by an intended recipient. This enables the file owner to distribute a file key without revealing it to a third party, and thus, addresses the key escrow problem in BE. However, in return, whenever a recipient is revoked, the proxy must re-encrypt a new encrypted file key (generated by the file owner) under each remaining non-revoked recipient. Further, existing PRE schemes do not support ciphertext update, and a PRE-based file system requires re-upload of the entire encrypted content under the new file key.

Attribute-Based Encryption. More fine-grained cryptographic access control can be achieved through attribute-based encryption (ABE) [64, 100, 84]. In ABE, each file is encrypted under a set of attributes, such that a recipient can decrypt properly only if the attribute set satisfies the access structure defined by her private key. This allows more fine-grained access control over encrypted shared files compared to other primitives such as BE and other conventional public key encryption. However, as with BE, a trusted third party is required to generate and distribute keys. Further, key revocation is non-trivial in the ABE setting. Sahai et al. [103] proposed the concept of revocable-storage ABE (RS-ABE), which allows access revocation to be performed by a third party via ciphertext delegation (involving both ciphertext and key update). More recent results by Lee et al. [83] show that RS-ABE ciphertexts are still prohibitively large for practical usage.

Others. Chu et al. [46] proposed a variant of PRE called proxy broadcast re-encryption, which allows a re-encrypted ciphertext to be broadcast to a group of recipients. However, the use of BE techniques

inevitably inherit similar shortcomings to those of BE-based systems.

On the other hand, Yu et al. [118] proposed a hybrid ABE & PRE approach for secure file sharing. As with ours, they let the file owner generate the ABE master secret and all the private keys required by the file recipients, and thus, eliminating key escrow. However, revoking ABE keys are complicated. During each revocation, the file owner must update all remaining non-revoked recipients' private keys, a non-trivial task which involves complex key management operations.

Last but not least, there are many other important related but somewhat orthogonal works on other aspects of encrypted file systems, for example, cloud storage architecture [81], proofs of retrievability [79], proofs of data possession [5], proofs of ownership [67], key aggregation for multi-file sharing [45], and so on.

4.1.4 Our Approach

As mentioned earlier, although broadcast encryption bears certain disadvantages which contradict to our ideal EFS properties, it is still the most relevant primitive which best suits our EFS model. Thus it is natural to start exploring the various BE schemes.

Our starting point is Lewko et al.'s public key BE scheme [85]. We work with this scheme because it uses only a small, constant number of public parameters PP. (This is in contrast to most of the other BE schemes, e.g., Boneh et al.'s [28], where the size of public parameters grow linearly with the number of users.) We use Lewko et al.'s BE scheme to protect the confidentiality of the file key (header) and the content (body) associated with a file (**P1**). This BE scheme employs a *key encapsulation mechanism* (KEM), which simultaneously generates a random (symmetric) file key k to protect the file content, and encrypts k using a public-key BE technique. The latter allows k to be recovered by a group of intended recipients from a BE ciphertext. The initial ciphertext size (when no real file recipient has been revoked yet) of the scheme is small, i.e., less than a hundred bytes. This is consistent with the requirement of small header sizes (**P2**). (We note in passing that data integrity can be protected using the standard method, e.g., computing a MAC with a key derived from the file key. Hence, we will not discuss this further in the remainder of the chapter.)

We then extend Lewko et al.'s BE scheme such that it becomes "updatable", i.e., a BE ciphertext can

be re-encrypted without revealing the underlying plaintext. This allows simple user revocation, since the file owner can now simply update the encrypted content such that a revoked recipient can no longer decrypt properly. Moreover, the file owner is not required to re-encrypt and re-upload the entire content using a new file key (**P4**). We achieve this in two steps:

- We apply a simple ciphertext re-randomization technique (also shown in [28, 103]). All the file owner needs to do is to pick a random value and generate three BE ciphertext components (also less than a hundred bytes) and upload them to the server. The cloud-based server then updates the initial uploaded BE ciphertext with the new ciphertext components. From the updated BE ciphertext, any non-revoked recipient is able to derive a new file key k' (**P3**).

- We also rely on symmetric-key proxy re-encryption encryption[3] (SK-PRE) [31] to let the cloud provider homomorphically update the encrypted content. With SK-PRE, the file owner computes and submits a re-encryption key Δ_k to the server. The server then re-encrypts the original encrypted content with Δ_k, such that the content are now effectively encrypted under the new file key k'.

In our first scheme (based on [85]), the file owner (instead of the server) acts as the key authority in order to eliminate the key escrow issue discussed before (**P5**). However, in return, she must now bear the computational and communication cost for generating and distributing BE private keys to all her intended file recipients. The overhead can be significant for a large group of recipients. We minimize such overhead by outsourcing BE key generation and distribution to the server without leaking any information about the BE keys.

Briefly speaking, the file owner first chooses a BE master secret key MS, which is known only to herself, and the corresponding public parameters PP. She also generates a *blinded secret key* BS (by blinding MS with a random value[4]) and a *transformation key* TK. Here BS is used by the server to generate and distribute a BE private key SK to each recipient. However, SK obtained from the server can only be used to recover a "blinded" version of the content shared by the file owner (otherwise,

[3] SK-PRE is analogous to PK-PRE in the symmetric-key setting. Such a primitive allows an untrusted third party to update the encryption key associated with a ciphertext without learning the underlying plaintext. See Section 4.2.1 for a formal definition.

[4] A similar blinding technique was also used by Green et al. [65] in their proposal for outsourcing decryption of ABE ciphertexts to the cloud.

clearly, the server would be able to decrypt any file owner's encrypted file). Each intended recipient also requires TK, which can be regarded as a second decryption key component, in order to transform the blinded content into the original plaintext form.

What is left now is to efficiently distribute TK to only the intended recipients. Using BE is possible, but then it defeats the purpose for letting the file owner be the key authority and outsourcing key generation to the server. Instead, we make use of public-key proxy re-encryption (PK-PRE) [6]. This is, we let the server be a proxy, whose role is to re-encrypt the encryption of TK, such that the resulting ciphertext is decryptable by the intended recipient. To realize this, the file owner must also generate a re-encryption key rk for each new recipient with whom she has not previously shared a file. As discussed, this is a one-time process and can be pre-computed (**P2**). All the required BS, TK, and rk values (all in encrypted) are part of the metadata file, which is uploaded to the key storage.

Our first construction based on Lewko et al.'s BE scheme is "near-ideal", that is, it achieves all the ideal properties as defined but one. As mentioned above, our approach does require a non-constant communication cost when a file is shared with recipients with whom no previous files were shared. That is, it requires a *one-time* generation and distribution of a PK-PRE key for every new recipient. When files are repeatedly shared with a user, or if the PK-PRE keys can be pre-computed for a subset of users, the cost can be offloaded to a one-time cost, thus achieving a $O(1)$ amortized cost.[5]

Although the first scheme achieves a near-ideal construction, one potential drawback of this approach is the proxy re-encryption technique used (both symmetric and public) may potentially lower down the real efficiency of the scheme. Thus served as a stepping stone, we continue investigating another variant of BE schemes called inclusive-BE, among which the construction given by Boneh et al. [28] is the most efficient. Inclusive-BE allows an encryptor to specify a set of *legitimate* recipients during file encryption (where the name "inclusive" comes from). This is in contrast with its dual form "exclusive-BE" (like Lewko et al.'s scheme [85]) which specifies a set of "revoked users" during encryption, and anyone who possesses a legitimate decryption key and not in the encryption set could decrypt the ciphertext successfully.

The greatest advantage of Boneh et al.'s BE scheme is that its BE ciphertext (or the header) is at

[5]The file upload cost incurred with our approach is still independent of the number of recipients in an amortized sense, i.e., the initial key distribution cost is written off over time.

constant size, regardless of the number of recipients. However, the downside of the base scheme comes from its public parameter size, which is linear w.r.t the number of users in the system. This has become the bottleneck of our exploration based on this scheme. Imagine the sheer computational cost occurred at the file owner side if she wishes to share a file with potentially hundred or thousands of recipients. Moreover, the public parameter creation involves group element exponentiation which is considered as computationally heavy and time consuming. On the other hand, we couldn't delegate this public parameter generation process directly and wholly to the server, as this process involves generation of the system-wide master secret, and a straightforward delegation to the server will compromise the security of each single file.

To overcome this obstacle, we take a novel approach that separates the generation of master secret key: we let the server (which is considered as having sufficient computational power) generate the (linear size many) group elements as public parameter PP, and associates the PP with a system-wide server master secret key MS_s. This PP could potentially accommodate, say millions of users. This is a one-time process and the size of this PP for 10 million users is roughly 400MB. However, for each file owner who uses this cloud-based server service, her intended recipients will definitely smaller than the system-wide number of users, thus her recipients does not necessarily need to download the whole PP, instead only those elements related to a certain file. A quick estimation measures the size of PP needed for a file shared with 1000 recipients is only around 40KB.

At the same time, the file owner will choose her owner master secret key MS_o. This is kept secret by herself. Together with the PP generated by the server, the file owner generates her recipient's decryption key using PP and MS_o (recall PP is associated with MS_s). With this owner master secret key hidden from the server, and embedded into recipients' decryption key, the server could not learn anything of the encrypted file.

For content update, instead of the symmetric-key proxy re-encryption technique, we employ a symmetric-key double encryption primitive to fit our construction. More details are given in the concrete construction.

The trade-off of our second construction based on Boneh et al.'s scheme comes to the distribution of recipient decryption key, and which make this approach also near-ideal. As mentioned, the file owner

needs to generate her own master secret key and embed it into the decryption keys, without using the public-key proxy re-encryption method, the file owner is required to encrypt this decryption key under each recipient's long-term public key. However, this is a one-time cost during initial setup, which is reasonable.

4.1.5 Outline

The rest of this chapter is organized as follow: in Section 4.2 we first define the concept of updatable broadcast encryption, then describe the building blocks used during construction, followed by a concrete scheme based on Lewko et al.'s exclusive-BE scheme and its security proofs. In Section 4.3, we provide the second construction based on Boneh et al.'s inclusive-BE scheme and also its security analysis. We summarise this chapter in Section 4.4.

4.2 Updatable Broadcast Encryption - Version 1

We are now ready to describe our first UBE scheme, which is an extension of Lewko et al.'s BE scheme [85]. We note that using Lewko et al.'s BE scheme for our purpose has two constraints: (i) it is a revocation system (also known as exclusive BE), which can only encrypt a message by specifying a set of revoked users; and (ii) the ciphertext size grows linearly in the number of revoked users. We side step the first constraint by revoking a dummy user when a file owner first encrypts a shared file. The second constraint does not pose any usability and efficiency concern either (w.r.t. the user). This is because the encrypted file has already been uploaded to the file server before any revocation takes place.

4.2.1 Building Blocks

Our first scheme makes use of two additional primitives defined below. For simplicity, we leave out the full details and use them in a black-box manner in our construction (described in Section 4.2.2).

Intuitively, we make use of public-key proxy re-encryption (PK-PRE) for key distribution between a file owner and her recipient set. Particularly, the file owner generates and encrypts a transformation key under her own public key and forwards it to the cloud provider, which in turn, re-encrypts the transformation key under each recipient's public key. PK-PRE ensures that the cloud provider learns

nothing about the transformation key, while allowing the re-encrypted transformation key is decryptable by only the intended recipient.

A PK-PRE scheme comprises the following algorithms [6]:

- PKGen(λ) $\rightarrow (pk, sk)$: The key generation algorithm takes as input a security parameter λ and outputs a public-private key pair (pk, sk).
- PKReGen(pk_A, sk_A, pk_B) $\rightarrow rk_{A \rightarrow B}$: The re-encryption key generation algorithm that takes as input the delegator's public-private key pair (pk_A, sk_A) and the delegatee's public key pk_B, and outputs a re-encryption key $rk_{A \rightarrow B}$.
- PKEnc(pk, m) $\rightarrow U$: The encryption algorithm takes as input a public key pk and a message m, and outputs the corresponding ciphertext U.
- PKReEnc($U_A, rk_{A \rightarrow B}$) $\rightarrow U_B$: The re-encryption algorithm takes as input a ciphertext U_A associated with A and a re-encryption key $rk_{A \rightarrow B}$. It outputs a ciphertext U_B that is decryptable by B.
- PKDec(sk, U) $\rightarrow m$: The decryption algorithm takes as input a private key sk and a ciphertext U, and outputs the corresponding message m if the key sk is correct.

On the other hand, we rely on symmetric-key proxy re-encryption (SK-PRE) to encrypt the content of a shared file and allow update to the encryption of the content. This is achieved by re-encrypting the content using key update material that can only be generated by the file owner.

An SK-PRE scheme consists of the following algorithms [31]:

- SKGen(λ) $\rightarrow k$: The key generation algorithm takes as input a security parameter λ and outputs a secret key k.
- SKEnc(k, m) $\rightarrow V$: The encryption algorithm takes as input a secret key k and a message m, and outputs the corresponding ciphertext V.
- SKReGen(k) $\rightarrow \Delta_k$: The re-encryption key generation algorithm takes as input a secret key k and outputs the corresponding key update Δ_k.

- $\mathsf{SKReEnc}(\Delta_k, V) \rightarrow V'$: The re-encryption algorithm takes as input a ciphertext V associated with key k and key update Δ_k. It outputs an updated ciphertext V' under a new key k'.

- $\mathsf{SKDec}(k, V) \rightarrow m$: The decryption algorithm takes as input a ciphertext V and a secret key k, and outputs the corresponding message m if the key k is correct.

4.2.2 Construction

We first define the new primitive: updatable broadcast encryption.

Definition. Let ID_i be the identity of user i and (pk_i, sk_i) denote the PK-PRE key pair generated from PKGen. (the key pair is stored in the identity key server.) Let (pk_o, sk_o) denote a file owner's key pair and (pk_s, sk_s) the file server's key pair. We then let R be a set of intended file recipients and $R_0 \subseteq R$ be a set of new recipients (with whom the file owner has never shared a file before). We also let S be a set of revoked recipients and it is initialized as $S = \{\mathrm{ID}_0\}$ with a dummy identity.

An updatable broadcast encryption (UBE) system consists of the following seven algorithms:

- $\mathsf{Setup}(\lambda) \rightarrow (\mathsf{PP}, \mathsf{MS})$: The key setup algorithm takes as input a security parameter λ and outputs a public parameter set PP and a master secret key MS.

- $\mathsf{TrapGen}(\mathsf{PP}, \mathsf{MS}, R_0, pk_s, pk_o, sk_o, pk_i) \rightarrow \mathsf{KM}$: The trapdoor generation algorithm takes as input the required key set and outputs the key material KM, containing the blinded secret BS, transformation key TK, and the necessary re-encryption keys $rk_{o \rightarrow i}$ for all $i \in R_0$.

- $\mathsf{Enc}(S, \mathsf{PP}, M) \rightarrow (\mathsf{CT}, V)$: The encryption algorithm takes as input the content M of a shared file and outputs the BE ciphertext CT and encrypted content V (using SKEnc).

- $\mathsf{OutGen}(\mathsf{KM}, sk_s, R, \mathrm{ID}_i, pk_i) \rightarrow \tilde{\mathsf{KM}}$: The outsourced key generation algorithm takes as input the key material KM. It outputs re-encrypted key material $\tilde{\mathsf{KM}}$ (using $\mathsf{PKReEnc}$), containing the BE private key SK_i and transformation key TK for each recipient $i \in R$.

- $\mathsf{RevUser}(S, \mathrm{ID}_i, \mathsf{PP}, k, \mathsf{CT}, V) \rightarrow (S', \mathsf{CT}', V')$: The revocation algorithm takes as input the revoked recipient's identity ID_i and the current ciphertext components (CT, V). It outputs updated ciphertext CT' and updated encrypted content V' (using $\mathsf{SKReEnc}$).

- $\mathsf{AddUser}(\mathsf{PP}, R_0, R, \mathrm{ID}_i, pk_i, pk_o, sk_o, sk_s) \rightarrow (R', \tilde{\mathsf{KM}})$: The addition algorithm takes as input a recipient's identity ID_i and outputs updated key material $\tilde{\mathsf{KM}}$.

- $\mathsf{Dec}(S, \mathsf{CT}, \tilde{\mathsf{KM}}, V, sk_i) \rightarrow M$: The decryption algorithm takes as input the ciphertext components (CT, V) and key material $\tilde{\mathsf{KM}}$, and outputs the content M if $\mathrm{ID}_i \notin S$.

The Setup, Enc, Dec algorithms are based on those of Lewko et al.'s scheme [85], while the others are new algorithms designed to meet the requirements of our scenario. All the above algorithms but OutGen are performed by the file owner. The cloud provider runs the OutGen algorithm to generate and distribute BE private keys to all the file recipients; and jointly runs $\mathsf{RevUser}$ and $\mathsf{AddUser}$ with the file owner to revoke and add a recipient, respectively. Each recipient runs Dec to access to the shared content.

Construction. We makes use a bilinear group $\mathbb{G}$ of prime order p. We assume that all identities are taken from the set $\mathbb{Z}_p$.[6] A file owner runs the following scheme each time she wants to share a file:

- $\mathsf{Setup}(\lambda) \rightarrow (\mathsf{PP}, \mathsf{MS})$: The algorithm picks random group generators $g, h \in \mathbb{G}$ and random values $\alpha, b, z \in \mathbb{Z}_p$. It also chooses a function $H : \mathbb{G}_T \rightarrow \mathbb{Z}_p$. The public parameters are

$$\mathsf{PP} = (g, g^b, g^{b^2}, h, h^b, e(g,g)^\alpha, H).$$

  The master secret key is $\mathsf{MS} = (\alpha, b, z)$. We note that new α and z values must be chosen for each shared file. The other parameters can be fixed and reused across different files.

- $\mathsf{TrapGen}(\mathsf{PP}, \mathsf{MS}, R_0, pk_s, pk_o, sk_o, pk_i) \rightarrow \mathsf{KM}$: The algorithm first computes the blinded master secret $\mathsf{BS} = g^{\alpha z}$ and the transformation key $\mathsf{TK} = z^{-1}$. It then performs the following steps:

  1. compute $U_s = \mathsf{PKEnc}(pk_s, \mathsf{BS})$;
  2. compute $U_o = \mathsf{PKEnc}(pk_o, \mathsf{TK})$;
  3. for each recipient $i \in R_0$, generate

$$rk_{o \rightarrow i} = \mathsf{PKReGen}(pk_o, sk_o, pk_i)$$

[6] In practice, we could use a collision resistant hash function mapping identity strings to $\mathbb{Z}_p$.

and compute $U_i = \mathsf{PKEnc}(pk_s, rk_{o\to i})$;

4. output $\mathsf{KM} = (U_s, U_o, U_1, \ldots, U_{|R_0|})$.

(KM is saved in the metadata file and uploaded to the key storage.)

- $\mathsf{Enc}(S, \mathsf{PP}, M) \to (\mathsf{CT}, V)$: The algorithm picks a random $s \in \mathbb{Z}_p$ and sets the ciphertext CT_0 as:

$$C_0 = g^s,\ C_{0,1} = g^{b\cdot s},\ C_{0,2} = (g^{b^2\cdot \mathrm{ID}_0} h^b)^s.$$

The algorithm also computes the encapsulated key $\mathsf{EK} = e(g,g)^{\alpha s}$, from which it derives a file key $k = H(\mathsf{EK})$ and sets $V = \mathsf{SKEnc}(k, M)$ as the encrypted content.[7] (Recall that CT is in the file header, while V is in the body. They are stored on the file server.)

- $\mathsf{OutGen}(\mathsf{KM}, sk_s, R, \mathrm{ID}_i, pk_i) \to \tilde{\mathsf{KM}}$: The algorithm first recovers BS and $rk_{o\to i}$ from U_s and U_i, respectively (for all $i \in R_0$) using private key sk_s. For each recipient $i \in R$, it then performs the following steps:

1. choose random $t \in \mathbb{Z}_p$, set the BE private key as

$$\mathsf{SK}_i = (D_0, D_1, D_2) = (g^{\alpha z} g^{b^2 t},\ (g^{b\cdot \mathrm{ID}_i} h)^t,\ g^{-t})$$

and compute $\tilde{U}_i = \mathsf{PKEnc}(pk_i, \mathsf{SK}_i)$;

2. compute $U_i' = \mathsf{PKReEnc}(U_o, rk_{o\to i})$, that is, re-encrypt U_o with the recipient's re-encryption key.

It sets $\tilde{\mathsf{KM}} = (\tilde{U}_i, U_i')$ and stores it on the key storage.

- $\mathsf{RevUser}(S, \mathrm{ID}_i, \mathsf{PP}, k, \mathsf{CT}, V) \to (S', \mathsf{CT}', V')$: The algorithm first adds the revoked recipient ID_i to S, that is, $S' = S \cup \{i\}$. It picks a random $s_i \in \mathbb{Z}_p$,[8] and computes the new encapsulated key as $\mathsf{EK}' = e(g,g)^{\alpha(s+s_i)}$. It then performs the following steps:

[7] We assume that the content M of a file is divided into message blocks and encrypted at the block level.

[8] To minimize storage of all s_i values used for revocation, the file owner can derive each s_i from a master seed and a pseudorandom function.

1. compute $k' = H(\mathsf{EK}')$ and key update $\Delta_k = -k \oplus k'$ for some operator $\oplus$ (depending on the SK-PRE scheme used), and encrypt it as $U_\Delta = \mathsf{PKEnc}(pk_s, \Delta_k)$;
2. set

$$C_0' = g^{s+s_i},\ C_{i,1} = g^{b \cdot s_i},\ C_{i,2} = (g^{b^2 \cdot \mathrm{ID}_i} h^b)^{s_i};$$

The above steps are all performed by the file owner, who then submits $(S', U_\Delta, C_0', C_{i,1}, C_{i,2})$ to the cloud, which in turn, performs the following steps:

1. recover $\Delta_k = \mathsf{PKDec}(sk_s, U_\Delta)$, update the encrypted content V as $V' = \mathsf{SKReEnc}(\Delta_k, V)$;
2. replace C_0 by C_0' and append $C_{i,1}, C_{i,2}$ to CT;
3. replace S by S'.

- $\mathsf{AddUser}(\mathsf{PP}, R_0, R, \mathrm{ID}_i, pk_i, pk_o, sk_o, sk_s) \rightarrow (R', \tilde{\mathsf{KM}})$: Adding a recipient i to the current set R is simple. The algorithm adds i to the recipient set R and runs the $\mathsf{TrapGen}$ and OutGen algorithms to add the corresponding $(\tilde{U}_i, U_i')$ components to $\tilde{\mathsf{KM}}$. However, no update to the encrypted content V is required.

  It is also straightforward to add a previously revoked recipient. This is done by "reversing" the revocation process, i.e., removing the associated s_i value from the ciphertext CT.

- $\mathsf{Dec}(S, \mathsf{CT}, \tilde{\mathsf{KM}}, V, sk_i) \rightarrow M$: The algorithm first recovers SK_i and z^{-1} from $\tilde{U}_i$ and U_i', respectively using private key sk_i. Let $r = |S|$. If $i \in S$, the algorithm aborts; otherwise, it computes

$$A = \frac{e(C_0, D_0)}{e(D_1, \prod_{j=1}^{r} C_{j,1}^{\frac{1}{\mathrm{ID}_i - \mathrm{ID}_j}}) \cdot e(D_2, \prod_{j=1}^{r} C_{j,2}^{\frac{1}{\mathrm{ID}_i - \mathrm{ID}_j}})}$$

  resulting in $A = e(g,g)^{\alpha s z}$. With z^{-1}, it is straightforward then to compute the file key k and recover the content M from V.

Correctness. We verify the correctness of the Dec algorithm of our UBE scheme:

$$
\begin{aligned}
A &= \frac{e(C_1, D_0)}{e(D_1, \prod_{j=1}^{r} C_{j,1}^{\frac{1}{\mathrm{ID}_i - \mathrm{ID}_j}}) \cdot e(D_2, \prod_{j=1}^{r} C_{j,2}^{\frac{1}{\mathrm{ID}_i - \mathrm{ID}_j}})} \\
&= \frac{e(g^s, g^{\alpha z} g^{b^2 t})}{e((g^{b \cdot \mathrm{ID}_i} h)^t, \prod_{j=1}^{r} g^{\frac{b \cdot s_j}{\mathrm{ID}_i - \mathrm{ID}_j}}) \cdot e(g^{-t}, \prod_{j=1}^{r} (g^{b^2 \cdot \mathrm{ID}_j} h^b)^{\frac{s_j}{\mathrm{ID}_i - \mathrm{ID}_j}})} \\
&= \frac{e(g,g)^{\alpha s z} \cdot e(g,g)^{s b^2 t}}{\prod_{j=1}^{r} (e(g,g)^{\frac{b^2 \mathrm{ID}_i t s_j}{\mathrm{ID}_i - \mathrm{ID}_j}} \cdot e(h,g)^{\frac{t b s_j}{\mathrm{ID}_i - \mathrm{ID}_j}})} \times \frac{e(g,g)^{\alpha s z} \cdot e(g,g)^{s b^2 t}}{\prod_{j=1}^{r} (e(g,g)^{\frac{-t b^2 \mathrm{ID}_j s_j}{\mathrm{ID}_i - \mathrm{ID}_j}} \cdot e(g,h)^{\frac{-t b s_j}{\mathrm{ID}_i - \mathrm{ID}_j}})} \\
&= \frac{e(g,g)^{\alpha s z} \cdot e(g,g)^{s b^2 t}}{\prod_{j=1}^{r} e(g,g)^{b^2 t s_j}} \\
&= e(g,g)^{\alpha s z}
\end{aligned}
$$

4.2.3 Security Analysis

We consider a standard chosen-plaintext attack (CPA) model in the BE setting [28, 85]. In our scheme, we further consider two types of probabilistic polynomial-time (PPT) adversaries:

- Type 1: malicious users, i.e., unintended (including revoked) file recipients, who are interested to access to the content of a shared file;
- Type 2: malicious cloud provider, who is interested to learn the content of the shared file from interacting with the file owner or the recipient.

To simulate a Type 1 adversary, we provide it access to two types of oracles: the BE private key (SK) extract oracle and the transformation key (TK) extract oracle. However, we consider a selective security model, where the adversary must first declare the revoked set S^* of users before given the PP and issues any query.[9] It is then given the public parameter set PP, and subsequently allowed to query the $(\mathsf{SK}_i, \mathsf{TK})$ values for any user $i \in S^*$ to these oracles in order to compute the corresponding file key k. (Here TK is fixed for any i in the simulation.) During the challenge phase, as usual, the adversary submits two files M_0 and M_1, where $|M_0| = |M_1|$, and is returned the encryption of M_β, i.e., the

[9] We achieve selective security against Type 1 adversary as our scheme is based on Lewko et al.'s selectively secure BE scheme [85].

ciphertext components (CT^*, V^*) associated with S^*, for a random bit $b \in \{0, 1\}$. The adversary wins the security game if it correctly guesses β.

On the other hand, to model a Type 2 adversary, we provide it access to the file key (k) extract oracle. (Since the cloud provider can generate any SK value for any user i from the blinded secret BS, clearly, no meaningful security can be achieved if it also learns the TK value.) It is allowed access to any file key associated with any revoked set S' in an adaptive manner (since a new file key is chosen each time a user is revoked), and subsequently we could achieve adaptive security against Type 2 adversary.[10] This is to reflect the scenario where the compromise of any file key k at any point during file sharing would not affect the confidentiality of the content after it has been updated under a new file key k'. The Type 2 adversary is also allowed to query the transformation key $\mathsf{TK} = z^{-1}$ for any non-challenge file. However, as this TK is refreshed for every file, by given out the non-challenge file TK won't affect the security of the challenged file. For simplicity we only focus on the challenge file and omit this transformation key query in the security proof. During the challenge phase, the adversary outputs a target revoked set S^*, with the restriction that at least one identity in S^* never appeared in any of the previous queried revoked set S'. The adversary wins if it correctly guesses the random bit β used to generate the challenge ciphertext (CT^*, V^*) associated with S^*.

Definition 11. *Our UBE scheme is selectively CPA-secure against Type 1 adversary, and adaptively CPA-secure against Type 2 adversary, if all PPT Types 1 & 2 adversaries have at most a negligible advantage in the above security games.*

We show that our UBE scheme meets the above definition of security.

Theorem 2. *Our UBE scheme is selectively CPA-secure against Type 1 adversaries, assuming that the underlying PK-PRE and SK-PRE schemes are also at least CPA-secure.*

The proof for Theorem 2 is largely similar to that of the Lewko et al.'s BE scheme [85], where the challenger (in the position of the owner and cloud) is against a malicious user. We are more interested in the scenario to defend the newly defined adversary, that is a malicious cloud provider, through the following theorem.

[10] Compare to selective security, adaptive security allows the adversary to only commit the revoked set S^* until the challenge phase.

Theorem 3. *Our UBE scheme is adaptively CPA-secure against Type 2 adversaries, assuming that the decisional bilinear Diffie-Hellman (DBDH) assumption holds, and the underlying PK-PRE and SK-PRE schemes are also at least CPA-secure.*

Proof. Suppose there exists a Type 2 adversary that has a non-negligible advantage over attacking our scheme, we will construct a PPT challenger that uses the adversary to solve the DBDH problem with a non-negligible probability.

The challenger is first given a DBDH challenge tuple

$$(g, g^A, g^B, g^C, T),$$

which is either sampled from $\mathcal{P}_{BDH}$, where $T = e(g,g)^{ABC}$, or from $\mathcal{R}_{BDH}$, where T is uniformly random in $\mathbb{G}_T$. The challenger interacts with the adversary as follows:

Setup. Given the DBDH tuple, the challenger picks random x, b, computes $h = g^x$, and forms the public parameters PP as

$$(g, g^b, g^{b^2}, h, h^b, e(g^A, g^B), H),$$

where H is as defined in the scheme. This implicitly sets $\alpha = AB$ (one of the master secret components in our UBE scheme). The challenger also picks random $d \in \mathbb{Z}_p$ and sets the blinded secret $\mathsf{BS} = g^{Ad}$. Note that since $\alpha = AB$, we have $\mathsf{BS} = g^{Ad} = g^{\alpha \cdot \frac{d}{B}}$. Thus the transformation key TK in the UBE scheme is implicitly set to $\frac{t}{B}$. (Here, both the α and TK values are unknown to the challenger.)

Query Phase. To answer the file key queries from the adversary, the challenger maintains a table $\mathcal{T}$, storing (i, s_i) pairs. The table is initially empty. Given a query for the file key k' associated with any set S', for each $i \in S'$, the challenger first checks if the corresponding entry exists in $\mathcal{T}$. If there is a match, it retrieves the pair (i, s_i); otherwise, it chooses a new random $s_i \in \mathbb{Z}_p$ and adds a new entry to the table. The challenger then computes the file key k' as $H(e(g^A, g^B)^{\sum s_j})$ for all $j \in S'$.

Challenge. The adversary outputs two files (M_0, M_1) of equal length. It also declares a target revoked set S^* such that there is at least one identity in S^* has never appeared in all previous queries associated with revoked set S'. Without losing generality and for the simplicity of the proof, we assume there is only one such different identity, ID_ℓ. The challenger flips a random coin $\beta \in \{0, 1\}$. For each

$i \in S^*, i \neq \ell$, it retrieves the matching (i, s_i) pair from the table $\mathcal{T}$ (such pair always exists as all these identities have been included in one or more S'). It then computes the challenge ciphertext component

$$V^* = \mathsf{SKEnc}(k^*, M_\beta),$$

where $k^* = H(e(g^A, g^B)^{\sum s_i} \cdot T)$ for all $i \in S^*, i \neq \ell$.

It also sets parts of the ciphertext component CT^* as

$$C_0 = g^C \cdot g^{\sum s_i}, \;\; C_{i,1} = g^{b \cdot s_i}, \;\; C_{i,2} = (g^{b^2 \cdot \mathrm{ID}_i} h^b)^{s_i},$$

for all $i \in S^*, i \neq \ell$.

To set the ciphertext component for ID_ℓ of CT^*, it computes

$$C_{\ell,1} = (g^C)^b, C_{\ell,2} = (g^C)^{b^2 \cdot \mathrm{ID}_\ell} \cdot (g^C)^{xb} = (g^{b^2 \cdot \mathrm{ID}_\ell} h^b)^C.$$

Remark: Here the secret share s_ℓ for ID_ℓ is implicitly set as $s_\ell = C$. For the general case where r many identities in S^* never appear in previous S', the challenger simply chooses $r-1$ random s_k values and implicitly sets the remaining one as $\frac{C}{\sum s_k}$. The ciphertext components $C_{k,1}, \; C_{k,2}$ for these r many identities could be computed from s_k and $\frac{C}{\sum s_k}$.

The challenge ciphertext (CT^*, V^*) is returned to the adversary. It is clear that the challenge ciphertext is a valid encryption of M_β with the correct distribution whenever $T = e(g,g)^{ABC} = e(g^A, g^B)^C = e(g,g)^{\alpha C}$, that is, when the input DBDH tuple is sampled from $\mathcal{P}_{BDH}$. On the other hand, when T is chosen uniformly random from $\mathbb{G}_T$, i.e., the tuple is sampled from $\mathcal{R}_{BDH}$, then the ciphertext is independent of β from the adversary's view.

Guess. The adversary outputs its guess β'. If $\beta' = \beta$, then the challenger responds to the DBDH challenge with 1, implying that $T = e(g,g)^{ABC}$. Otherwise, the challenger outputs 0, implying that T is randomly chosen from $\mathbb{G}_T$.

It is clear that, when the DBDH tuple is sampled from $\mathcal{P}_{BDH}$, then the adversary's view is identical

to its view in a real attack game. On the other hand, if the DBDH tuple is sampled from $\mathcal{R}_{BDH}$, then the bit β is actually information-theoretically hidden from the adversary. Thus, if the adversary could break our scheme with a non-negligible probability, then we could use it to solve the DBDH problem with a non-negligible probability.

□

4.3 Updatable Broadcast Encryption - Version 2

We are now ready to describe our second construction, which is an extension of Boneh et al.'s BE scheme [28]. This construction differs from the original one mainly in two aspects: (i) the ciphertext update functionality of revoking or adding users is achieved through a symmetric-key double encryption technique; (ii) both the server and file owner generate part of the master secret key and keep it secret to themselves. This is to reduce the work load of an individual file owner during initial setup.

4.3.1 Building Blocks

Our UBE scheme makes use of two additional primitives defined below.

Intuitively, we make use of a standard public-key encryption (PKE) for key material distribution between a file owner, the recipients and the server.

A PKE scheme comprises the following algorithms:

- $\mathsf{PKGen}(\lambda) \rightarrow (pk, sk)$: The key generation algorithm takes as input a security parameter λ and outputs a public-private key pair (pk, sk).
- $\mathsf{PKEnc}(pk, m) \rightarrow U$: The encryption algorithm takes as input a public key pk and a message m, and outputs the corresponding ciphertext U.
- $\mathsf{PKDec}(sk, U) \rightarrow m$: The decryption algorithm takes as input a private key sk and a ciphertext U, and outputs the corresponding message m if the key sk is correct.

On the other hand, we rely on symmetric-key encryption (SKE) to encrypt the content of a shared file and allow update to the encryption of the content. We first define a standard SKE primitive:

An SKE scheme consists of the following algorithms:

- $\mathsf{SKGen}(\lambda) \rightarrow k$: The key generation algorithm takes as input a security parameter λ and outputs a secret key k.
- $\mathsf{SKEnc}(k, m) \rightarrow V$: The encryption algorithm takes as input a secret key k and a message m, and outputs the corresponding ciphertext V.
- $\mathsf{SKDec}(k, V) \rightarrow m$: The decryption algorithm takes as input a ciphertext V and a secret key k, and outputs the corresponding message m if the key k is correct.

To encrypt a file content, we use a double encryption algorithm SKEnc^* that encrypts the content using two symmetric keys k_0 and k_1 simultaneously. To allow ciphertext update, the update algorithm SKUpd will first decrypt the content with old symmetric key k, and re-encrypt it with a new symmetric key k'. We formally define the two new algorithms below:

- $\mathsf{SKEnc}^*(k_0, k_1, m) \rightarrow V$: The double encryption algorithm takes as input two secret keys k_0, k_1 and a message m, computes $\mathsf{SKEnc}^*(k_0, k_1, m) = \mathsf{SKEnc}(k_1, \mathsf{SKEnc}(k_0, m))$ and outputs the ciphertext V.
- $\mathsf{SKUpd}(k, k', V) \rightarrow V'$: The update algorithm takes as input an old secret key k, a new key k' and the ciphertext V needs to be updated, it computes $\mathsf{SKUpd}(k, k', V) = \mathsf{SKEnc}(k', \mathsf{SKDec}(k, V))$ and outputs the updated ciphertext V'.

4.3.2 Construction

The syntax of our second construction is slightly different from the first one due to the nature of our base scheme, different building blocks used, and to suit for the correct scenario. Thus, for ease of exposition, we choose to describe our second scheme using some new notations without ambiguity.

Definition. As usual, let i denote user index and (pk_i, sk_i) denote the public/private key pair generated from PKGen. Let (pk_o, sk_o) denote a file owner's key pair and (pk_s, sk_s) the file server's key pair. Let S denote the recipient set. For each of the following algorithms, a prefix $O.$, $S.$ or $U.$ indicates the algorithm is performed by the file owner, the server or the user.

An updatable broadcast encryption (UBE) system consists of the following eight algorithms:

- $\mathsf{S.Setup}(\lambda, n) \rightarrow (\mathsf{PP}, \mathsf{MS}_s)$: The server setup algorithm takes as input a security parameter λ, the maximal number of users n and outputs a system-wide public parameter set PP and a server master secret key MS_s.

- $\mathsf{O.Setup}(\mathsf{PP}) \rightarrow \mathsf{MS}_o$: The owner setup algorithm takes as input the public parameter and outputs an owner master secret key MS_o.

- $\mathsf{O.KeyGen}(\mathsf{PP}, \mathsf{MS}_o, i, pk_i) \rightarrow \mathsf{KM}$: The key generation algorithm takes as input the public parameter PP, the owner master secret key MS_o, the user index i and corresponding user public key pk_i, it outputs the user BE private key material KM.

- $\mathsf{O.Enc}(S, \mathsf{PP}, \mathsf{MS}_o, M) \rightarrow (\mathsf{EK}, \mathsf{CT}, V)$: The encryption algorithm takes as input a shared file content M, a recipient set S, the public parameter PP and the owner master secret key MS_o, outputs a file encapsulated key EK, the BE ciphertext CT, and encrypted content V.

- $\mathsf{O.RevUser}(S, i, \mathsf{PP}, \mathsf{EK}, \mathsf{CT}) \rightarrow \mathsf{KM_u}$: The revocation algorithm takes as input the revoked recipient's index i, an old file encapsulated key EK and the current BE ciphertext components CT, together with the public parameter PP and recipient set S, it outputs the ciphertext update key material $\mathsf{KM_u}$.

- $\mathsf{O.AddUser}(S, i, \mathsf{PP}, \mathsf{CT}) \rightarrow \mathsf{KM_u}$: The addition algorithm takes as input a recipient's index i, the current ciphertext components CT, the public parameter PP and the current recipient set S. It outputs the ciphertext update key material $\mathsf{KM_u}$.

- $\mathsf{S.Update}(S, \mathsf{KM_u}, sk_s, \mathsf{CT}, V) \rightarrow (S', \mathsf{CT}', V')$: The update algorithm takes as input the current recipient set S, ciphertext update key material $\mathsf{KM_u}$, server private key sk_s and the ciphertext components (CT, V), it outputs the new recipient set S' and the updated ciphertext components (CT', V').

- $\mathsf{U.Dec}(S, \mathsf{CT}, V, \mathsf{KM}, sk_i) \rightarrow M$: The decryption algorithm takes as input the ciphertext components (CT, V), BE private key material KM and user private key sk_i, and outputs the content M if $i \in S$.

The S.Setup, O.KeyGen, O.Enc, U.Dec algorithms are based on those of Boneh et al.'s scheme [28], while the others are new algorithms designed to meet the requirements.

Construction. We make use a bilinear group $\mathbb{G}$ of prime order p. We assume that all user indices are taken from the set $\mathbb{Z}_p$.[11]

- $\mathsf{S.Setup}(\lambda, n) \rightarrow (\mathsf{PP}, \mathsf{MS}_s)$: The algorithm picks a random group generator $g \in \mathbb{G}$ and a random value $\alpha \in \mathbb{Z}_p$. It computes $g_i = g^{\alpha^i} \in \mathbb{G}$ for $i = 1, 2, \ldots, n, n+2, \ldots, 2n$. It also chooses a hash function $H : \mathbb{G}_T \rightarrow \mathbb{Z}_p$. The system-wide public parameters are

$$\mathsf{PP} = (g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, H).$$

  The server master secret key is $\mathsf{MS}_s = \alpha$ and is kept secret by the cloud provider.

- $\mathsf{O.Setup}(\mathsf{PP}) \rightarrow \mathsf{MS}_o$: The algorithm picks random values $\gamma, z \in \mathbb{Z}_p$, it sets $v = g^\gamma$ and computes g^z. It stores (γ, v, g^z) as MS_o.

- $\mathsf{O.KeyGen}(\mathsf{PP}, \mathsf{MS}_o, i, pk_i) \rightarrow \mathsf{KM}$: The algorithm first computes the BE private key d_i of user i as $d_i = g_i^\gamma g^z$. It then computes $\mathsf{KM} = \mathsf{PKEnc}(pk_i, d_i)$.

- $\mathsf{O.Enc}(S, \mathsf{PP}, \mathsf{MS}_o, M) \rightarrow (\mathsf{EK}, \mathsf{CT}, V)$: For a shared file content M, we assume it is divided into s many blocks m_ℓ for $\ell \in [s]$ during encryption. The algorithm picks a random $t \in \mathbb{Z}_p$ and calculates (C_0, C_1) as

$$C_0 = g^t,\ C_1 = (v \cdot \prod_{j \in S} g_{n+1-j})^t$$

  It duplicates $C_0,\ C_1$ by setting $C_0' = C_0,\ C_1' = C_1$. CT is then set as $\mathsf{CT} = (C_0, C_1, C_0', C_1')$. The algorithm also computes the encapsulated key $\mathsf{EK} = \frac{e(g_1, g_n)^t}{e(g,g)^{zt}}$, from which it derives file keys $k_0 = H(\mathsf{EK}||0)$ and $k_1 = H(\mathsf{EK}||1)$. The algorithm then computes $V_\ell = \mathsf{SKEnc}^*(k_0, k_1, m_\ell) = \mathsf{SKEnc}(k_1, \mathsf{SKEnc}(k_0, m_\ell))$ and sets $V = \{V_\ell\}$ for $\ell \in [s]$ as the encrypted content. It also sets $\mathsf{EK}_0 = \mathsf{EK}$. We obtain the duplicated copy C_0' and C_1' as they will be updated each time a revocation or addition occurs, and will be used by the user to recover latest EK. At the same time, users could obtain the very first EK_0 through the original C_0, C_1 components.

[11] In practice, we could use a collision resistant hash function mapping identity strings to $\mathbb{Z}_p$.

- $\mathsf{O.RevUser}(S, i, \mathsf{PP}, \mathsf{EK}, \mathsf{CT}) \rightarrow \mathsf{KM_u}$: The algorithm first removes the revoked recipient i from S, that is, $S' = S \backslash \{i\}$. It picks a random $t' \in \mathbb{Z}_p$[12], retrieves the latest EK, and computes the new encapsulated key as $\mathsf{EK}' = \mathsf{EK} \cdot \frac{e(g_1, g_n)^{t'}}{e(g,g)^{zt'}} = \frac{e(g_1, g_n)^{t+t'}}{e(g,g)^{z(t+t')}}$. It then performs the following steps:

  1. compute $k_1' = H(\mathsf{EK}'||1)$, $k_1 = H(\mathsf{EK}||1)$, set $\Delta = k_1'||k_1$, and encrypt Δ as $U_\Delta = \mathsf{PKEnc}(pk_s, \Delta)$;
  2. set
  $$\tilde{C}_1 = (v \cdot \prod_{j \in S} g_{n+1-j})^t / g_{n+1-i}^t$$
  $$\tilde{C}_0' = g^{t+t'},\ \tilde{C}_1' = (v \cdot \prod_{j \in S} g_{n+1-j})^{t+t'} / g_{n+1-i}^{t+t'}$$
  3. set
  $$\mathsf{EK} = \mathsf{EK}',\ N = N'.$$

  The above steps are all performed by the file owner, who then sets $\mathsf{KM_u} = (S', U_\Delta, \tilde{C}_1, \tilde{C}_0', \tilde{C}_1')$ and sends it to the server.

- $\mathsf{O.AddUser}(S, i, \mathsf{PP}, \mathsf{CT}) \rightarrow \mathsf{KM_u}$: Adding a recipient i to the current set S is simple. The algorithm first adds i to the recipient set S. Updating CT is similar as revoking a user, simply multiply the g_{n+1-i}^t into the C_1, C_1' components, and no re-randomization of the exponent t is required (hence no key update U_Δ is given to the server).

- $\mathsf{S.Update}(S, \mathsf{KM_u}, sk_s, \mathsf{CT}, V) \rightarrow (S', \mathsf{CT}', V')$: Upon receiving the ciphertext update key material $\mathsf{KM_u}$ from the file owner, the server performs the following steps:

  1. replace S by S'.
  2. set
  $$C_1 = \tilde{C}_1,\ C_0' = \tilde{C}_0',\ C_1' = \tilde{C}_1'.$$

  Note: C_0' component is unchanged when adding a user, thus no update of this component is required under this scenario.

[12] In practice, this t value could be derived from a secret seed value to minimize owner storage

3. recover $\Delta = \mathsf{PKDec}(sk_s, U_\Delta)$ and further parse Δ as k_1', k_1, update the encrypted content V components V_ℓ as $V_\ell' = \mathsf{SKUpd}(k_1, k_1', V_\ell)$;

   Note: when adding a user to the system, the V components are unchanged, and thus no U_Δ is sent to the server and step 3 is not performed.

- $\mathsf{U.Dec}(S, \mathsf{CT}, V, \mathsf{KM}, sk_i) \rightarrow M$: The algorithm first recovers d_i from KM using private key sk_i. If $i \notin S$, the algorithm aborts; otherwise, it computes

$$\mathsf{EK}_0 = \frac{e(g_i, C_1)}{e(d_i \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, C_0)}$$

similarly it can compute

$$\mathsf{EK} = \frac{e(g_i, C_1')}{e(d_i \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, C_0')}$$

resulting $\mathsf{EK}_0 = \frac{e(g_1, g_n)^t}{e(g,g)^{zt}}$ (where t is the very first randomly chosen exponent during initial encryption) and $\mathsf{EK} = \frac{e(g_1, g_n)^{\tilde{t}}}{e(g,g)^{z\tilde{t}}}$ (where $\tilde{t}$ is the latest exponent). With EK_0 and EK, it is easy to compute file keys k_0, k_1, recover the content m_ℓ and further M from V.

Correctness. We verify the correctness of the $\mathsf{U.Dec}$ algorithm of our UBE scheme for EK_0 only. The correctness of EK could be obtained in a similar manner:

$$\begin{aligned}
\mathsf{EK}_0 &= \frac{e(g_i, C_1)}{e(d_i \cdot \prod_{j \in S, j \neq i} g_{n+1-j+1}, C_0)} \\
&= \frac{e(g^{\alpha^i}, (v \cdot \prod_{j \in S} g_{n+1-j})^t)}{e(g_i^\gamma g^z \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, g^t)} \\
&= \frac{e(g^{\alpha^i}, g_{n+1-i}^t) \cdot e(g^{\alpha^i}, (v \cdot \prod_{j \in S, j \neq i} g_{n+1-j})^t))}{e(g_i^\gamma \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, g^t) \cdot e(g^z, g^t)} \\
&= \frac{e(g, g_{n+1})^t \cdot e(g^t, v^{\alpha^i} \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i})}{e(g,g)^{zt} \cdot e(v^{\alpha^i} \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, g^t)} \\
&= \frac{e(g_1, g_n)^t}{e(g,g)^{zt}}
\end{aligned}$$

4.3.3 Security Analysis

Same as previous construction, we consider a standard chosen-plaintext attack (CPA) model in the BE setting [28, 85]. Recall that we consider two types of probabilistic polynomial-time (PPT) adversaries:

- Type 1: malicious users, i.e., unintended (including revoked) file recipients, who are interested to access to the content of a shared file;
- Type 2: malicious cloud provider, who is interested to learn the content of the shared file from interacting with the file owner or the recipient.

However, as the second construction is different from the previous one, the capabilities of these two types of adversary are different with the previous scheme. To simulate a Type 1 adversary, we provide it access to the BE private key (SK) extract oracle. However, we consider a selective security model, where the adversary must first declare the target recipient set S^* of users before given the PP and issues any query.[13] It is then given the public parameter set PP, and subsequently allowed to query the d_i values for any user $i \notin S^*$ to the oracle in order to compute the corresponding file key k. During the challenge phase, the challenger will run the encryption algorithm to obtain the BE ciphertext CT^* and the corresponding file encapsulated key k, the challenger then picks a random $\beta \in \{0, 1\}$, sets $k_\beta = k$ and picks a random $k_{1-\beta}$. It then gives $(\mathsf{CT}^*, k_0, k_1)$ to the adversary. The adversary wins the security game if it correctly guesses which key k_β could be derived from CT^*.

On the other hand, to model a Type 2 adversary, we provide it access to the file encapsulated key (k) extract oracle. It is allowed access to any file encapsulated key associated with any recipient set S' in an adaptive manner (since a new file encapsulated key is chosen each time a user is revoked), and subsequently we could achieve adaptive security against Type 2 adversary.[14] This is to reflect the scenario where the compromise of any file encapsulated key k at any point during file sharing would not affect the confidentiality of the content after it has been updated under a new file encapsulated key k'. During the challenge phase, the adversary outputs a target recipient set S^*, with the restriction that all previously queried recipient sets S' are not subsets of S^*. This restriction reflects that a file

[13] We achieve selective security against Type 1 adversary as our scheme is based on Boneh et al.'s selectively secure BE scheme [28].

[14] Compare to selective security, adaptive security allows the adversary to only commit the revoked set S^* until the challenge phase.

encapsulated key k is not refreshed when adding a user, but will be refreshed whenever a revocation occurs. The adversary wins if it correctly guesses the random bit β used in the challenge ciphertext $(\mathsf{CT}^*, k_0, k_1)$.

We now give the security model against Type 1 adversary, described as a security game between a challenger and a Type 1 adversary. For simplicity, we assume the challenger will impersonate both the file owner and the server, thus we combine the S.Setup and O.Setup as Setup, and unify the master secret key notation as MS. The game is as follow:

- Setup. The challenger runs Setup to obtain the public parameters PP and the master secret key MS. It gives the PP to the adversary and keeps the MS itself.
- Query Phase 1. The adversary adaptively queries the challenger for decryption keys corresponding to user index i. The challenger generates the decryption key and gives it to the adversary.
- Challenge. The adversary submits a challenged user set S^*, subjects to the constraint that any previously queried user index i should not be included in the set S^*. The challenger selects a random bit $\beta \in \{0, 1\}$, generates the challenge ciphertext CT^*, file encapsulated key k and sets $k_b = k$. It also selects a random file encapsulated key k_{1-b} and sends $(\mathsf{CT}^*, k_0, k_1)$ to the adversary.
- Query Phase 2. The adversary continues to submit decryption key queries as in Phase 1, with the same restriction as in the Challenge phase.
- Guess. The adversary outputs its guess $\beta' \in \{0, 1\}$ for β.

The advantage of the Type 1 adversary in this game is defined as $Adv_{Type1} = |Pr[\beta = \beta'] - \frac{1}{2}|$ where the probability is taken over the random bits used by the challenger and the Type 1 adversary.

Definition 12. *An Updatable-BE scheme is secure against Type 1 adversary if all PPT adversaries have at most a negligible advantage in the above game.*

We say that an Updatable-BE scheme is *selective* secure against Type 1 adversary if we restrict the adversary to outputs the target challenge user set before the Setup phase. Our scheme is selective secure.

We now define the security model against Type 2 adversary as a game between the challenger and the adversary. As in our scheme the public parameters are generated by the server, thus during the Setup phase of the security game, the PP is provided by the adversary to the challenger. The game is as follows:

- Setup. The adversary runs S.Setup to generate the public parameters PP and master secret keys MS_s. It sends PP to the challenger and keeps MS to itself.
- Query Phase 1. The adversary adaptively queries the challenger for the file encapsulated key corresponding to the recipient set S'. The challenger generates the file encapsulated key and gives it to the adversary.
- Challenge. The adversary submits a challenged user set S^*, subjects to the restriction that all previously queried recipient sets S' are not subsets of S^*. The challenger selects a random bit $\beta \in \{0, 1\}$, generates the challenge ciphertext CT^* and file encapsulated key k and sets $k_b = k$. It also selects a random file encapsulated key k_{1-b} and sends $(\mathsf{CT}^*, k_0, k_1)$ to the adversary.
- Query Phase 2. The adversary continues to submit queries with the same restriction as in Phase 1.
- Guess. The adversary outputs its guess $\beta' \in \{0, 1\}$ for β.

The advantage of the Type 2 adversary in this game is defined as $Adv_{Type2} = |Pr[\beta = \beta'] - \frac{1}{2}|$ where the probability is taken over the random bits used by the challenger and the Type 2 adversary.

Definition 13. *Our UBE scheme is selectively CPA-secure against Type 1 adversary, and adaptively CPA-secure against Type 2 adversary, if all PPT Types 1 & 2 adversaries have at most a negligible advantage in the above security games.*

We show that our UBE scheme meets the above definition of security.

Theorem 4. *Our UBE scheme is selectively CPA-secure against Type 1 adversaries, assuming that the underlying PK-PRE and SK-PRE schemes are also at least CPA-secure.*

The proof for Theorem 4 is largely similar to that of the Boneh et al.'s BE scheme [28]. Similarly as our first construction, we concentrate in the scenario of defending a malicious cloud provider, described in the following theorem.

Theorem 5. *Our UBE scheme is adaptively CPA-secure against Type 2 adversaries, assuming that the decisional bilinear Diffie-Hellman (DBDH) assumption holds, and the underlying PK-PRE and SK-PRE schemes are also at least CPA-secure.*

Proof. Suppose there exists a Type 2 adversary that has a non-negligible advantage over attacking our scheme, we will construct a PPT challenger that uses the adversary to solve the DBDH problem with a non-negligible probability.

The challenger is first given a DBDH challenge tuple

$$(g, g^A, g^B, g^C, T),$$

which is either sampled from $\mathcal{P}_{BDH}$, where $T = e(g,g)^{ABC}$, or from $\mathcal{R}_{BDH}$, where T is uniformly random in $\mathbb{G}_T$. The challenger interacts with the adversary as follows:

- Setup. The adversary picks its own master secret key α, a hash function H, and generates the public parameter PP as

  $$(g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, H)$$

  and sends PP to the challenger. The challenger will pick a random γ and gives $v = g^\gamma$ to the adversary. The challenger also implicitly sets $z = AB$ which later will be used during the challenge phase.

- Query. To answer any file encapsulates key query for recipient set S', the challenger picks a random t and computes $k = e(g_1, g_n)^t$ and passes it to the adversary.

- Challenge. The adversary outputs a challenge recipient set S^*, the challenger first checks whether S^* satisfies the restriction, if not outputs $\perp$, otherwise it computes

  $$C_0 = g^C, \hat{C}_1 = (g^C)^\gamma;$$

Here the $\hat{C}$ is actually only a part of the correct form of C_1. However, as the adversary has the control of α, and with $C_0 = g^C$ given, the adversary could obtain the correct C_1 format using g^C and α.

The challenger sets the partial file encapsulated key k as $k = T$, it picks a random $\beta \in \{0, 1\}$ and sets $k = k_\beta$. It also picks a random $k_{1-\beta}$. Here note again that as the adversary has control of α, thus with the given g^C, it could compute and distinguish the numerator with certainty. Hence the security game reduces to let the adversary distinguish the denominator from a well formed DBDH tuple or a random one.

The challenger then sends $(C_0, \hat{C}_1, k_0, k_1)$ to the adversary.

- Guess. The adversary outputs its guess β'. If $\beta' = \beta$, then the challenger responds to the DBDH challenge with 1, implying that $T = e(g, g)^{ABC}$. Otherwise, the challenger outputs 0, implying that T is randomly chosen from $\mathbb{G}_T$.

It is clear that, when the DBDH tuple is sampled from $\mathcal{P}_{BDH}$, then the adversary's view is identical to its view in a real attack game. On the other hand, if the DBDH tuple is sampled from $\mathcal{R}_{BDH}$, then the bit β is actually information-theoretically hidden from the adversary. Thus, if the adversary could break our scheme with a non-negligible probability, then we could use it to solve the DBDH problem with a non-negligible probability. □

4.4 Summary

In this chapter, we study the problem of encrypted file sharing (EFS) system. We propose criteria for an ideal large scale EFS system, investigate and analyzed why current solutions could not meet those criteria. We further define a new primitive called updatable broadcast encryption and provide two concrete constructions, which both achieve near-ideal solutions for the EFS system. One future work might be carrying out real implementation works of these two schemes based on some real-life applications, such as Dropbox, and investigate their potential deployment value.

Chapter 5

Multi-Message Broadcast Encryption and Attribute-Based Encryption with Compact Ciphertexts

Contents

In this chapter, we investigate the possibility of reusing random values across multiple message encryption under broadcast encryption (BE) and attribute-based encryption (ABE). Both primitives allow a data owner to encrypt and share information with a set of intended recipients, and current BE and ABE schemes are largely designed for encryption of single messages. However, it is often desirable to encrypt multiple messages simultaneously and share them with potentially different sets of recipients. While this can be done straightforwardly by applying any BE or ABE scheme to each individual message in parallel, the resulting ciphertexts are likely to be large. A fundamental reason for this is that each individual message is typically encrypted using a fresh, unique random value to have an appropriate level of security guarantee. In this chapter, we provide two concrete constructions which reuse the randomness for multiple message encryption: one is based on BE and the other is based on ABE. Both our schemes achieve significant ciphertext savings

5.1 Introduction

5.1.1 Motivations

Cryptographic access control through encryption techniques restricts access to protected information while ensuring confidentiality of the information. Designing a secure, practical and fine-grained cryptographic access control solution has been an interesting and challenging problem. A broad range of solutions have been proposed thus far, see for example [6, 118, 43, 115, 112, 113, 3, 106, 39, 90]. Among these, broadcast encryption (BE) [56, 28, 51, 52, 61] and attribute-based encryption (ABE) [64, 22, 97, 117, 84] are two attractive primitives designed for different usage scenarios. The former

provides basic access control, in the sense that an encryptor explicitly specifies a recipient set such that any recipient in the specified set is able to decrypt a broadcast encrypted message successfully. On the other hand, ABE provides more fine-grained access control. It allows encryption of messages in two ways: key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE). The difference between the two variants is that in KP-ABE, messages are encrypted under attributes, and decryption keys are associated with access structures (or policies); while CP-ABE is the other way round.

Existing BE and ABE schemes largely focus on providing access control to *single* message, although it is often desirable to grant access to *multiple* messages or files *simultaneously* in many application scenarios. One such example is encrypted file systems (EFS), such as the Windows EFS [92]. In the Windows EFS, each user is typically allowed access to multiple files. However, each file F is encrypted under an encryption key k_F, which in turn, is encrypted such as any user authorized to access file F is able to decrypt successfully and recover k_F. Other encrypted file systems, such as SiRiUS [62] and Plutus [80], also use a similar technique to share secret (symmetric) file encryption keys. Another relevant application scenario is access control to electronic health record (EHR) [4, 66]. Typically, a patient's EHR could contain different types of information, such as contact information, family medical history, symptom and treatment and insurance policy, which may require enforcement of different access policies. Hence, different parts or sections of the record may need to be encrypted differently, in the sense that each part is accessible only to some but not all personnels.

A straightforward but somewhat naïve approach to realize access control in the above scenarios is by applying a BE or ABE scheme on multiple messages or files in parallel. For example in the Windows EFS, the encryption key k_{F_i} for each file F_i can be protected using a BE scheme and stored in the header, and the whole header containing all the encryption keys k_{F_i} is then broadcast to all users. Similarly, each section of a health record requiring different access control can be protected by an ABE scheme. Although current BE schemes, such as that of [28], already enjoy very compact ciphertext sizes, a straightforward approach still results ciphertexts which grow linearly with respect to the number of encrypted files. Such expansion can be even more significant in ABE schemes. In the EHR example, the size of ciphertexts (corresponding to encryption of multiple sections) will grow quadratically. This can be costly. An IDC report shows that data generation is out-pacing storage availability [58]. Moreover,

it is becoming very common for users (data owners) to store their personal or work related files on the cloud, where the storage capacity made available to each user is limited and one must pay for additional storage. For example, Dropbox for Business charges USD15 per user per month with minimum 5 users [1].

Motivated by the above requirements, the goal of this chapter is then to propose a primitive which encrypts *multiple* messages with *different* access control requirements for each message, while having *compact* ciphertexts.

5.1.2 Technical Challenges

Blinding factor. Typically in BE and ABE schemes, a blinding factor is used as some form of a one-time pad to hide a plaintext. The blinding factor is computed from some public information and an encryptor-chosen random value. We call such public information the "base value" and the random value is known only to the encryptor. Moreover, the encryptor must include one or more "helper values" in the ciphertext in order to allow any authorized user to decrypt successfully. Such helper values contain information associated with the used random value, which in turn, can be used to reconstruct the blinding factor, while still retaining the secrecy of the random value through certain hard mathematical assumptions. In a more complicated system like KP-ABE, the "helper values" could be in linear size with respect to the number of attributes[1]. On the other hand, in order to achieve ciphertext indistinguishability, a fresh random value needs to be picked for each message encryption. Thus, when multiple messages are encrypted, the helper values need to be different for each message, and this consequently results ciphertexts that grow quadratically.

Randomness reuse. One natural idea to achieve compact ciphertexts when encrypting multiple messages is to reuse the same encryptor-chosen random value for all messages. However, this must be treated with great care. As pointed out by Bellare et al. in [19]:

> "It is a well-known fact that privacy in the sense of IND-CPA is not met if two messages are encrypted using the same randomness under the same key."

[1] We notice that there has been a recent proposal of ABE scheme with constant-size ciphertexts [74]. We discuss such variant and its performance trade-off in Section 5.4.2.

Bellare et al. proposed a new primitive called multi-recipient encryption scheme (MRES) and studied the possibility of randomness reuse in order to save bandwidth and achieve better computation efficiency. They gave a *randomness reproducibility* theorem which shows that, indeed, there exist public key encryption schemes that allow random values to be reused safely across multiple encryptions and yet, the indistinguishability of the resulting ciphertexts can be guaranteed. Take the ElGamal encryption used in [19] as an example: given a message M, the encryption algorithm first picks a random value r, then computes the ciphertext as $(g^r, X^r M)$, where g is a system-wide public parameter and X is a recipient's public key. By reusing the same randomness from r, the value g^r is used across all different messages, and thus the total ciphertext size is reduced by almost half. At the same time, using the reproducibility theorem, the system can still be proven secure assuming that each recipient has a unique public key value X.

Limitation. Nevertheless, MRES is designed for a *one-to-one* message-recipient case (i.e., each message is encrypted under a specific public key). The randomness reproducibility theorem guarantees that a system is secure as long as a distinct public key is used only *once* during encryption.

On the other hand, existing BE and ABE schemes are designed for a *one-to-many* message recipient case, where each message is encrypted under multiple public keys. However, to address our aforementioned requirements, we need a primitive that deals with a *many-to-many* case, where multiple messages can be encrypted simultaneously, and each of the messages is encrypted under a set of public keys. The recipients corresponding to the public key sets can be all different or have some overlaps. Clearly, we can no longer use the same random value across all the messages and rely on just a recipient's public key as a base value to compute a blinding factor. Otherwise, an authorized recipient for a particular message is able to compute the *same* blinding factor that allows recovery of not only the intended message(s) but also other messages intended for other recipients.

Therefore, we come to the following two contradicting requirements: on the one hand, we wish to enforce different access policies to different messages (which in turn requires a unique blinding factor for encryption of each message); on the other hand, we also wish to reuse the same randomness across multiple messages to obtain compact ciphertexts.

5.1.3 Related Work

We are not the first to study the possibility of reusing randomness in the broadcast encryption setting. Phan et al. recently proposed multi-channel broadcast encryption [99] which supports encryption of multiple messages intended for different sets of recipients. They proposed a scheme with constant-size ciphertexts by reusing the same random value across all messages during encryption. However, their scheme has two drawbacks. First, they considered a rather restricted usage scenario, i.e., Pay-TV. In such scenario, the key authority and the encryptor are the *same* entity, and thus, it is straightforward to introduce and embed new randomness for each message. (This is because the encryptor knows the master secret key, from which new random values can be determined.) Second, even though their scheme deals with encryption of multiple messages, it supports only *disjoint* recipient sets. This is a rather strong requirement which may not always be realistic. In our Windows EFS example, the recipient sets are likely to overlap. Our proposed constructions have no such restriction.

Gasti and Merlo [59] also considered randomnness reuse in a BE scheme. They proposed two constructions based on the BE scheme of Delerablée et al. [52]. Their first scheme is also restricted in the sense that, the recipient sets for all messages must be the *same*. Their second construction removes this requirement, however, their randomness reuse technique does not seem to be useful in reducing the bandwidth requirement. In order to reuse the same random value k used for the formation of each message encryption key, they directly embed a new random value into each key to provide indistinguishability of encrypted session keys. Furthermore, the helper values for these new random values are directly embedded into the ciphertext. Thus, overall, the ciphertext size of their construction is linear with respect to both the number of revoked users (those who have no access to that particular message) of each message, and the number of messages that have been encrypted.

Libert et al. studied randomness reuse in an anonymous broadcast encryption setting [87]. In order to achieve anonymity (without revealing the target recipient set), their scheme encrypts a message ℓ times for a recipient set S with cardinality $|S| = \ell$. Using a key derivation function, the same random value could be reused across all ℓ encryptions of the *same* message. Hence the helper value will be the same and could be reused, and thus reducing the ciphertext size. However, it is not immediately clear how their technique can be extended to our many-to-may message-recipient setting.

Bellare et al. generalized their randomness reuse technique and MRES, and proposed the concept of stateful public key cryptosystems [20]. Further work on stateful public key encryption can be found in [16, 13], which include adaptation to the identity-based encryption (IBE) setting.

5.1.4 Our Approach & Contributions

We explore techniques for enhancing BE and ABE schemes for supporting many-to-many message-recipient pairs, that is, encryption of multiple messages under different access policies, while achieving compact ciphertexts.

The blinding factor for a message in BE and ABE schemes take the form of $e(g,g)^{\alpha s}$, where e is a bilinear map, g is a group element, and s is the encryptor-chosen random value. Here $e(g,g)^{\alpha}$ is publicly known and what we call a base value. The random value s also appears in the corresponding helper values in the ciphertext, for example the g^s term in the BE scheme of [28], or the T_i^s terms in the KP-ABE scheme of [84]. Typically, during encryption of a message, a fresh random s needs to be chosen, and thus the corresponding helper values are different for each message. However, if we were to reuse the same s across multiple messages, we must get some other unique randomness (with respect to each message) from elsewhere. We achieve this through embedding a set of distinct base values in the public parameters during system setup, such that the base values can be combined with the (reused) random value s chosen by the encryptor. This way, the blinding factor for each message will be different. Although this approach allows us to reuse the same random value s across encryption of all messages, it increases the public parameter size proportionate to the maximal number of messages the system can encrypt at one time. To avoid such blow-up in the public parameter size, we publish only one base value in the public parameters from which users (encryptors) can derive other (pre-defined) base values. We realize this by employing an RSA-based *key rotation* technique [80], such that the published base value can be "rotated" to obtain a different base value for a different message. (We elaborate more on this in Section 5.3)

This chapter makes the following contributions. We first formalize two new concepts called *multi-message broadcast encryption* (MM-BE) and *multi-message key-policy attribute-based encryption* (MM-KP-ABE), which allow an encryptor take multiple messages as input during encryption and enforce

different access rights on each message (i.e., targeting different recipient sets). We also give security definitions for our new primitives. We then provide two concrete schemes. Our first scheme provides basic access control (i.e., messages are encrypted under recipients' public keys) and is based on the BE scheme by Boneh et al. [28]. Our MM-BE scheme reduces the ciphertext size by almost half compared to the described trivial approach. The second scheme is based on the key-policy attribute-based encryption (KP-ABE) by Lewko et al. [84], and thus allows more fine-grained access control (i.e., messages are encrypted under attributes). Our MM-KP-ABE scheme enjoys even more savings, reducing the ciphertext size from quadratic (trivial approach) to linear. (We provide further efficiency analysis for our schemes in Sections 5.3.2 & 5.4.2, respectively.)

5.1.5 Outline

The rest of this chapter is organized as follow: In Section 5.2, we formally define our two new primitives and their security models. In Sections 5.3 & 5.4, we present two concrete constructions: one from BE, and the other from KP-ABE, respectively. We summarize this chapter and highlight some open problems in Section 5.5.

5.2 Primitive Definitions

5.2.1 Definition of MM-BE

We first give the definition of multi-message broadcast encryption (MM-BE), which consists of four algorithms[2]:

- $\mathsf{Setup}(1^k, n, m) \rightarrow (\mathsf{MK}, \mathsf{PP})$. This algorithm is run by the authority. The setup algorithm takes as input the security parameter k, the total number of users n and maximal number of messages m that the system could encrypt at one time. It outputs a master secret key MK and public parameters PP.

[2]In some schemes, like [28], the key generation algorithm is combined with the setup algorithm. For ease of exposition, we separate them in our description

- $\mathsf{KeyGen}(\mathsf{MK}, \mathsf{PP}, i) \rightarrow d_i$. The key generation algorithm takes as input the master secret key MK, public parameters PP and a user index i. It outputs the user's private key d_i.

- $\mathsf{Enc}(\mathsf{PP}, \mathsf{M}, \mathsf{S}) \rightarrow (\mathsf{Hdr}, \mathsf{CT})$. The encryption algorithm takes as input the public parameters PP, a set $\mathsf{M} = \{M_1, \ldots, M_m\}$ of messages and the corresponding set S of user sets $\{S_1, \ldots, S_m\}$, where each set $S_j \; \forall j \in [m]$ contains the user indices of the intended recipients for message M_j. It outputs a set (Hdr, K) contains pairs in the form of (Hdr_j, K_j) where Hdr_j is called the header and $K_j \in \mathcal{K}$ is a message encryption key. We will often refer to Hdr as the broadcast ciphertext.

  Let M_j be a message to be broadcast to the set S_j and let C_j be the encryption of M_j under the symmetric key K_j. The broadcast to users consists $(\mathsf{S}, \mathsf{Hdr}, \mathsf{CT})$ where

$$\mathsf{S} = \{S_1, \ldots, S_j, \ldots, S_m\};$$

$$\mathsf{Hdr} = \{\mathit{Hdr}_1, \ldots, \mathit{Hdr}_j, \ldots, \mathit{Hdr}_m\};$$

$$\mathsf{CT} = \{C_1, \ldots, C_j, \ldots, C_m\}$$

  for $\forall j \in [m]$.

- $\mathsf{Dec}(\mathsf{Hdr}, d_i, i, \mathsf{PP}, \mathsf{S}) \rightarrow (K_j, \perp)$. The decryption algorithm takes as input the header Hdr, a private key d_i, public parameter PP and user sets S. It outputs either a message encryption key K_j for message MK_j if $i \in S_j$, or $\perp$ otherwise. The key K_j then can be used to decrypt the C_j component to obtain the message M_j.

As usual, we require the system to be consistent, namely for all security parameters λ, all PP and MK output by the Setup algorithm, all messages encryption key K_j, and all corresponding header components Hdr_j output by the Enc algorithm encrypted under recipient set S_j, we have

$$\mathsf{Dec}(\mathit{Hdr}_j, d_i, i, \mathsf{PP}, S_j) = K_j \quad \text{if} \quad i \in S_j.$$

5.2.2 Security Model of MM-BE

We are now ready to define an adaptive CPA security model for MM-BE using the following game between an adversary $\mathcal{A}$ and a challenger. Both the adversary and challenger are given n, the total number of system users, and m, the maximal number of messages the system could encrypt at any given time. Here, we adapt the security model to the traditional BE setting, that is, the adversary is given two sets of symmetric encryption keys and challenged on which key set is encrypted and stored in the header (see Section 5.3.3 for further details).

Setup. The challenger runs $\mathsf{Setup}(1^k, n, m)$ to obtain the master secret key MK and public parameters PP. It passes PP to the adversary.

Query phase 1. Adversary $\mathcal{A}$ may adaptively issue private key queries with user index i. The challenger runs $\mathsf{KeyGen}(\mathsf{MK}, \mathsf{PP}, i)$ and passes d_i to $\mathcal{A}$.

Challenge. Adversary $\mathcal{A}$ outputs a set of user sets $\mathsf{S}^* = \{S_1^*, \ldots, S_m^*\}$ he wishes to attack, subject to the constraint that for all i queried in Phase 1, $i \notin \bigcup S_j^* \quad \forall j \in [m]$. The challenger runs $\mathsf{Enc}(\mathsf{S}^*, \mathsf{PP})$ to obtain $(\mathsf{Hdr}^*, \mathsf{K}^*)$. The challenger then picks a random $b \in \{0, 1\}$. It sets $K_b = K^*$ and picks a random encryption key set K_{1-b} in $\mathcal{K}$. It then gives $(\mathsf{Hdr}^*, K_0, K_1)$ to the adversary.

Query phase 2. Adversary $\mathcal{A}$ continues to issue private key queries, subject to the same constraint as in challenge phase. The challenger responds as in phase 1.

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

Definition 14. *Let $Adv_{\mathcal{A}}$ denote the probability that $\mathcal{A}$ wins the game. We say that an MM-BE system is adaptive CPA secure if for all polynomial-time adversary $\mathcal{A}$, we have that $|Adv_{\mathcal{A}} - \frac{1}{2}| < \epsilon$.*

Selective Security. In the selective security setting, we require the adversary to output the target set of user sets S^* before the setup phase. Our MM-BE scheme is selectively secure, same as [28].

Chosen-Ciphertext Attack (CCA) Security. The CCA security is defined by allowing the adversary issue decryption queries. Our scheme is CPA secure. CCA security could be obtained by adopting the standard technique as in [36].

5.2.3 Definition of MM-KP-ABE

We now give the definition of multi-message key-policy attribute-based encryption (MM-KP-ABE), which consists of four algorithms:

- $\mathsf{Setup}(1^k, n, m) \rightarrow (\mathsf{MK}, \mathsf{PP})$. This algorithm is run by the authority. The setup algorithm takes as input the security parameter k, the total number of users n and maximal number of messages m that the system could encrypt at one time. It outputs a master secret key MK and public parameters PP.

- $\mathsf{KeyGen}(\mathsf{MK}, \mathsf{PP}, (A, \rho)) \rightarrow sk_i$. The key generation algorithm takes as input the master secret key MK, public parameters PP and a user access structure (A, ρ). It outputs the user's private key sk_i.

- $\mathsf{Enc}(\mathsf{PP}, \mathsf{M}, \mathsf{S}) \rightarrow \mathsf{CT}$. The encryption algorithm takes as input the public parameters PP, a set $\mathsf{M} = \{M_1, \ldots, M_m\}$ of messages and the corresponding set S of attribute sets $\{S_1, \ldots, S_m\}$, where each set S_j $\forall j \in [m]$ contains the attributes of the intended recipients for message M_j. It outputs a *set* $\mathsf{CT} = \{C_1, \ldots, C_m\}$ of ciphertexts.

- $\mathsf{Dec}(\mathsf{CT}, sk_i) \rightarrow (M_j, \perp)$. The decryption algorithm takes as input the ciphertext set CT and a private key sk_i. It outputs either a message M_j if sk_i is the correct private key, or $\perp$ otherwise.

As usual, we require the system to be consistent, namely for all security parameters λ, all PP and MK output by the Setup algorithm, all messages M_j, and all corresponding ciphertext components C_j output by the Enc algorithm encrypted under attribute set S_j, we have $\mathsf{Dec}(C_j, sk_i) = M_j$ if the access structure (A, ρ) satisfies the attribute set S_j.

5.2.4 Security Model of MM-KP-ABE

We now define an adaptive CPA security model for MM-KP-ABE using the following game between an adversary $\mathcal{A}$ and a challenger. Both the adversary and challenger are given n, the total number of system users, and m, the maximal number of messages the system could encrypt at any given time.

Setup. The challenger runs $\mathsf{Setup}(1^k, n, m)$ to obtain the master secret key MK and public parameters PP. It passes PP to the adversary.

Query phase 1. Adversary $\mathcal{A}$ may adaptively issue private key queries with access structure (A_i, ρ_i). The challenger runs $\mathsf{KeyGen}(\mathsf{MK}, \mathsf{PP}, (A_i, \rho_i))$ and passes sk_i to $\mathcal{A}$.

Challenge. Adversary $\mathcal{A}$ outputs a set of attribute sets $\mathsf{S}^* = \{S_1^*, \ldots, S_m^*\}$ he wishes to attack, subject to the constraint that the private key sk_i associated with (A_i, ρ_i) for all i has ever been queried in Phase 1 could not be satisfied by S_j^* $\forall j \in [m]$. The adversary then outputs two sets of messages $\mathsf{M}_0^* = \{M_{0,1}^*, \ldots, M_{0,m}^*\}$ and $\mathsf{M}_1^* = \{M_{1,1}^*, \ldots, M_{1,m}^*\}$, such that $|\mathsf{M}_0^*| = |\mathsf{M}_1^*|$ and $|M_{0,j}^*| = |M_{1,j}^*|$ for $1 \leq j \leq m$. The challenger picks a random $b \xleftarrow{\$} \{0,1\}$ and runs $\mathsf{Enc}(\mathsf{PP}, \mathsf{M}_b^*, \mathsf{S}^*)$. The resulting challenge $\mathsf{CT}^* = \{C_1^*, \ldots, C_m^*\}$ is then returned to $\mathcal{A}$.

Query phase 2. Adversary $\mathcal{A}$ continues to issue private key queries, subject to the same constraint as in challenge phase. The challenger responds as in phase 1.

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

Definition 15. *Let $Adv_{\mathcal{A}}$ denote the probability that $\mathcal{A}$ wins the game. We say that an MM-KP-ABE system is adaptive CPA secure if for all polynomial-time adversary $\mathcal{A}$, we have that $|Adv_{\mathcal{A}} - \frac{1}{2}| < \epsilon$.*

Selective Security. As usual, in the selective security setting, we require the adversary to output the target set of attribute sets S^* before the setup phase.

Chosen-plaintext Attack (CCA) Security. The CCA security is defined the same as MM-BE.

In the above two security models, we allow the adversary to attack a set of user sets to capture our notion of multi-messages. This is in contrast to existing works where only one targeted user set is specified by the adversary.

5.3 MM-BE Construction

5.3.1 Construction

We now provide a construction based on Boneh et al's BE scheme [28]. Intuitively, we pre-embed some randomnesses in the public parameters that the system could use during encryption. This allows

the encryptor to reuse his own random value picked during encryption across multiple messages, while ensuring that the blinding factors for all messages are still distinct. However, to encrypt m different messages, we require m distinct random values. To minimize the number of public parameters for embedding the required randomness, we employ an RSA-based key rotation technique [80].

Essentially, we publish a base value, from which unique random values can be derived, as part of the public parameters. The base value is computed by rotating forward an RSA decryption key m times over the public parameter $e(g, g)$. To derive a random value for encrypting message M_j, the encryptor simply rotates backward this public base value $j - 1$ times[3].

The description of our construction is as follows:

- $\mathsf{Setup}(1^k, n, m)$: The setup algorithm first generates a standard RSA public/private key pair $(\mathbf{e}, \mathbf{d})$ and the corresponding modulus $\mathbf{n}$. Let $\mathbb{G}$ be a bilinear group of prime order p. The algorithm picks a random generator $g \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p$. It computes $g_i = g^{\alpha^i} \in \mathbb{G}$ for $i \in [2n] \backslash \{n + 1\}$. Next, it picks a random value v and a hash function $H(\cdot) : Z_p \to Z_p$. It calculates $v_1 = H(v), v_2 = H(v_1), \ldots, v_m = H(v_{m-1})$. For $1 \leq j \leq m$, it computes $d_j = \mathbf{d}^{m-j+1}$. It also computes

$$B_m = e(g, g)^{\mathbf{d}} \pmod{\mathbf{n}},$$

$$B_{m-1} = B_m^{\mathbf{d}} \pmod{\mathbf{n}} = e(g, g)^{\mathbf{d}^2} \pmod{\mathbf{n}},$$

$$\ldots,$$

$$B_1 = B_2^{\mathbf{d}} \pmod{\mathbf{n}} = e(g, g)^{\mathbf{d}^m} \pmod{\mathbf{n}}.$$

  The public parameters are:

$$\mathsf{PP} = (g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, B_1 = e(g, g)^{\mathbf{d}^m}, v, H(\cdot), (\mathbf{e}, \mathbf{n})).$$

  and the master secret key MK is $(\alpha, \mathbf{d})$.

  Note here the v_j values are embedded into user private keys and we publish only the base value v

[3]By doing so, we need an isomorphism between the group $\mathbb{G}_T$ and a cyclic subgroup of $\mathbb{Z}_{\mathbf{n}}^*$, where $\mathbf{n}$ is the RSA modulus. Such isomorphism exists, for example reader may refer to [12].

and $H(\cdot)$.

- $\mathsf{KeyGen}(\mathsf{MK}, \mathsf{PP}, i)$: The private key for each user i is set to be

$$d_i = \{d_{i,j} = v_j^{\alpha^i} g^{d_j}\} \qquad \forall 1 \leq j \leq m.$$

- $\mathsf{Enc}(\mathsf{PP}, \mathsf{M}, \mathsf{S})$: The encryption algorithm takes as input a set of user set $\mathsf{S} = \{S_1, \ldots, S_m\}$ and the public parameters. To generate the message encryption key K_j for each user set $S_j \in \mathsf{S}$, the algorithm picks a random $t \in \mathbb{Z}_p$ and sets

$$K_j = \frac{e(g_{n+1}, g)^t}{B_j^t} = \frac{e(g_{n+1}, g)^t}{e(g, g)^{\mathbf{d}^{m-j+1} \cdot t}} = \frac{e(g_{n+1}, g)^t}{e(g, g)^{d_j \cdot t}}.$$

  The value $e(g_{n+1}, g)$ can be computed as $e(g_n, g_1)$. The value $e(g, g)^{d_j} = e(g, g)^{\mathbf{d}^{m-j+1}}$ is obtained through an RSA decryption operation: given $B_1 = e(g, g)^{\mathbf{d}^m}$ and $(\mathbf{e}, \mathbf{n})$, the encryptor calculates

$$B_j = B_1^{\mathbf{e}^{j-1}} = e(g, g)^{\mathbf{d}^{m-j+1}(\mathbf{de})^{j-1}} = e(g, g)^{\mathbf{d}^{m-j+1}} \pmod{\mathbf{n}}.$$

  The algorithm also obtains v_j by evaluating j times $H(\cdot)$ over the value v. The header for set S_j is then set as

$$\mathit{Hdr}_j = (g^t, (v_j \cdot \prod_{k \in S_j} g_{n+1-k})^t).$$

  and each message M_j is encrypted using K_j.

  Note that the same randomness t is *reused* for different messages. However since the encryption algorithm uses different $e(g, g)^{\mathbf{d}^j}$ values for different messages, the message encryption key K_j (which is considered as the blinding factor for the message) is always unique for each user set. The header set for m messages is then $\mathsf{Hdr} = (\mathit{Hdr}_0, \mathit{Hdr}_1, \ldots, \mathit{Hdr}_m)$ where $\mathit{Hdr}_0 = g^t$ (Note here we abuse the notation of Hdr_j by referring it to only its second component).

- $\mathsf{Dec}(S, i, d_i, \mathsf{Hdr}, \mathsf{PP})$: Let $\mathsf{Hdr} = (\mathit{Hdr}_0, \mathit{Hdr}_j)$ $\forall j \in [m]$. To obtain the message encryption key

K_j, if $i \in S_j$, then calculate

$$K_j = e(g_i, \mathit{Hdr}_1)/e(d_{i,j} \cdot \prod_{\substack{k \in S_j \\ k \neq i}} g_{n+1-k+i}, \mathit{Hdr}_0).$$

Remark: Instead of using a hash function $H(\cdot)$ to generate different v_j values, one could use the same technique as in [28, Section 3.2], by directly choosing m many random v values and publish them. This will increase the public parameter size by m group elements.

CORRECTNESS:

$$\begin{aligned}
K_j &= e(g_i, \mathit{Hdr}_1)\ /e(d_{i,j} \cdot \prod_{\substack{k \in S_j \\ k \neq i}} g_{n+1-k+i}, \mathit{Hdr}_0) \\
&= e(g^{\alpha^i}, (v_j \cdot \prod_{k \in S_j} g_{n+1-k})^t)\ /e(v_j^{\alpha^i} g^{d_j} \cdot \prod_{\substack{k \in S_j \\ k \neq i}} g_{n+1-k+i},\ g^t) \\
&= e(g^{\alpha^i}, g_{n+1-i}^t) \cdot e(g^{\alpha^i}, (v_j \cdot \prod_{\substack{k \in S_j \\ k \neq i}} g_{n+1-k})^t)\ /(e(v_j^{\alpha^i} \cdot \prod_{\substack{k \in S_j \\ k \neq i}} g_{n+1-k+i},\ g^t) \cdot e(g^{d_j},\ g^t)) \\
&= e(g_{n+1}, g)^t \cdot e(g^{\alpha^i}, (v_j \cdot \prod_{\substack{k \in S_j \\ k \neq i}} g_{n+1-k})^t)\ /(e(v_j^{\alpha^i} \cdot \prod_{\substack{k \in S_j \\ k \neq i}} g_{n+1-k+i},\ g^t) \cdot e(g, g)^{d_j \cdot t}) \\
&= e(g_{n+1}, g)^t \cdot e(g, v_j^{\alpha^i} \cdot \prod_{\substack{k \in S_j \\ k \neq i}} g_{n+1-k+i})^t\ /(e(v_j^{\alpha^i} \cdot \prod_{\substack{k \in S_j \\ k \neq i}} g_{n+1-k+i},\ g)^t \cdot e(g, g)^{d_j \cdot t}) \\
&= e(g_{n+1}, g)^t\ /e(g, g)^{d_j \cdot t}
\end{aligned}$$

5.3.2 Efficiency and Trade-off

Let us compare the efficiency of the above scheme with a naïve approach (i.e., trivial combination of multiple instances of broadcast encryption in [28]). Here, we consider encryption of m messages.

- Header (ciphertext) size: Since t is reused across all messages, the size of our header set for m messages is $m+1$ group elements (i.e., the reused $\mathit{Hdr}_0 = g^t$ element plus m other different Hdr_j elements), while the naïve approach has $2m$ group elements. Thus our scheme has only almost

half the size of that of the trivial approach. We stress that such saving on ciphertexts is especially appealing in a cloud environment, where users have to pay for storage.

- Public parameter size: Our public parameter size has slightly increased from $2n + 1$ group elements (in the trivial approach) to $2n + 4$ (exclusive of the description of $H(\cdot)$). Such increment is very marginal compared to the savings we get in return in the ciphertexts.

- Private key size: A normal broadcast encryption private key consists of only one group element, however our scheme requires m elements (i.e., one group element per message). Nevertheless, private key generation and distribution is a one-off process during system setup. Further, the key is stored at the user-side, and thus the increased storage requirement for private keys seems acceptable, particularly for cloud storage applications.

5.3.3 Security

We now prove selective CPA security of the scheme described in Section 5.2.2.

Theorem 6. *Let $\mathbb{G}$ be a bilinear map of prime order p. Our scheme is selective CPA-secure assuming the decisional n-BDHE assumption holds in $\mathbb{G}$.*

Proof. Suppose there exist an adversary $\mathcal{A}$ which has advantage ϵ in breaking our scheme, then we could build an algorithm $\mathcal{B}$ that has advantage ϵ in solving the n-BDHE problem in $\mathbb{G}$. Algorithm $\mathcal{B}$ proceeds as follows:

- **Init** Algorithm $\mathcal{B}$ runs $\mathcal{A}$ and receives the target set of user sets $S^* = \{S_1^*, \ldots, S_m^*\}$.

- **Setup** By taking the n-BDHE challenge $(g, h, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, Z)$ as input where Z is either $e(g_{n+1}, h)$ or a random element of $\mathbb{G}_T$, algorithm $\mathcal{B}$ proceeds to form the public parameters. It first chooses a random v and a hash function $H(\cdot)$. The crux of the proof is how to simulate $H(\cdot)$ to form different v_j, which should be properly distributed in the attacker's view: $\mathcal{B}$ chooses random $u_j \in \mathbb{Z}_p$ and sets $v_j = g^{u_j}(\prod_{k \in S_j} g_{n+1-k})^{-1}$ for $j \in [m]$. It also generates the RSA public/private key pair $(\mathbf{e}, \mathbf{d})$ and the corresponding modulus $\mathbf{n}$, and computes the corresponding

$d_j = \mathbf{d}^{m-j+1}$ for $1 \leq j \leq m$ and $B_1 = e(g,g)^{\mathbf{d}^m} \pmod{\mathbf{n}}$. It then gives $\mathcal{A}$ the public parameters

$$PK = (g, g_1, \ldots, g_n, g_{n+2}, g_{2n}, B_1, v, H(\cdot), (\mathbf{e}, \mathbf{n})).$$

- **Query phase** Adversary $\mathcal{A}$ may adaptively query the private key of user i. Algorithm $\mathcal{B}$ computes $d_{i,j}$ as:

$$d_{i,j} = g_i^{u_j} \prod_{k \in S_j} (g_{n+1-k+i})^{-1} \cdot g^{d_j}.$$

Indeed, we have $d_{i,j} = (g^{u_j} \prod_{k \in S_j} (g_{n+1-k})^{-1})^{\alpha^i} \cdot g^{d_j} = v_j^{\alpha^i} \cdot g^{d_j}$ as required. $\mathcal{B}$ will answer the private key query in this form for queries in both phases 1 & 2.

- **Challenge** To generate the challenge, $\mathcal{B}$ computes the header set as

$$\mathsf{Hdr} = \{h, \mathit{Hdr}_1, \ldots, \mathit{Hdr}_m\} = (h, h^{u_1}, \ldots, h^{u_m})$$

It then randomly chooses a bit $b \in \{0,1\}$, sets $K_b = \{K_{j,b} = \frac{Z}{e(g,\, h)^{d_j}} \mid \forall j \in [m]\}$ and picks m random $K_{j,1-b}$ in $\mathbb{G}_T$ to form $K_{1-b} = \{K_{j,1-b} \mid \forall j \in [m]\}$. It then gives $(\mathsf{Hdr}, K_b, K_{1-b})$ as the challenge to $\mathcal{A}$.

We claim that when $Z = e(g_{n+1}, h)$, then $(\mathsf{Hdr}, K_b, K_{1-b})$ is a valid challenge to $\mathcal{A}$ as in a real attack. To see this, write $h = g^t$ for some (unknown) $t \in \mathbb{Z}_p$. Then we have

$$\begin{aligned} h^{u_j} = (g^t)^{u_j} = (g^{u_j})^t &= (g^{u_j}(\prod_{k \in S_j} g_{n+1-k})^{-1}(\prod_{k \in S_j} g_{n+1-k}))^t \\ &= (v_j \prod_{k \in S_j} g_{n+1-k})^t. \end{aligned}$$

Therefore, by definition Hdr is a valid header set. Also we have $\frac{e(g_{n+1},g)^t}{e(g,g)^{d_j t}} = \frac{e(g_{n+1},h)}{e(g,h)^{d_j}} = \frac{Z}{e(g,\, h)^{d_j}} = K_{j,b}$, and hence $(\mathsf{Hdr}, K_b, K_{1-b})$ is a valid challenge to $\mathcal{A}$.

On the other hand, when Z is random in $\mathbb{G}_T$, then both $K_{j,b}, K_{j,1-b}$ are just random and independent elements in $\mathbb{G}_T$. This concludes the proof for Theorem 6.

□

5.4 MM-KP-ABE Construction

5.4.1 Construction

We now present a more fine-grained access control mechanism compared to our previous BE-based scheme: multi-message key-policy attribute-based encryption (MM-KP-ABE). In our construction, we use the KP-ABE scheme of [84] as a building block. Our goal is to reuse the value s and helper values T_i^s in a typical KP-ABE ciphertext across encryption of all different messages. As with our previous MM-BE scheme, we pre-embed randomness in the public parameters such that distinct base values can be derived through the RSA-based key rotation technique. However, since we now deal with composite order bilinear groups, we employ multi-prime RSA [73], i.e., the underlying RSA modulus is a product of three primes.

Our scheme involves n distinct attributes and, as before, it could encrypt up to m messages at any given time. Let A be a matrix and ρ be a map from each row x of A to an attribute $\rho(x)$. The scheme is as follows:

- $\mathsf{Setup}(1^k, n, m)$: The setup algorithm chooses a bilinear group G of order $N = p_1p_2p_3$ (i.e., three distinct primes). It also chooses an RSA public/private key pair $(\mathbf{e}, \mathbf{d})$ with modulus N (which is the same as the composite group order of G). It computes $d_j = \mathbf{d}^{m-j+1}$ for $1 \leq j \leq m$, and $B_m = e(g,g)^{\mathbf{d}} \pmod N, B_{m-1} = B_m^{\mathbf{d}} \pmod N = e(g,g)^{\mathbf{d}^2} \pmod N, \ldots, B_1 = B_2^{\mathbf{d}} \pmod N = e(g,g)^{\mathbf{d}^m} \pmod N$. It then randomly chooses $g \in G_{p_1}$ and for each attribute i, it chooses a random $s_i \in \mathbb{Z}_N$. The public parameters are then set to be $\mathsf{PP} = (g, B_1 = e(g,g)^{\mathbf{d}^m}, (\mathbf{e}, N), T_i = g^{s_i}\ \forall i \in [n])$. The master secret keys are $\mathsf{MK} = d_j$ for all $1 \leq j \leq m$ and a generator X_3 of G_{p_3}.

- $\mathsf{Enc}(\mathsf{M}, \mathsf{S}, \mathsf{PP})$: To encrypt a file with m messages, the encryption algorithm chooses a random $s \in \mathbb{Z}_N$ which will be used across all messages. For each message $M_j \in \mathsf{M}$ with attribute set $S_j \in \mathsf{S}$, the corresponding ciphertext component is set to be

$$C_j = M_j e(g,g)^{d_j s} = M_j e(g,g)^{\mathbf{d}^{m-j+1}s}$$

where $e(g,g)^{\mathbf{d}^{m-j+1}}$ is computed from $B_1 = e(g,g)^{\mathbf{d}_m}$ and $(\mathbf{e}, N)$. The algorithm also computes $C_0 = g^s, \tilde{C}_i = T_i^s \ \forall i \in [n]$. The ciphertext set CT for m messages is thus

$$C_j = M_j e(g,g)^{d_j s},\ C_0 = g^s,\ \tilde{C}_i = T_i^s \ \ \forall i \in [n], j \in [m].$$

Note that strictly speaking, the ciphertext set CT should contain only the $\tilde{C}_i$ values for $\forall i \in \bigcup S_j$ (where $\bigcup S_j \subseteq [n]$), instead of all attributes. However for notation simplicity we write $\forall i \in [n]$. This also applies to all ciphertexts formed in our security proof.

- $\mathsf{KeyGen}((A,\rho), \mathsf{MK}, \mathsf{PP})$: The key generation algorithm chooses random vectors u_j such that $1 \cdot u_j = d_j$. (Here, 1 denotes the vector with the first entry equals to 1 and the rest equal to 0). For each row A_x of A, it chooses a random $r_x \in \mathbb{Z}_N$, random elements $W_{x,j}, V_x \in G_{p_3}$ for $j \in [m]$, and sets the secret key sk to be

$$K^1_{x,j} = g^{A_x \cdot u_j} T^{r_x}_{\rho(x)} W_{x,j} \ \ \forall j \in [m],$$
$$K^2_x = g^{r_x} V_x.$$

Note that for each attribute x, the private key comprises j many K^1 components, while only one K^2 component.

- $\mathsf{Dec}(\mathsf{CT}, \mathsf{PP}, sk)$: Let S_j denotes the set of attributes associated with C_j and let (A,ρ) denote the matrix and row labeling associated with sk. If S_j satisfies A, the decryption algorithm computes constants ω_x such that $\sum_{\rho(x)\in S_j} \omega_x A_x = 1$. It then computes:

$$\prod_{\rho(x)\in S_j} \frac{e(C_0, K^1_{x,j})^{\omega_x}}{e(C_{\rho(x)}, K^2_x)^{\omega_x}} = \prod_{\rho(x)\in S_j} \frac{e(g,g)^{s\omega_x A_x \cdot u_j} e(g, T_{\rho(x)})^{s r_x \omega_x}}{e(g, T_{\rho(x)})^{s r_x \omega_x}}$$
$$= e(g,g)^{s \sum_{\rho(x)\in S_j} \omega_x A_x \cdot u_j} = e(g,g)^{s d_j}.$$

The message M_j is then recovered as $C_j / e(g,g)^{s d_j}$.

5.4.2 Efficiency and Trade-off

Since our scheme is based on Lewko et al.'s KP-ABE scheme [84], we analyze how much savings can be gained by our scheme in comparison with a trivial combination of m instances of Lewko at al.'s KP-ABE scheme.

- Ciphertext size: In our scheme, the size of the ciphertext set for m messages is $m + n + 1$, where m is the number of C_j, n the number of T_i^s values and one g^s component. On the other hand, when using m parallel KP-ABE schemes, the size of the ciphertext set is $2m + \sum_{j \in [m]} |S_j|$, where $2m$ is the number of C_j and C_0 (since different random values s are used), and $|S_j|$ denotes the size of the attribute set for each message M_j, thus $\sum_{j \in [m]} |S_j|$ gives the total number of elements in the form of T_i^s. In the worst case where each attribute set $|S_j| = n - 1$ and differs by one attribute, the size of the ciphertext set becomes $2m + m(n-1) = m(n+1)$. Clearly our scheme requires much smaller storage for ciphertexts, especially when n is large.

- Public parameter size: Our scheme requires only one additional group element in the public parameters compared to the original KP-ABE scheme: the RSA decryption key $\mathbf{e}$ used for generating the corresponding blinding factors.

- Private key size: Let A be the access matrix of a user and k be the number of rows of A, then the private key size of the trivial method is $2k$, while ours is $mk + k = (m+1)k$, where mk is the number of the $K_{x,j}^1$ component and k is the number of the K_x^2 component.

Note that there exist recent proposals of ABE schemes that achieve constant-size ciphertexts, for example [7, 11, 74]. Among these, the KP-ABE scheme by Hohenberger and Waters [74] seems to have the smallest ciphertext size (i.e., only 3 group elements). However the trade-off is that their scheme expands the user private key size by a factor of n, where n is the total number of attributes in the entire system.

5.4.3 Security

Overview. We prove the security of our MM-KP-ABE scheme by adopting the dual system encryption technique by Waters [116]. In our proof, private keys and ciphertexts take two forms: normal or semi-

functional. A normal private key could decrypt a ciphertext, which is either normal or semi-functional; while a semi-functional private key can only decrypt a normal ciphertext. When using a semi-functional key to decrypt a semi-functional ciphertext, the decryption will fail. We then use a hybrid argument through a sequence of games to prove the security of our scheme. We first change the challenge ciphertext to semi-functional, then gradually change the private keys into semi-functional one by one. At the very last step, we change the semi-functional ciphertext into an encryption of a random message, in which the adversary has no advantage at all.

Our security proof largely follows that of Lewko et al's KP-ABE scheme [84], with some additional treatment for the RSA key rotation used in our system. Especially in the last transition, we require the exponent α given in the hard assumption to be the value after rotating forward a random RSA private key $\mathbf{d}$ for m times, instead of a randomly picked value.

Proofs. We first define semi-functional ciphertexts and keys.

Semi-functional Ciphertext A semi-functional ciphertext is formed as follows: let g_2 denote a generator of G_{p_2}, c a random exponent, and z_i are random values. Then:

$$C_0 = g^s g_2^c;\ \tilde{C}_i = T_i^s g_2^{cz_i}\ \ \forall i \in S.$$

Semi-functional Key We define a semi-functional key in two types. A semi-functional key of type 1 is defined as follows: choose a random vector u_2 and set $\delta_x = A_x \cdot u_2$. Additionally choose random exponents γ_x. The key is then defined as:

$$K_{x,j}^1 = g^{A_x \cdot u_j} T_{\rho(x)}^{r_x} W_{x,j} g_2^{\delta_x + \gamma_x z_{\rho(x)}},$$
$$K_x^2 = g^{r_x} V_x g_2^{\gamma_x}.$$

A semi-functional key of type 2 is formed by letting $\gamma_x = 0$ in the type 1 key. More precisely, type 2 key is:

$$K_{x,j}^1 = g^{A_x \cdot u_j} T_{\rho(x)}^{r_x} W_{x,j} g_2^{\delta_x},$$
$$K_x^2 = g^{r_x} V_x.$$

When a semi-functional key is used to decrypt a semi-functional ciphertext, there is an additional term $e(g_2, g_2)^{\sum_{\rho(x)\in S} c\omega_x \delta_x} = e(g_2, g_2)^{cu_2 \cdot 1}$ that hinders the decryption. However, when $u_2 \cdot 1 = 0$ the decryption will succeed. We call a semi-functional key with $u_2 \cdot 1 = 0$ a *nominally* semi-functional key. Also note that the z_i values are common to semi-functional ciphertext and semi-functional keys of type 1.

We now define the sequence of games used in our security proof. The first game, $Game_{real}$ is the real security game, i.e., the ciphertext and all keys are normal. In $Game_0$, all keys are normal, but the challenge ciphertext is semi-functional. In $Game_{k,1}$, the first $k-1$ keys are semi-functional of type 2, the k^{th} key is semi-functional of type 1, and keys after the k^{th} private key query are formed as normal keys. In $Game_{k,2}$, the first k keys are semi-functional of type 2 and the rest of the keys are normal (so in $Game_{q,2}$, all keys are semi-functional of type 2). In $Game_{final}$, all keys are semi-functional of type 2, and the challenge ciphertext is a semi-functional encryption of random messages. We prove the security through the following lemmas.

Lemma 6. *Suppose there exists a polynomial-time algorithm $\mathcal{A}$ such that $Game_{real}Adv_{\mathcal{A}} - Game_0Adv_{\mathcal{A}} = \epsilon$. Then there exists a polynomial-time algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 1.*

Proof. Algorithm $\mathcal{B}$ is given $\{g, X_3, T\}$ and it will simulate either $Game_{real}$ or $Game_0$ with $\mathcal{A}$. The public parameters are formed by first choosing exponents s_i randomly, then $\mathcal{B}$ chooses an RSA key pair $(\mathbf{e}, \mathbf{d})$. It also computes the corresponding d_j values using $\mathbf{d}$ and calculates $B_1 = e(g, g)^{\mathbf{d}_m}$. It then passes the public parameters $(g, B_1, \tilde{T}_i = g^{s_i}, (\mathbf{e}, N))$ to $\mathcal{A}$. $\mathcal{B}$ could respond all private key queries made by $\mathcal{A}$ through the normal key generation since $\mathcal{B}$ knows the master secret key.

To form the challenge ciphertext for a set of attribute sets, $\mathcal{B}$ implicitly sets s so that g^s is the part of T in the G_{p_1} subgroup (since T is the product of g^s and possibly an element of G_{p_2}):

$$C_j = M_j e(g, g)^{d_j s} = M_j e(g, T)^{d_j},\ C_0 = T,\ \tilde{C}_i = T^{s_i}\ \forall i \in [n],\ \forall j \in [m].$$

Note that this implicitly sets $z_i = s_i$. If $T = g^s$, this is a properly distributed normal ciphertext. If $T = g^s X_2$ (for some random $X_2 \in G_{p_2}$), then this is a properly distributed semi-functional ciphertext.

Thus $\mathcal{B}$ can use the output of $\mathcal{A}$ to break Assumption 1. □

Lemma 7. *Suppose there exists a polynomial-time algorithm $\mathcal{A}$ such that $Game_{k-1,2}Adv_{\mathcal{A}} - Game_{k,1}Adv_{\mathcal{A}} = \epsilon$. Then there exists a polynomial-time algorithm $\mathcal{B}$ with advantage negligibly close to ϵ in breaking Assumption 2.*

Proof. Algorithm $\mathcal{B}$ receives $\{g, X_3, g^s X_2, Y_2 Y_3, T\}$. It will simulate either $Game_{k-1,2}$ or $Game_{k_1}$ with $\mathcal{A}$. $\mathcal{B}$ first chooses an RSA key pair $(\mathbf{e}, \mathbf{d})$, it computes the corresponding d_j values from $\mathbf{d}$ and produces $B_1 = e(g,g)^{\mathbf{d}^m}$. It also chooses random values s_i and sets the public parameters as $(g, B_1, \tilde{T}_i = g^{s_i}, (\mathbf{e}, N))$ and passes to $\mathcal{A}$.

To form the challenge ciphertext for a set of attribute sets, $\mathcal{B}$ sets:

$$C_j = M_j e(g, g^s X_2)^{d_j},\ C_0 = g^s X_2,\ \tilde{C}_i = (g^s X_2)^{s_i}\ \ \forall i \in [n],\ \forall j \in [m].$$

Note that this implicitly sets $s_i = z_i$.

To form the normal keys for the queries after the k^{th} private key query, $\mathcal{B}$ uses its knowledge of MK and runs the regular key generation algorithm. To create semi-functional keys of type 2 for the queries before the k^{th} private key query, $\mathcal{B}$ chooses random vectors u_j such that $u_j \cdot 1 = d_j$, a random vector u'_2, random values r_x and random group elements $W_{x,j}, V_x \in G_{p_3}$. Then the semi-functional key is defined as:

$$K^1_{x,j} = g^{A_x \cdot u_j} \tilde{T}^{r_x}_{\rho(x)} W_{x,j} (Y_2 Y_3)^{A_x \cdot u'_2},$$
$$K^2_x = g^{r_x} V_x.$$

Note here that for $Y_2 = g_2^c$ for some random c, the u_2 value in our description of semi-functional keys above corresponds to $u_2 = cu'_2$.

For the k^{th} key, $\mathcal{B}$ will make it either normal or *nominally* semi-functional of type 1, depending on the T value given in the challenge. $\mathcal{B}$ chooses random vector u_2 such that $u_2 \cdot 1 = 0$ (this is to form the nominal semi-functional key), and random vectors u'_j such that $u'_j \cdot 1 = d_j$. It implicitly sets

$u_j = ru_2 + u'_j$, where g^r is the G_{p_1} part of T. It also randomly chooses $\gamma_x, W_{x,j}, V_x$ and computes:

$$K^1_{x,j} = g^{A_x \cdot u'_j} T^{A_x \cdot u_2 + \gamma_x z_{\rho(x)}} W_{x,j},$$
$$K^2_x = T^{\gamma_x} V_x.$$

Note that this sets $r_x = r\gamma_x$. Depending on the value of T, this is either a normal key of a nominally semi-functional key of type 1.

We next argue that, if the k^{th} key is a nominally semi-functional key, then this key will still have the same distribution as with a regular semi-functional key of type 1 from the adversary's view. The argument is the same as in [84, Section A.1.3]. Hence, depending on the value T, $\mathcal{B}$ has properly simulated either $Game_{k-1,2}$ or $Game_{k,1}$ with probability negligibly close to 1, thus it can use the output of $\mathcal{A}$ to gain an advantage negligibly close to ϵ in breaking Assumption 2. □

Lemma 8. *Suppose there exists a polynomial-time algorithm $\mathcal{A}$ such that $Game_{k,1}Adv_{\mathcal{A}} - Game_{k,2}Adv_{\mathcal{A}} = \epsilon$. Then there exists a polynomial-time algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 2.*

Proof. Algorithm $\mathcal{B}$ is given $\{g, X_3, g^s X_2, Y_2 Y_3, T\}$. It will simulate either $Game_{k,1}$ or $Game_{k,2}$ with $\mathcal{A}$. $\mathcal{B}$ first chooses random s_i values, an RSA key pair $(\mathbf{e}, \mathbf{d})$, computes the corresponding d_j values and calculates $B_1 = e(g,g)^{\mathbf{d}_m}$. $\mathcal{B}$ then passes the public parameters $(g, B_1, (\mathbf{e}, N), \tilde{T}_i = g^{s_i})$ to $\mathcal{A}$. This implicitly sets $z_i = s_i$.

To form the challenge ciphertext for a set of attribute sets, $\mathcal{B}$ sets:

$$C_j = M_j e(g^s X_2, g)^{d_j},\ C_0 = g^s X_2,\ \tilde{C}_i = (g^s X_2)^{s_i}\ \ \forall i \in [n],\ \forall j \in [m].$$

To form the normal keys for the queries after the k^{th} private key query, $\mathcal{B}$ can produce them through regular key generation algorithm, with the knowledge of MK. To create semi-functional keys of type 2 for the queries before the k^{th} private key query, $\mathcal{B}$ first chooses random vectors u_j such that $u_j \cdot 1 = d_j$, a random vector u'_2, random values r_x, and random group elements $W_{x,j}, V_x \in G_{p_3}$. The semi-functional

keys of type 2 can be defined as:

$$K^1_{x,j} = g^{A_x \cdot u_j} \tilde{T}^{r_x}_{\rho(x)} W_{x,j} (Y_2 Y_3)^{A_x \cdot u'_2},$$
$$K^2_x = g^{r_x} V_x.$$

Note here that for $Y_2 = g_2^c$, the u_2 component which appears in the description of semi-functional key corresponds to $u_2 = cu'_2$.

For the k^{th} key, it will be either semi-functional of type 1 or type 2, depending on the value T given in the assumption. $\mathcal{B}$ first chooses random vectors u_j such that $u_j \cdot 1 = d_j$ and a random vector u_2. $\mathcal{B}$ also chooses random values γ_x, this will implicitly set $r_x = r\gamma_x$, where r appears in g^r and g^r is the G_{p_1} part of T. Lastly, $\mathcal{B}$ also chooses random $W_{x,j}, V_x \in G_{p_3}$. The k^{th} key is then formed as:

$$K^1_{x,j} = g^{A_x \cdot u_j} (Y_2 Y_3)^{A_x \cdot u_2} T^{\gamma_x s_{\rho(x)}} W_{x,j},$$
$$K^2_x = T^{\gamma_x} V_x.$$

If $T \in G$, then this is a properly distributed semi-functional key of type 1. If $T \in G_{p_1 p_3}$, this will be a properly distributed semi-functional key of type 2. Thus $\mathcal{B}$ uses the output of $\mathcal{A}$ to gain advantage in breaking Assumption 2. □

Lemma 9. *Suppose there exists a polynomial-time algorithm $\mathcal{A}$ such that $Game_{q,2} Adv_{\mathcal{A}} - Game_{final} Adv_{\mathcal{A}} = \epsilon$. Then there exists a polynomial-time algorithm $\mathcal{B}$ with advantage negligibly close to ϵ in breaking Assumption 3.*

Proof. The algorithm $\mathcal{B}$ first receives $\{g, g^{\mathbf{d}^m} X_2, X_3, g^s Y_2, Z_2, T\}$ and an additional RSA public key $\mathbf{e}$. Note here instead of the traditional $g^\alpha X_2$, $\mathbf{d}^m$ must be embedded into the assumption for successful generation of private keys. However we argue that, since both $\mathbf{d}^m$ and α are embedded as exponents and further randomized by X_2, $g^{\mathbf{d}^m} X_2$ and $g^\alpha X_2$ will have the same distribution. Additionally, the RSA public key $\mathbf{e}$ is given to $\mathcal{B}$. Here T is either $e(g,g)^{\mathbf{d}^m s}$ or a random group element from G_T.

$\mathcal{B}$ will simulate either $Game_{q,2}$ or $Game_{final}$ with $\mathcal{A}$. It sets the public parameters as

$$(g, e(g, g^{\mathbf{d}^m} X_2), (\mathbf{e}, N), \tilde{T}_i = g^{s_i})$$

where s_i are chosen randomly, and implicitly set as $s_i = z_i$. $\mathcal{B}$ will also calculate $T_j = T^{\mathbf{e}^{j-1}}\ \forall j \in [m]$ to form the corresponding blinding factors for each message.

To form the challenge ciphertext for a set of attribute sets, $\mathcal{B}$ computes:

$$C_j = M_j T_j,\ C_0 = g^s Y_2,\ \tilde{C}_i = (g^s Y_2)^{s_i}\ \ \forall i \in [n],\ \forall j \in [m].$$

If $T = e(g,g)^{\mathbf{d}^m s}$, then this will be a semi-functional encryption of M_b. If T is random, then this will be a semi-functional encryption of a random message set and will leak no information of b to the adversary.

To form semi-functional key of type 2 for an access structure (A, ρ) (where A has w rows), for each $j \in [m]$, $\mathcal{B}$ chooses $u_1^j, \ldots, u_{w-1}^j$ randomly and implicitly sets $u_w^j = d_j - \sum_{i=1}^{w-1} u_i^j$ (note here $\mathcal{B}$ does not know the value of d_j, it implicitly sets u_w^j using the value $g^{\mathbf{d}^m} X_2$). $\mathcal{B}$ also chooses a random vector u_2 of length w. It chooses random r_x and $W_{x,j},\ V_x$ and sets:

$$K_{x,j}^1 = g^{\sum_{i=1}^{w-1}(A_{x,i} - A_{x,w})u_i^j} (g^{d_j} X_2)^{A_{x,w}} \tilde{T}_{\rho(x)}^{r_x} Z_2^{A_x \cdot u_2} W_{x,j},$$
$$K_x^2 = g^{r_x} V_x.$$

Note here the value $g^{d_j} X_2$ is obtained through $g^{\mathbf{d}^m} X_2$ and $(\mathbf{e}, N)$. The above formation gives a properly distributed semi-functional key of type 2. $\mathcal{B}$ can use the output of $\mathcal{A}$ in breaking Assumption 3. □

We now prove the following theorem:

Theorem 7. *If Assumptions 1, 2 and 3 hold, then our scheme is secure.*

Proof. If Assumptions 1, 2 and 3 hold, then by previous lemmas we have shown that the real security game is indistinguishable from $Game_{final}$, where the adversary has no advantage at all. Hence the adversary has a negligible advantage in breaking our scheme. □

5.5 Summary

In this chapter, we studied the problem of how to minimize the ciphertext storage requirement for encryption of multiple messages under different access policies. This is particularly useful when cryptographic access control needs to be enforced on encrypted data, which in turn, is stored on the cloud. We

proposed the concept of multi-message broadcast encryption (MM-BE) and multi-message key-policy attribute-based encryption (MM-KP-ABE), and their respective security models. Moreover, we provided two concrete constructions based on BE and KP-ABE respectively. We show that our schemes achieve significant ciphertext savings compared to a naïve approach with some reasonable trade-offs.

There are two immediate open problems. The first is to further reduce the user private key size while maintaining the current ciphertext size savings. The second is to achieve constant-size ciphertexts, such as that of the multi-channel BE scheme proposed in [99] but under our more general setting. That is, the encryptor can be any user in the system, and different messages can be encrypted under different (and potentially overlapping) access policies.

Chapter 6

Spatial Encryption Supporting Non-Monotone Access Structure

Contents

In this chapter, we investigate a variant of spatial encryption (SE) we call ciphertext-policy SE (CP-SE), which combines the properties of SE and those from ciphertext-policy attribute-based encryption (CP-ABE). We present techniques for generic construction of CP-SE from ciphertext-policy inner product encryption (CP-IPE). Our techniques are property-preserving in the sense that if the CP-IPE scheme from which we derive our CP-SE scheme is fully secure, then so is the resulting CP-SE scheme. Moreover, interestingly, we show that it is possible to perform transformation of the opposite direction, that is, how to construct a CP-IPE scheme given a CP-SE scheme.

6.1 Introduction

6.1.1 Motivation

One ongoing research direction in identity-based encryption (IBE) [27, 48] is to generalize the notion of identities (or identifiers). Instead of encrypting under an identity and allowing decryption by a secret key related to the identity, one could encrypt under a policy (or attribute) and decrypt using a secret key related to one or more roles (or predicates) that satisfy the policy. Attribute-based encryption (ABE) [64, 100, 22, 84], inner product encryption (IPE) [82, 95], and spatial encryption (SE) [30, 68] are some instantiations of such generalization. These are also known as functional encryption (FE) [32].

One attractive property of SE is that it is a modular IBE system that can be adapted to many different purposes. We can embed in SE many other types of IBE systems, such as, hierarchical IBE (HIBE), inclusive IBE, co-inclusive IBE, broadcast HIBE and forward-secure IBE. As a consequence, we can construct a "product" scheme by embedding multiple instances of SE to obtain additional system

properties.

In this chapter, we extend the study of SE. Particularly, we inject some flavor of cryptographic access control into SE. We achieve this by enriching SE with a non-monotone access structure [97] to ciphertexts. This way, a user's secret key can be associated with an access structure over some attributes and that reflects the access policy ascribed to the user. We call our access-structure endowed SE *ciphertext-policy SE* (CP-SE), analogous to ciphertext-policy ABE (CP-ABE) [22, 117]. As with the case of CP-ABE, we can handle any access structure that can be represented by a boolean formula involving `AND`, `OR`, `NOT`, and threshold operations.

Combining the flexibility of SE and the expressiveness of CP-ABE, we can have many useful applications. We provide two applications of CP-SE in the following and more in Section 6.6:

- Forward-secure ABE: It is well-known that HIBE can be used to construct a forward-secure public-key encryption scheme [35]. Since HIBE is trivially embeddable in SE, forward-secure encryption is also embeddable in SE. In the latter, policies are timestamps associated to secret keys and roles are intervals of timestamps. With CP-SE, we can now embed hierarchical ABE (HABE) in it. This implies that we can treat the interval of a timestamp as one of many attributes that we may have in an access structure, giving us forward-secure ABE. To decrypt a ciphertext, a user must possess the key corresponding not only to the correct interval, but also some other attributes, for example, the user must be a member of the IEEE Computer Society `AND` a Ph.D student.

- Encrypted emails: SE can efficiently address the problem of sending an encrypted email to multiple recipients who may trust different private key generators (PKG) [30]. When sending an email, a user encrypts to the email addresses of the intended recipients and chooses the PKGs trusted by those recipients. The encrypted email is generated using the product between broadcast HIBE (for recipients) and broadcast IBE (for PKGs). Using CP-SE, we provide an additional dimension to the described solution for email encryption. That is, we can now impose the requirement of having multiple attributes, with the correct email address being one of them, for successful decryption of an encrypted email.

6.1.2 Our Approach and Contribution

In spatial encryption (SE), a secret key is associated with an affine space $\mathcal{S}$ in $\mathbb{Z}_q^n$, while a ciphertext is associated with a vector $\vec{x}$ in $\mathbb{Z}_q^n$ for some integer n and prime q. The secret key could decrypt the ciphertext iff $\vec{x} \in \mathcal{S}$. In our work, we define CP-SE in a natural way. In our definition, a secret key, as with SE, is associated with an affine space $\mathcal{S}$. However, a ciphertext is associated with a non-monotone access structure $\mathcal{A}$, that is a span program $\mathcal{M}$ with attribute vectors $\{\vec{x}_1, \ldots, \vec{x}_a\}$. The span program $\mathcal{M}$ takes as input the component-wise inclusion by space relations for each attribute vector, for example, $\{\vec{x}_j \in \mathcal{S} \text{ or not }\}_{j\in[a]}$. We then require that a ciphertext for $\mathcal{A}$ be decryptable by a secret key for $\mathcal{S}$ iff the truth-value vector of $(\gamma(\vec{x}_1 \in \mathcal{S} \text{ or not}), \ldots, \gamma(\vec{x}_a \in \mathcal{S} \text{ or not}))$ is accepted by $\mathcal{M}$. Here $\gamma(\psi)$ is a predicate function outputting 1 if ψ is true, or 0 otherwise.

We encountered some difficulties when trying to *directly* construct a provably secure CP-SE scheme. The known proof techniques for ABE [22, 117] and SE [30], respectively, do not seem to be applicable to our CP-SE setting, even for the case of selective security and a scheme that supports only monotone access structure. This is not surprising due to the expressiveness and flexibility of the access structure and vector attributes in CP-SE. We have also considered the dual system proof techniques for a fully secure ABE scheme from Lewko et al. [84]. However, key delegation in CP-SE complicates the relevant security arguments and adoption of the proof techniques of [84] do not seem to work. Moreover, we observed that even if it is possible to construct CP-SE that supports monotone access structure, it is not clear how one could enforce negation in SE (to obtain non-monotone access structure), i.e., a ciphertext for a vector $\vec{x}$ can be decrypted by a secret key for a space $\mathcal{S}$ iff $\vec{x} \notin \mathcal{S}$, and achieve full security at the same time. We note that the SE scheme supporting negation by Attrapadung and Libert [10] is only co-selective secure and it imposes some restrictions on the associated spaces.

Hence, instead of giving a concrete construction of CP-SE directly, we discovered that it is possible to derive a fully secure CP-SE scheme from a ciphertext-policy IPE (CP-IPE) scheme [96], which allows inner products of attribute vectors. The challenge of this is to embed attributes and the access structure required by CP-SE in CP-IPE such that one can obtain a secret share (associated with an attribute within a span program) required by CP-SE if and only if it can be obtained from CP-IPE. Note that our CP-SE is defined very differently from CP-IPE. The former is more natural and concise where an attribute is

expressed simply by a vector. On the other hand, the latter makes use of more complicated structure where each attribute is expressed by a pair of sub-universe identity and variable n-dimensional vector (see Definition 23).

Our main result in this work is a generic construction of CP-SE from CP-IPE. Our transformation techniques are property-preserving. In other words, if we derive a CP-SE scheme from a CP-IPE scheme that is fully secure under a simple assumption and in prime order bilinear groups, for example the scheme of Okamoto and Takashima [96], then the resulting CP-SE scheme inherits similar properties. We note that our transformation techniques (from CP-IPE to CP-SE) is related to but different from the techniques used in [41]. Here, we deal with non-monotone access structure, while the work of [41] is about generic construction from SE to hierarchical IPE (HIPE), vice versa.

As a side contribution, and interestingly, we also show that it is possible to construct CP-IPE from CP-SE. This implies that CP-IPE and CP-SE are equivalent (under some reduction).

6.1.3 Outline

We organize the rest of this chapter as follows. In Section 6.2 & 6.3, we provide the preliminaries, the definitions and security models of CP-SE and CP-IPE. In Section 6.4, we present generic construction of CP-SE from CP-IPE, followed by the construction of CP-IPE from CP-SE in Section 6.5. In Section 6.6, we give some applications of CP-SE. We conclude this chapter in Section 6.7.

6.2 Preliminaries

6.2.1 Span Programs and Non-Monotone Access Structures

We first describe the concept of span programs typically required by ABE.

Definition 16 (Span Programs [18]). *Let $\{p_1, \ldots, p_n\}$ be a set of variables. A span program over $\mathbb{Z}_q$ is a labeled matrix $\mathcal{M} := (M, \rho)$ where M is an $(a \times b)$ matrix over $\mathbb{Z}_q$ and ρ is a labeling of the rows of M by literals from $\{p_1, \ldots, p_n, \neg p_1, \ldots, \neg p_n\}$ (every row is labeled by one literal), i.e., $\rho : [a] \rightarrow \{p_1, \ldots, p_n, \neg p_1, \ldots, \neg p_n\}$.*

A span program accepts or rejects an input by the following criterion. For every input sequence

$\delta \in \{0,1\}^n$ define the submatrix M_δ of M consisting of those rows whose labels are set to 1 by the input, i.e., either rows labeled by some p_t such that $\delta_t = 1$ or rows labeled by some $\neg p_t$ such that $\delta_t = 0$. (i.e., $\gamma : [a] \to \{0,1\}$ is defined by $\gamma(j) = 1$ if $[\rho(j) = p_t] \wedge [\delta_t = 1]$ or $[\rho(j) = \neg p_t] \wedge [\delta_t = 0]$, and $\gamma(j) = 0$ otherwise. Let $M_\delta := (M_j)_{\gamma(j)=1}$, where M_j is the j-th row of M.)

The span program $\mathcal{M}$ accepts δ if and only if $\vec{I} \in \mathsf{span}\langle M_\delta \rangle$, i.e., some linear combination of the rows of M_δ gives the all one vector $\vec{I}$, where $\vec{I} = (1, 0, \ldots, 0)$. A span program computes a Boolean function f if it accepts exactly those inputs δ where $f(\delta) = 1$.

A span program is called monotone if the labels of the rows are only the positive literals $\{p_1, \ldots, p_n\}$. Otherwise, it is non-monotone.

We now introduce the notion of a non-monotone access structure with evaluating map γ by using the inclusion by spaces of attribute vectors.

Definition 17 (Inclusion of attribute vectors and access structures)**.** *We define an attribute to be a variable p of a span program $\mathcal{M} := (M, \rho)$, i.e., a vector $p := \vec{x} \in \mathbb{Z}_q^n$. An access structure $\mathcal{A}$ is a span program $\mathcal{M} := (M, \rho)$ along with variables $p := \vec{x}$, $p' := \vec{x}'$, ..., i.e., $\mathcal{A} := (M, \rho)$ such that $\rho : [a] \to \{\vec{x}, \vec{x}', \ldots, \neg(\vec{x}), \neg(\vec{x}'), \ldots\}$. We define $\mathcal{S}$ to be an affine space in $\mathbb{Z}_q^n$.*

When $\mathcal{S}$ is given the access structure $\mathcal{A}$, map $\gamma : [a] \to \{0,1\}$ for span program $\mathcal{M} := (M, \rho)$ is defined as follows: For $j \in [a]$, set $\gamma(j) = 1$ if $[\rho(j) = \vec{x}_t] \wedge [\vec{x}_t \in \mathcal{S}]$ or $[\rho(j) = \neg(\vec{x}_t)] \wedge [\vec{x}_t \notin \mathcal{S}]$. Set $\gamma(j) = 0$ otherwise.

Access structure $\mathcal{A} := (M, \rho)$ accepts $\mathcal{S}$ iff $\vec{I} \in \mathsf{span}\langle (M_j)_{\gamma(j)=1} \rangle$.

We use the following secret-sharing scheme for a non-monotone access structure or span program.

Definition 18. *A secret-sharing scheme for a span program $\mathcal{M} := (M, \rho)$ is:*

- *Let M be $a \times b$ matrix. Let $\vec{f} := (s, f_2, \ldots, f_b) \xleftarrow{\$} \mathbb{Z}_q^b$. Then, $s := \vec{I} \cdot \vec{f}^\top$ is the secret to be shared, and $\vec{s}^\top := (s_1, \ldots, s_a)^\top := M \cdot \vec{f}^\top$ is the vector of a shares of the secret s and the share s_j belongs to $\rho(j)$.*

- *If the span program $\mathcal{M} := (M, \rho)$ accept δ, or access structure $\mathcal{A} := (M, \rho)$ accepts $\mathcal{S}$, i.e., $\vec{I} \in \mathsf{span}\langle (M_j)_{\gamma(j)=1} \rangle$ with $\gamma : [a] \to \{0,1\}$, then there exist constants $\{\alpha_j \in \mathbb{Z}_q | j \in I\}$ such*

that $I \subseteq \{j \in [a] | \gamma(j) = 1\}$ *and* $\Sigma_{j \in I} \alpha_j s_j = s$. *Furthermore, these constants* $\{\alpha_j\}$ *can be computed in time polynomial in the size of matrix* M.

6.3 Primitive Definitions

6.3.1 Ciphertext-Policy Spatial Encryption

In what follows, we first borrow the definition and security model of functional encryption (FE) from [32]. We then, using similar syntax, provide the definition and security model for CP-SE. As in [32], we first describe a functionality $\mathcal{F}$ of the syntactic definition of FE. The functionality $\mathcal{F}$ describes the functions of a plaintext that can be learned from the ciphertext:

Definition 19. *A functionality* $\mathcal{F}$ *defined over* $(\mathcal{K}, \mathcal{X})$ *is a function* $\mathcal{F} : \mathcal{K} \times \mathcal{X} \rightarrow \{0,1\}^*$ *described as a (deterministic) Turing Machine. The set* $\mathcal{K}$ *is called the key space and the set* $\mathcal{X}$ *is called the plaintext space. We require that the key space* $\mathcal{K}$ *contains a special key called the empty key denoted* ϵ.

An FE scheme for the functionality $\mathcal{F}$ enables one to evaluate $\mathcal{F}(k, x)$ given the encryption of x and a secret key sk_k for k. The algorithm for evaluation $\mathcal{F}(k, x)$ using sk_k is called *decrypt*. More precisely, an FE scheme is defined as follows:

Definition 20. *A functional encryption scheme (FE) for a functionality* $\mathcal{F}$ *defined over* $(\mathcal{K}, \mathcal{X})$ *is a tuple of four probabilistic polynomial-time (PPT) algorithms (*Setup, KeyGen, Enc, Dec*) satisfying the following correctness condition for all* $k \in \mathcal{K}$ *and* $x \in \mathcal{X}$*:*

$(\mathsf{PP}, \mathsf{MK}) \leftarrow \mathsf{Setup}(1^k)$	(*generate a public and master secret key pair*)
$\mathsf{sk}_k \leftarrow \mathsf{KeyGen}(\mathsf{PP}, \mathsf{MK}, k)$	(*generate a secret key for* k)
$c \leftarrow \mathsf{Enc}(\mathsf{PP}, x)$	(*encrypt plaintext* x)
$y \leftarrow \mathsf{Dec}(\mathsf{PP}, \mathsf{sk}_k, c)$	(*use* sk_k *to compute* $\mathcal{F}(k, x)$ *from* c)

then we require that $y = \mathcal{F}(k, x)$ *with probability* 1.

The empty key ϵ: The special key ϵ in $\mathcal{K}$ captures all the information about the plaintext that intentionally leaks from the ciphertext. Henceforth, we assume that every FE scheme contains the empty key ϵ

in the key space $\mathcal{K}$ and we will not explicitly mention it.

We now define the security model for FE. For the plaintext pair $(x_{(0)}, x_{(1)})$ of an attacker's choice, we need the following requirement to make the experiment non-trivial:

$$\mathcal{F}(k, x_{(0)}) = \mathcal{F}(k, x_{(1)}) \text{ for all } k \text{ for which the attacker has } \mathsf{sk}_k. \tag{6.1}$$

Then we define a security game for an FE scheme as follows:

Definition 21. *For $\beta = 0, 1$ define an experiment β for an adversary $\mathcal{A}$ as follows:*

- Setup*: It runs* $(\mathsf{PP}, \mathsf{MK}) \leftarrow \mathsf{Setup}(1^k)$ *and gives* PP *to* $\mathcal{A}$.
- Query*: $\mathcal{A}$ adaptively submits queries k_i in $\mathcal{K}$ for $i = 1, 2, \ldots$ and in return, it receives* $\mathsf{sk}_{k_i} \leftarrow \mathsf{KeyGen}(\mathsf{PP}, \mathsf{MK}, k_i)$.
- Challenge*: $\mathcal{A}$ submits two plaintexts $x_{(0)}$, $x_{(1)} \in \mathcal{X}$ satisfying requirement (6.1) and in return, it receives* $\mathsf{Enc}(\mathsf{PP}, x_{(\beta)})$.
- Guess*: $\mathcal{A}$ continues to issue key queries as before subject to requirement (6.1) and eventually outputs a bit in* $\{0, 1\}$.

For $\beta = 0, 1$ let W_β be the event that the adversary outputs 1 *in Experiment β and define*

$$\mathsf{Adv}^{FE}_{\mathcal{A}}(\lambda) := |Pr[W_0] - Pr[W_1]|.$$

Definition 22. *An FE scheme is secure if for all PPT adversaries $\mathcal{A}$ the function $\mathsf{Adv}^{FE}_{\mathcal{A}}(\lambda)$ is negligible.*

We now define CP-SE. In CP-SE, a plaintext $x \in \mathcal{X}$ is itself a pair $(\mathsf{ind}, m) \in \mathcal{I} \times \mathcal{M}$ where ind is called an index and m is called the payload message.

In an n-dimensional CP-SE scheme supporting non-monotone access structure, a functionality $\mathcal{F}$ is defined over a key space and an index space using affine spaces and non-monotone access structure respectively as defined in Definition 17. The key space $\mathcal{K}$ corresponds to all affine spaces $\mathcal{S}$, while the index space $\mathcal{I}$ corresponds to all non-monotone access structures $\mathcal{A} := (M, \rho)$. Here,

$$\mathcal{F}(\mathcal{S},(\mathcal{A},m)) := \begin{cases} m & \text{if } \mathcal{A} := (M,\rho) \text{ accepts } \mathcal{S} \\ \perp & \text{otherwise.} \end{cases}$$

Let $x_{(0)} = (\mathsf{ind}_{(0)}, m_{(0)}), x_{(1)} = (\mathsf{ind}_{(1)}, m_{(1)}) \in \mathcal{X}$ be the adversary's choice of plaintext pair. The security game for CP-SE can then be defined using Definition 21 with the following variations:

- If the adversary outputs challenge indices $\mathsf{ind}_{(0)}, \mathsf{ind}_{(1)}$ before the Setup phase, the security game is then under the *selective security* model. Otherwise it is under the *full security* model.
- If the adversary outputs challenge indices such that $\mathsf{ind}_{(0)} = \mathsf{ind}_{(1)}$, the security game is then under the *payload-hiding* security model, that is $\mathcal{F}(\epsilon,(\mathsf{ind},m)) = (\mathsf{ind},|m|)$. Otherwise it is under the *attribute-hiding* security model, that is $\mathcal{F}(\epsilon,(\mathsf{ind},m)) = |m|$.

Delegation: This is an additional, but optional PPT algorithm that is denoted by Del. It takes as input a secret key sk_k for $k \in \mathcal{K}$ and outputs another secret key $\mathsf{sk}_{k'}$ for $k' \in \mathcal{K}$ that satisfies a partial order relation denoted as $k' \preceq k$. We note that the key delegation queries must also satisfy requirement (6.1) in the security definition of the FE system. One can also add a delegation mechanism to CP-SE in a similar way to SE. In CP-SE, the partial relation is represented by the subspace relation, i.e., $\mathcal{S}' \preceq \mathcal{S}$ iff $\mathcal{S}'$ is a subspace of $\mathcal{S}$. Note that delegable CP-SE can support only monotone access structure, since a secret key for a space can always delegate to another secret key for a subspace which avoids some specific vector.

6.3.2 Ciphertext-Policy Inner Product Encryption

We use the similar syntax of FE to recall the definition and security model for CP-IPE [96]. We first give the notion of a non-monotone access structure with evaluating map γ by using inner-products of attribute vectors.

Definition 23 (Inner products of attribute vectors and access structures [96])**.** *$\mathcal{U}_i$ ($t = 1,\ldots,d$ and $\mathcal{U}_t \subset \{0,1\}^*$) is a sub-universe, a set of attributes, each of which is expressed by a pair of sub-universe*

id and n_i-dimensional vector, i.e., $(i, \vec{v})$, where $i \in [d]$ and $\vec{v} \in \mathbb{Z}_q^{n_i} \backslash \{\vec{0}\}$. We denote such structure as $\vec{n} := (d; n_1, \ldots, n_d)$.

We define such an attribute to be a variable p of a span program $\mathcal{M} := (M, \rho)$, i.e., $p := (i, \vec{x})$. An access structure $\mathcal{A}$ is a span program $\mathcal{M} := (M, \rho)$ along with variables $p := (i, \vec{x})$, $p' := (i', \vec{x}')$, ..., i.e., $\mathcal{A} := (M, \rho)$ such that $\rho : [a] \to \{(i, \vec{x}), (i', \vec{x}'), \ldots, \neg(i, \vec{x}), \neg(i', \vec{x}'), \ldots\}$.

Let Π be a set of attributes, i.e., $\Pi := \{(i, \vec{v}_i) | \vec{v}_i \in \mathbb{Z}_q^{n_i} \backslash \{\vec{0}\}, i \in T\}$, where T is a subset of $[d]$.

When Π is given the access structure $\mathcal{A}$, map $\gamma : [a] \to \{0, 1\}$ for span program $\mathcal{M} := (M, \rho)$ is defined as follows: For all $j \in [a]$, set $\gamma(j) = 1$ if $[\rho(j) = (i, \vec{x}_t)] \wedge [(i, \vec{v}_i) \in \Pi] \wedge [\vec{x}_t \cdot \vec{v}_i = 0]$ or $[\rho(j) = \neg(i, \vec{x}_t)] \wedge [(i, \vec{v}_i) \in \Pi] \wedge [\vec{x}_t \cdot \vec{v}_i \neq 0]$. Set $\gamma(j) = 0$ otherwise.

Access structure $\mathcal{A} := (M, \rho)$ accepts Π iff $\vec{I} \in \mathsf{span}\langle (M_j)_{\gamma(j)=1} \rangle$.

In a CP-IPE scheme supporting non-monotone access structure, a functionality $\mathcal{F}$ is defined over a key space and an index space using affine spaces and non-monotone access structure, respectively (see Definition 23). The key space $\mathcal{K}$ corresponds to all attribute sets Π, while the index space $\mathcal{I}$ corresponds to all non-monotone access structures $\mathcal{A} := (M, \rho)$. Here,

$$\mathcal{F}(\Pi, (\mathcal{A}, m)) := \begin{cases} m & \text{if } \mathcal{A} := (M, \rho) \text{ accepts } \Pi \\ \perp & \text{otherwise.} \end{cases}$$

We may allow a vector $\vec{v}_i$ of an attribute set $\Pi := \{(i, \vec{v}_i) | \vec{v}_i \in \mathbb{Z}_q^{n_i} \backslash \{\vec{0}\}, i \in T\}$ to be a wildcard vector $\vec{v}_i^*$, i.e., $\vec{v}_i^*$ can be replaced by any vector in $\mathbb{Z}_q^{n_i} \backslash \{\vec{0}\}$ when we run the key delegation algorithm. Given $\Pi := \{(i, \vec{v}_i) | \vec{v}_i \in \mathbb{Z}_q^{n_i} \backslash \{\vec{0}\}, i \in T\}$ and $\Pi' := \{(i, \vec{v}_i) | \vec{v}_i \in \mathbb{Z}_q^{n_i} \backslash \{\vec{0}\}, i \in T'\}$, then $\Pi' \preceq \Pi$ iff $(i, \vec{v}_i) \in \Pi$ or $(i, \vec{v}_i^*) \in \Pi$ for all $(i, \vec{v}_i) \in \Pi'$.

6.3.3 Notation

In the remainder of the chapter, if not explicitly specified, we assume that all vectors are row vectors in $\mathbb{Z}_q^n$ for some integer n and prime q and spaces are Euclidean spaces spanned by row vectors. Table 6.1 summarizes some notation used in the remainder of this chapter.

Note that we use symbol M for a matrix of a span program and A for an affine space. Moreover, we use subscript to indicate the set type of a scheme. For example, we use $\mathcal{K}_{CP-SE}$ (resp. $\mathcal{I}_{CP-SE}$) to

Table 6.1: Notation.

$t \xleftarrow{\$} T$	t is chosen uniformly at random from a set T
$\mathcal{S}(\vec{v}_1, \ldots, \vec{v}_r)$	the space spanned by $\{\vec{v}_1, \ldots, \vec{v}_r\}$
$\mathcal{S}^{\perp}(\vec{v}_1, \ldots, \vec{v}_r)$	the orthogonal space of $\mathcal{S}(\vec{v}_1, \ldots, \vec{v}_r)$, i.e., the space spanned by all $\vec{x}$ where $\vec{x} \cdot \vec{v}_i = 0$ for $i \in [r]$
$\mathcal{B}(\mathcal{S})$	a basis of $\mathcal{S}$
$\mathcal{B}^{\perp}(\mathcal{S})$	a basis of $\mathcal{S}^{\perp}$
$\mathsf{dim}(\mathcal{S})$	the dimension of $\mathcal{S}$
A_i	the i-th row of matrix A
$\mathcal{S}(A, \vec{y})$	the affine space $\{\vec{z}A + \vec{y} : \vec{z} \in \mathbb{Z}_q^r\}$, where $A \in \mathbb{Z}_q^{r \times n}$

denote the key space $\mathcal{K}$ (resp. index space $\mathcal{I}$) with regards to CP-SE.

6.4 Generic Construction of CP-SE from CP-IPE

We first describe the construction of CP-SE that works in Euclidean spaces. We then explain in Section 6.4.5 how one can derive a CP-SE scheme in affine spaces from a CP-SE scheme in Euclidean spaces.

6.4.1 Concepts from Linear Algebra

The key generation and delegation algorithms of our generic construction are resemblant to the transformation from HIPE to SE of [41], thus we also require the following lemmata.

Lemma 10. *Given a space $\mathcal{S}$, then $\mathsf{dim}(\mathcal{S}) + \mathsf{dim}(\mathcal{S}^{\perp}) = n$ and $(\mathcal{S}^{\perp})^{\perp} = \mathcal{S}$.*

Lemma 11. *Given a space $\mathcal{S}$, there exists a polynomial-time algorithm* BasisGen *taking as input $\mathcal{S}$ and outputting a basis $\mathcal{B}^{\perp}$ of $\mathcal{S}^{\perp}$.*

Lemma 12. *Given a space $\mathcal{S}$, a subspace $\mathcal{S}'$ and a basis $\mathcal{B}^{\perp}$ of $\mathcal{S}^{\perp}$, there exists a polynomial-time algorithm* BasisDel *taking as input $\mathcal{S}, \mathcal{S}', \mathcal{B}^{\perp}$ and outputting a basis $\mathcal{B}'^{\perp}$ of $\mathcal{S}'^{\perp}$, which contains all the vectors of $\mathcal{B}^{\perp}$.*

6.4.2 Intuition

We are now ready to describe our idea of generic construction from CP-IPE to CP-SE. Intuitively, we need to embed attributes and an access structure required by our CP-SE scheme in the original CP-IPE scheme, such that one can obtain a secret share (associated with an attribute vector within a span program) required by the CP-SE scheme if and only if it can be obtained from the CP-IPE.

More specifically, in order to construct an n-dimensional CP-SE scheme, we require a CP-IPE scheme with structure $\vec{n} := (n-1; n, \ldots, n)$:

1. A secret key for an $(n-r)$-dimensional space $\mathcal{S} \in \mathcal{K}_{CP-SE}$ is generated using the KeyGen algorithm of the CP-IPE scheme. We embed a basis $\mathcal{B}^{\perp}(\mathcal{S}) = \{\vec{v}_1, \ldots, \vec{v}_r\}$ in the first r levels and pad $(n-1-r)$ attribute vectors in $\mathcal{B}^{\perp}(\mathcal{S})$ for the remaining levels. This allows us to extract the j-th secret share from a positive literal since we will divide the j-th secret share into $n-1$ parts when we encrypt a message. That is, we generate a secret key for the $(n-1)$ attribute vector set $\{(1, \vec{v}_1), \ldots, (r, \vec{v}_r), (r+1, \vec{v}_1), \ldots, (n-1, \vec{v}_1)\} \in \mathcal{K}_{CP-IPE}$ by using the KeyGen algorithm of the CP-IPE scheme. It is then treated as a secret key for the $(n-r)$-dimensional space $\mathcal{S} \in \mathcal{K}_{CP-SE}$.

2. To encrypt a message associated with a non-monotone access structure $\mathcal{A} := (M, \rho) \in \mathcal{K}_{CP-SE}$, we define a new non-monotone access structure $\mathcal{A}' := (M', \rho') \in \mathcal{I}_{CP-IPE}$. We embed each variable $\rho(j) = \vec{x}_t$ (resp. $\rho(j) = \neg(\vec{x}_t)$) of $\mathcal{A}$ in the $n-1$ variables

$$\{(1, \vec{x}_t), \ldots, (n-1, \vec{x}_t)\} \text{ (resp.}\{\neg(1, \vec{x}_t), \ldots, \neg(n-1, \vec{x}_t)\}) \text{ of } \mathcal{A}'.$$

   More precisely:

   - If $\rho(j) = \vec{x}_t$, we divide the j-th secret share into $n-1$ parts such that one can obtain the j-th secret share from $\rho(j) = \vec{x}_t$ iff she can obtain the j-th secret share from $\{(1, \vec{x}_t), \ldots, (n-1, \vec{x}_t)\}$. In other words, we require $\vec{x}_t \in \mathcal{S}$ iff $\vec{x}_t \cdot \vec{v}_i = 0$ for all $\vec{v}_i \in \mathcal{B}^{\perp}(\mathcal{S})$.
   - If $\rho(j) = \neg(\vec{x}_t)$, we use $n-1$ copies of the j-th secret share such that one can obtain the j-th secret share from $\rho(j) = \vec{x}_t$ iff she can obtain the j-th secret share from one of

$\{(1, \vec{x}_t), \ldots, (n-1, \vec{x}_t)\}$. In other words, we require $\vec{x}_t \notin \mathcal{S}$ iff $\vec{x}_t \cdot \vec{v}_i \neq 0$ for some $\vec{v}_i \in \mathcal{B}^{\perp}(\mathcal{S})$.

Note that since the dimension of the orthogonal space of $\mathcal{S} \in \mathcal{K}_{CP-SE}$ is at most $n-1$ (by Lemma 10), we require that the structure $\vec{n}$ has $d-1$ levels.

6.4.3 Construction

We now formally specify the construction of a CP-SE scheme from a CP-IPE scheme following the idea described before.

Given an affine space $\mathcal{S} \in \mathcal{K}_{CP-SE}$, we define a new attribute set $\Pi \in \mathcal{K}_{CP-IPE}$. We first run the $\mathsf{BasisGen}(\mathcal{S})$ algorithm of Lemma 11 to obtain $\mathcal{B}^{\perp}(\mathcal{S}) = \{\vec{v}_1, \ldots, \vec{v}_r\}$. We then set $\Pi := \{(i, \vec{v}_i), (j, \vec{v}_1) : i \in [r], j \in [r+1, n-1]\}$.

Given an access structure $\mathcal{A} := (M, \rho) \in \mathcal{I}_{CP-SE}$, we define a new access structure $\mathcal{A}' := (M', \rho') \in \mathcal{I}_{CP-IPE}$. Let $M \in \mathbb{Z}_q^{a \times b}$, we require that $M' \in \mathbb{Z}_q^{a' \times b'}$, where $a' = a(n-1)$ and $b' = a(n-2) + b$. For j from 1 to a, we set M' and ρ' as follows:

- If $\rho(j) = \vec{x}_t$, then $\rho'((j-1)(n-1)+k) = (k, \vec{x}_t)$ for all $k \in [n-1]$ and

$$\begin{pmatrix} M'_{(j-1)(n-1)+1} \\ M'_{(j-1)(n-1)+2} \\ \cdots \\ M'_{(j-1)(n-1)+n-2} \\ M'_{j(n-1)} \end{pmatrix} = \begin{pmatrix} M_j, 0, \ldots, 0, 1, 0, 0, \ldots, 0, 0, 0, \ldots, 0 \\ M_j, 0, \ldots, 0, 0, 1, 0, \ldots, 0, 0, 0, \ldots, 0 \\ \cdots \\ M_j, 0, \ldots, 0, 0, 0, 0, \ldots, 0, 1, 0, \ldots, 0 \\ M_j, 0, \ldots, 0, 1, 1, 1, \ldots, 1, 1, 0, \ldots, 0 \end{pmatrix}$$

  where 1's are on the $((j-1)(n-2)+1+b)$-th up to $(j(n-2)+b)$-th scalar. From the definition of M', it is not difficult to see that to obtain the j-th share, one should obtain all the $((j-1)(n-1)+1)$-th to $(j(n-1))$-th shares.

- If $\rho(j) = \neg(\vec{x}_t)$, then $\rho'((j-1)(n-1)+k) = \neg(k, \vec{x}_t)$ for all $k \in [n-1]$ and

$$\begin{pmatrix} M'_{(j-1)(n-1)+1} \\ M'_{(j-1)(n-1)+2} \\ \cdots \\ M'_{(j-1)(n-1)+n-2} \\ M'_{j(n-1)} \end{pmatrix} = \begin{pmatrix} M_j, 0, \ldots, 0 \\ M_j, 0, \ldots, 0 \\ \cdots \\ M_j, 0, \ldots, 0 \\ M_j, 0, \ldots, 0 \end{pmatrix}.$$

From the definition of M', it is not difficult to see that to obtain the j-th share, one should obtain one of the $((j-1)(n-1)+1)$-th to $(j(n-1))$-th shares.

Given a CP-IPE scheme with four algorithms: Setup_{CP-IPE}, KeyGen_{CP-IPE}, Enc_{CP-IPE} and Dec_{CP-IPE}, we construct a CP-SE scheme with the corresponding four algorithms: Setup_{CP-SE}, KeyGen_{CP-SE}, Enc_{CP-SE} and Dec_{CP-SE}, as follows:

$\mathsf{Setup}_{CP-SE}(1^k, n)$ runs $\mathsf{Setup}_{CP-IPE}(1^k, \vec{n})$ and outputs public parameters PP and a master key MK.

$\mathsf{KeyGen}_{CP-SE}(\mathsf{PP}, \mathsf{MK}, \mathcal{S})$ runs $\mathsf{KeyGen}_{CP-IPE}(\mathsf{PP}, \mathsf{MK}, \Pi)$ and outputs a secret key $\mathsf{sk}_{\mathcal{S}}$.

$\mathsf{Enc}_{CP-SE}(\mathsf{PP}, \mathcal{A}, m)$ runs $\mathsf{Enc}_{CP-SE}(\mathsf{PP}, \mathcal{A}', m)$ and outputs a ciphertext c.

$\mathsf{Dec}_{CP-SE}(\mathsf{PP}, \mathsf{sk}_{\mathcal{S}}, c)$ runs $\mathsf{Dec}_{CP-IPE}(\mathsf{PP}, \mathsf{sk}_{\mathcal{S}}, c)$ and outputs a message m.

We now show that the resulting CP-SE scheme works correctly and is indeed secure. Since our generic construction of CP-SE requires only a single instance of a CP-IPE scheme (with some additional efficient algorithms), we use techniques similar to those for the Embedding Lemma of [30, 68].

Theorem 8. *The CP-SE scheme constructed from the CP-IPE scheme works correctly.*

Proof. Given a plaintext $(\mathcal{A}, m) \in \mathcal{X}_{CP-SE}$ and an $(n-r)$-dimensional space $\mathcal{S} \in \mathcal{K}_{CP-SE}$, we transform them into a plaintext $(\mathcal{A}', m) \in \mathcal{X}_{CP-IPE}$ and an attribute set $\Pi := \{(i, \vec{v}_i), (j, \vec{v}_1) : i \in [r], j \in [r+1, n-1]\} \in \mathcal{K}_{CP-IPE}$ respectively, where $\mathcal{B}^{\perp}(\mathcal{S}) = \{\vec{v}_1, \ldots, \vec{v}_r\}$ is a basis of $\mathcal{S}^{\perp}$. From our construction, we only need to show that one can obtain a secret share required by the CP-SE scheme if and only if she can obtain it from the CP-IPE scheme. Hence, we require

- $[\rho(j) = \vec{x}_t] \wedge [\vec{x}_t \in \mathcal{S}]$
  iff $[\rho'((j-1)(n-1)+k) = (k, \vec{x}_t)] \wedge [(k, \vec{v}_k) \in \Pi] \wedge [\vec{x}_t \cdot \vec{v}_k = 0]$ for all $k \in [n-1]$,

- $[\rho(j) = \neg(\vec{x}_t)] \wedge [\vec{x}_t \notin \mathcal{S}]$
  iff $[\rho'((j-1)(n-1)+k) = \neg(k, \vec{x}_t)] \wedge [(k, \vec{v}_k) \in \Pi] \wedge [\vec{x}_t \cdot \vec{v}_k \neq 0]$ for some $k \in [n-1]$.

The above items are equivalent to

- $\vec{x}_t \in \mathcal{S}$ iff $\vec{x}_t \cdot \vec{v}_k = 0$ for all $k \in [r]$,

- $\vec{x}_t \notin \mathcal{S}$ iff $\vec{x}_t \cdot \vec{v}_k \neq 0$ for some $k \in [r]$.

This is clear from the definition of $\mathcal{B}^{\perp}(S)$. This implies that the resulting CP-SE scheme inherits the decryptability ($\mathcal{F}(\mathcal{S}, (\mathcal{A}, m)) := m$ iff $\mathcal{A} := (M, \rho)$ accepts $\mathcal{S}$) as defined in Section 6.3.1 from the original CP-IPE scheme. □

Theorem 9. *For any adversary $\mathcal{C}$ against the CP-SE scheme in the same security model for the original CP-IPE scheme, there is an adversary $\mathcal{D}$ against the CP-IPE scheme, running in about the same time as $\mathcal{C}$, such that*

$$\mathsf{Adv}_{\mathcal{C}}^{CP-SE}(\lambda) \leq \mathsf{Adv}_{\mathcal{D}}^{CP-IPE}(\lambda).$$

Moreover, the CP-SE scheme preserves properties from the original CP-IPE scheme.

Proof. Given any adversary $\mathcal{C}$ against the CP-SE scheme in the same security model for the original CP-IPE scheme (fully/selective secure, attribute/payload-hiding), we simulate an adversary $\mathcal{D}$ with advantage $\mathsf{Adv}_{\mathcal{C}}^{CP-SE}(\lambda)$ against the CP-IPE scheme as follows:

- Setup: It runs a real game $\mathcal{RG}$ of CP-IPE and forwards PP to adversary $\mathcal{C}$.

- Query: It answers $\mathcal{C}$'s queries by querying $\mathcal{RG}$'s key generation oracle.

- Challenge: It forwards $\mathcal{A}$'s challenge to $\mathcal{RG}$ and then returns $\mathcal{RG}$'s output to $\mathcal{A}$.

- Guess: It answers $\mathcal{C}$'s queries as Query phase and $\mathcal{D}$ forwards $\mathcal{C}$'s guess to $\mathcal{RG}$.

In the above security game, we can efficiently transform the elements in $\mathcal{K}_{CP-SE}$ and $\mathcal{X}_{CP-SE}$, as required by the CP-IPE setting. From Theorem 8 and its proof, all the oracles of the CP-SE scheme can be simulated correctly. Moreover, the plaintexts $x_{(0)} = (\mathcal{A}_{(0)}, m_{(0)})$, $x_{(1)} = (\mathcal{A}_{(1)}, m_{(1)}) \in \mathcal{X}_{CP-SE}$, any space $\mathcal{S} \in \mathcal{K}_{CP-SE}$ of $\mathcal{C}$'s choices and those in the CP-IPE setting satisfy requirement (6.1) (and other restrictions of the security model simultaneously). Thus, the simulation is perfect and we conclude that $\mathsf{Adv}_{\mathcal{D}}^{CP-IPE}(\lambda)$ is at least $\mathsf{Adv}_{\mathcal{C}}^{CP-SE}(\lambda)$.

It is clear that properties such as full/selective security (under simple assumptions) and attribute or payload-hiding are preserved in the transformation since the model of security game we simulate for the CP-SE scheme is identical to that for the original CP-IPE scheme. Moreover, should the original CP-IPE scheme work in prime order bilinear groups and have short ciphertexts (and other properties), the derived CP-SE scheme would also inherit such properties since the CP-SE scheme can be viewed as a "restricted" form or an embedding of the CP-IPE scheme. □

6.4.4 CP-SE with Key Delegation

We could also construct a CP-SE scheme with key delegation from a CP-IPE scheme that also has key delegation and that supports monotone access structure. This requires only slight modification to the KeyGen algorithm in Section 6.4.3. Other algorithms are unchanged.

Given a space $\mathcal{S} \in \mathcal{K}_{CP-SE}$, we define an attribute set as $\Pi \in \mathcal{K}_{CP-IPE}$. Let $\mathcal{S}$ be an $(n-r)$-dimensional space. We run the $\mathsf{BasisGen}(\mathcal{S})$ algorithm to obtain $\mathcal{B}^{\perp}(\mathcal{S}) = \{\vec{v}_1, \ldots, \vec{v}_r\}$. We then set $\Pi = \{(i, \vec{v}_i), (j, \vec{v}_j^*) : i \in [r], j \in [r+1, n-1]\}$, where $\vec{v}_j^*$ is the wildcard vector $\mathbb{Z}_q^{n_j} \backslash \{\vec{0}\}$ as defined in Section 6.3.2. Given another space $\mathcal{S}' \in \mathcal{K}_{CP-SE}$, where $\mathcal{S}' \preceq \mathcal{S}$, we run the $\mathsf{BasisDel}(\mathcal{S}, \mathcal{S}', \mathcal{B}^{\perp}(\mathcal{S}))$ algorithm to obtain $\mathcal{B}^{\perp}(\mathcal{S}) = \{\vec{v}_1, \ldots, \vec{v}_r, \vec{v}_{r+1}, \ldots, \vec{v}_{r'}\}$. We subsequently set $\Pi' = \{(i, \vec{v}_i), (j, \vec{v}_j^*) : i \in [r'], j \in [r'+1, n-1]\}$. Since $\Pi' \preceq \Pi$, then we can run the Del algorithm of CP-IPE and the resulting key is set to be the secret key for $\mathcal{S}'$.

6.4.5 Construction in Affine Spaces

Given an $(n+1)$-dimensional CP-SE scheme in Euclidean spaces, we could construct an n-dimensional CP-SE scheme in affine spaces by using the similar embedding techniques of [41].

Given an affine space $\mathcal{S}(A, \vec{y}) = \{\vec{z}A + \vec{y} : \vec{z} \in \mathbb{Z}_q^r\}$, where $A \in \mathbb{Z}_q^{r \times n}$, we embed it in $\mathcal{S}((A_1, 0), \ldots, (A_r, 0), (\vec{y}, 1))$. Given an access structure $\mathcal{A}$, we embed each associated vector $\vec{x} \in \mathbb{Z}_q^n$ in $(\vec{x}, 1)$. Then it is not difficult to check that $\vec{x} \in \mathcal{S}(A, \vec{y})$ iff $(\vec{x}, 1) \in \mathcal{S}((A_1, 0), \ldots, (A_r, 0), (\vec{y}, 1))$. Moreover, if $\mathcal{S}(A', \vec{y}')$ is a subspace of $\mathcal{S}(A, \vec{y})$, namely there is some matrix $T \in \mathbb{Z}_q^{r' \times r}$ and vector $\vec{z} \in \mathbb{Z}_q^r$ such that $M' = TA$ and $\vec{y}' = \vec{y} + \vec{z}A$, then $\mathcal{S}((A'_1, 0), \ldots, (A'_{r'}, 0), (\vec{y}', 1))$ is a subspace of $\mathcal{S}((A_1, 0), \ldots, (A_r, 0), (\vec{y}, 1))$, and vice versa.

6.5 Generic Construction of CP-IPE from CP-SE

Interestingly, we observe that one could also construct a CP-IPE scheme from a CP-SE scheme. In other words, CP-IPE and CP-SE are two instances of FE that are equivalent under some reduction. The reduction from CP-SE to CP-IPE is linear, while the reverse direction is quadratically expensive. In this section, we show how to construct a CP-IPE scheme from a CP-SE scheme. Here, we make use of a CP-SE scheme in Euclidean spaces.

6.5.1 Intuition

Our generic construction utilizes similar space partitioning techniques of embedding HIPE in SE [41]. To construct a CP-IPE scheme with structure $\vec{n} := (d; n_1, \ldots, n_d)$, we require an n'-dimensional CP-SE scheme, where $n' = \sum_{i=1}^{i=d}(n_i + 1)$. Given an attribute vector $(i, \vec{v})$ of CP-IPE and $\hat{b} \in \{0, 1\}$, we use $\vec{V}^{(i,\hat{b})} = (0, \ldots, 0, (\vec{v}, \hat{b}), 0, \ldots, 0)$ to denote an n'-dimensional vector of CP-SE, where $(\vec{v}, \hat{b})$ is embedded in the $(\sum_{k=1}^{k=i-1}(n_k + 1) + 1)$-th up to the $(\sum_{k=1}^{k=i}(n_k + 1))$-th scalars. Let $\vec{I}^{(i)} = (0, \ldots, 0, 1, 0, \ldots, 0)$ be an n'-dimensional vector, where the $(\sum_{k=1}^{k=i}(n_k + 1))$-th scalar is 1. Here, we analyze and show how to embed attributes and access structures:

1. A secret key for an attribute set $\Pi := \{(i, \vec{v}_i) | i \in T\}$ is generated using the KeyGen algorithm. We generate a secret key for the orthogonal spaces (denoted as $\mathcal{S}^{\perp}(\Pi)$) of the Euclidean space (denoted as $\mathcal{S}(\Pi)$) spanned by all the vectors as follows: for i from 1 to d, if $i \in T$, then $\vec{V}_i^{(i,0)}$ is included; if $i \notin T$, then $\vec{I}^{(i)}$ is included. In other words, $\mathcal{S}^{\perp}(\Pi)$ contain the following vectors: if $i \in T$, all vectors $\vec{X}^{(i,0)}$ and $\vec{X}^{(i,1)}$ are included, where $\vec{x} \cdot \vec{v}_i = 0$; if $i \notin T$, all vectors $\vec{X}^{(i,0)}$ are

included.

2. To encrypt a message associated with a non-monotone access structure $\mathcal{A} := (M, \rho) \in \mathcal{I}_{CP-IPE}$, we define a new non-monotone access structure $\mathcal{A}' := (M, \rho') \in \mathcal{I}_{CP-SE}$ as follows:

   - if $\rho(j) = (i, \vec{x}_t)$, we embed it in the variable $\vec{X}_t^{(i,1)}$ such that one can obtain the j-th secret share from $\rho(j) = (i, \vec{x}_t)$ iff she can obtain it from $\vec{X}_t^{(i,1)}$. In other words, we have if $i \in T$, then $\vec{x}_t \cdot \vec{v}_i = 0$ iff $\vec{X}_t^{(i,1)} \in \mathcal{S}^{\perp}(\Pi)$; if $i \notin T$, then $\vec{X}_i^{(i,1)} \notin \mathcal{S}^{\perp}(\Pi)$, since $\vec{X}_t^{(i,1)} \cdot \vec{I}^{(i)} = 1$.
   - if $\rho(j) = \neg(i, \vec{x}_t)$, we embed it in the variable $\neg(\vec{X}_t^{(i,0)})$ such that one can obtain the j-th secret share from $\rho(j) = \neg(i, \vec{x}_t)$ iff she can obtain it from $\neg(\vec{X}_t^{(i,0)})$. In other words, we have if $i \in T$, then $\vec{x}_t \cdot \vec{v}_i \neq 0$ iff $\vec{X}_t^{(i,0)} \notin \mathcal{S}^{\perp}(\Pi)$; if $i \notin T$, then $\vec{X}_t^{(i,0)} \in \mathcal{S}^{\perp}(\Pi)$, since $\vec{X}_t^{(i,0)} \cdot \vec{I}^{(i)} = 0$.

We note that the j-th secret share will be never obtained if $i \notin T$ where the additional $(\sum_{k=1}^{k=i}(n_k+1))$-th scalar plays the central role.

6.5.2 Construction

In this section, we follow the same notation of Section 6.5.1. As mentioned before, to construct a CP-IPE scheme with structure $\vec{n} := (d; n_1, \ldots, n_d)$, we require an n'-dimensional CP-SE scheme with non-monotone access structure. Given an attribute set $\Pi := \{(i, \vec{v}_i) | i \in T\} \in \mathcal{K}_{CP-IPE}$ and an access structure $\mathcal{A} := (M, \rho) \in \mathcal{I}_{CP-IPE}$, we define $\mathcal{S}(\Pi) \in \mathcal{K}_{CP-SE}$ and a new access structure $\mathcal{A}' := (M, \rho') \in \mathcal{I}_{CP-SE}$ as in Section 6.5.1.

Given a CP-SE scheme with four algorithms: Setup_{CP-SE}, KeyGen_{CP-SE}, Enc_{CP-SE} and Dec_{CP-SE}, we construct a CP-IPE scheme with the corresponding four algorithms: Setup_{CP-IPE} , KeyGen_{CP-IPE} , Enc_{CP-IPE} and Dec_{CP-IPE}, as follows:

$\mathsf{Setup}_{CP-IPE}(1^k, \vec{n})$ runs $\mathsf{Setup}_{CP-SE}(1^k, n')$ and outputs public parameters PP and a master key MK.

$\mathsf{KeyGen}_{CP-IPE}(\mathsf{MK}, \mathsf{PP}, \Pi)$ runs $\mathsf{BasisGen}(\mathcal{S}(\Pi))$ and outputs $\mathcal{B}^{\perp}(\mathcal{S}(\Pi))$. It then runs $CP-SE$ key generation $\mathsf{KeyGen}_{CP-SE}(\mathsf{PP}, \mathsf{MK}, \mathcal{S}^{\perp}(\Pi))$ and outputs a secret key sk_{Π} with $\mathcal{S}^{\perp}(\Pi)$.

$\mathsf{Enc}_{CP-IPE}(\mathsf{PP}, \mathcal{A}, m)$ runs $\mathsf{Enc}_{CP-SE}(\mathsf{PP}, \mathcal{A}', m)$ and outputs a ciphertext c.

$\mathsf{Dec}_{CP-IPE}(\mathsf{PP}, \mathsf{sk}_\Pi, c)$ runs $\mathsf{Dec}_{CP-SE}(\mathsf{PP}, \mathsf{sk}_\Pi, c)$ and outputs a message m.

We now show that the resulting CP-IPE works correctly and is indeed secure in the following theorems.

Theorem 10. *The CP-IPE scheme transformed from the CP-SE scheme works correctly.*

Proof. Given a plaintext $(\mathcal{A}, m) \in \mathcal{X}_{CP-IPE}$ and an attribute set $\Pi \in \mathcal{K}_{CP-SE}$, we transform them into a plaintext $(\mathcal{A}', m) \in \mathcal{X}_{CP-SE}$ and a space $\mathcal{S}^\perp(\Pi) \in \mathcal{K}_{CP-SE}$ respectively. From our construction, we show that one can obtain the j-th secret share (associated with an attribute within a span program) required by the CP-IPE if and only if she can obtain the j-th shares from the CP-SE. In other words (assume $[\rho(j) = (i, \vec{x}_t)]$ or $[\rho(j) = \neg(i, \vec{x}_t)]$):
Case 1: $i \in T$,

- $[\rho(j) = (i, \vec{x}_t)] \wedge [(i, \vec{v}_i) \in \Pi] \wedge [\vec{x}_t \cdot \vec{v}_i = 0]$ iff $[\rho'(j) = \vec{X}_t^{(i,1)}] \wedge [\vec{X}_t^{(i,1)} \in \mathcal{S}^\perp(\Pi)]$,
- $[\rho(j) = \neg(i, \vec{x}_t)] \wedge [(i, \vec{v}_i) \in \Pi] \wedge [\vec{x}_t \cdot \vec{v}_i \neq 0]$ iff $[\rho'(j) = \neg(\vec{X}_t^{(i,0)})] \wedge [\vec{X}_t^{(i,0)} \notin \mathcal{S}^\perp(\Pi)]$,

the above items are equivalent to:

- $\vec{x}_t \cdot \vec{v}_i = 0$ iff $\vec{X}_t^{(i,1)} \in \mathcal{S}^\perp(\Pi)$,
- $\vec{x}_t \cdot \vec{v}_i \neq 0$ iff $\vec{X}_t^{(i,0)} \notin \mathcal{S}^\perp(\Pi)$,

which can be obtained from the definition of $\mathcal{S}^\perp(\Pi)$ and the below property

- $\vec{x}_t \cdot \vec{v}_i = 0 \Leftrightarrow \vec{X}_t^{(i,1)} \cdot \vec{V}_i^{(i,0)} = 0 \Leftrightarrow \vec{X}_t^{(i,1)} \in \mathcal{S}^\perp(\Pi)$,
- $\vec{x}_t \cdot \vec{v}_i \neq 0 \Leftrightarrow \vec{X}_t^{(i,0)} \cdot \vec{V}_i^{(i,0)} \neq 0 \Leftrightarrow \vec{X}_t^{(i,0)} \notin \mathcal{S}^\perp(\Pi)$.

Case 2: $i \notin T$,

- if $\rho(j) = (i, \vec{x}_t)$, then $\vec{X}_t^{(i,1)} \cdot \vec{I}^{(\mu_i+i)} = 1$, that is $\vec{X}_t^{(i,1)} \notin \mathcal{S}^\perp(\Pi)$,
- if $\rho(j) = \neg(i, \vec{x}_t)$, then $\vec{X}_t^{(i,0)} \cdot \vec{I}^{(\mu_i+i)} = 0$, that is $\vec{X}_t^{(i,0)} \in \mathcal{S}^\perp(\Pi)$,

which means one can never obtain the j-th secret share.

This implies that the resulting CP-IPE scheme inherits the decryptability from the original CP-SE scheme, i.e., $\mathcal{F}(\Pi, (\mathcal{A}, m)) := m$ iff $\mathcal{A} := (M, \rho)$ accepts Π as defined in Section 6.3.2. □

Theorem 11. *For any adversary $\mathcal{C}$ against the CP-IPE scheme in the same security model for the original CP-SE scheme, there is an adversary $\mathcal{D}$ against the CP-SE scheme, running in about the same time as $\mathcal{C}$, such that*

$$\mathsf{Adv}_{\mathcal{C}}^{CP-IPE}(\lambda) \leq \mathsf{Adv}_{\mathcal{D}}^{CP-SE}(\lambda).$$

Moreover, the CP-IPE scheme preserves properties from the original CP-SE scheme.

This theorem could be proved in a very similar way as the proof of Theorem 9.

6.5.3 Construction of CP-IPE from CP-SE with Key Delegation

Here, we show how to construct a CP-IPE scheme with key delegation from a delegatable CP-SE scheme. Since delegatable CP-SE can support only monotone access structure, the resulting CP-IPE scheme inherits the same property.

To construct a CP-IPE scheme with hierarchy $\vec{n} := (d; n_1, \ldots, n_d)$, we require an n-dimensional CP-SE scheme, where $n = \sum_{i=1}^{d} n_i$. Here, let $\mu_i = \sum_{k=1}^{k=i} n_k$. Given an attribute vector $(i, \vec{v})$, we use $\vec{V}^{(i)} = (0, \ldots, 0, \vec{v}, 0, \ldots, 0)$ to denote an n-dimensional vector, where $\vec{v}$ is embedded in the $(\mu_{i-1}+1)$-th scalar up to the μ_i-th scalar. Let $\vec{I}_k = (0, \ldots, 0, 1, 0, \ldots, 0)$ be an n-dimensional vector, where the k-th scalar is 1. This requires slight modification to the KeyGen, Enc algorithms in Section 6.5.2 and other algorithms are unchanged.

1. A secret key for an attribute set $\Pi := \{(i, \vec{v}_i) | i \in T\}$ is generated using the KeyGen algorithm of the CP-SE scheme. We generate a secret key for the orthogonal space $\mathcal{S}^{\perp}(\Pi)$ of the Euclidean space spanned by all the vectors as follows: for i from 1 to d, if $i \in T$ and $\vec{v}_i \neq \vec{v}_i^*$, then $\vec{V}_i^{(i)}$ is included; if $i \notin T$, then $\{\vec{I}_{\mu_{i-1}+1}, \ldots, \vec{I}_{\mu_i}\}$ are included. In other words, the orthogonal spaces contain the following vectors: if $i \in T$ and $\vec{v}_i \neq \vec{v}_i^*$, all vectors $\vec{X}^{(i)}$ are included, where $\vec{x} \cdot \vec{v}_i = 0$; if $i \in T$ and $\vec{v}_i = \vec{v}_i^*$, all vectors $\vec{X}^{(i)}$ are included; if $i \notin T$, no vector $\vec{X}^{(i)}$ is included.

2. To encrypt a message associated with a monotone access structure $\mathbb{S} := (M, \rho) \in \mathcal{I}_{CP-IPE}$, we define a new monotone access structure $\mathbb{S}' := (M, \rho') \in \mathcal{I}_{CP-SE}$. We embed each attribute vector $\rho(j) = (i, \vec{x}_t)$ into the vector $\vec{X}_t^{(i)}$.

3. To delegate a secret key to an attribute set $\Pi' := \{(i, \vec{v}_i) | i \in T'\}$ and $\Pi' \preceq \Pi$. It is not difficult to see that $\mathcal{S}(\Pi)$ is a subspace of $\mathcal{S}(\Pi')$, then $\mathcal{S}^{\perp}(\Pi')$ is a subspace of $\mathcal{S}^{\perp}(\Pi)$. We run the Del algorithm of the CP-SE scheme.

6.6 Application

Here, we describe some applications of CP-SE.

SE Supporting Negation. It is clear that SE supporting negation [10] is a very special case of CP-SE. In an SE scheme supporting negation, a secret key is associated with an affine space $\mathcal{S}$ and an access structure of a ciphertext is a span program associated with only one vector $\vec{x}$, where $\mathcal{F}(\mathcal{S}, (\mathcal{A}, m)) = m$ iff $\vec{x} \notin \mathcal{S}$.

Subspace Inclusion Encryption. We generalize the notion of SE such that ciphertexts are associated with affine spaces instead of vectors, i.e., given a space $\mathcal{S} \in \mathcal{K}$ and a space $\mathcal{S}' \in \mathcal{I}$, $\mathcal{F}(\mathcal{S}, (\mathcal{S}', m)) = m$ iff $\mathcal{S}'$ is a subspace of $\mathcal{S}$. We call it subspace inclusion encryption (SIE). We utilize the same embedding technique in Section 6.4.5, that is we embed an affine space $\mathcal{S}(A, \vec{y})$ in the Euclidean space $\mathcal{S}((A_1, 0), \ldots, (A_r, 0), (\vec{y}, 1))$. This is essential for embedding SIE in CP-SE, since an access structure of CP-SE is a span program associated with a set of attribute vectors while each Euclidean space has a basis. Hence, we now require only a CP-SE scheme in Euclidean spaces and can embed SIE as follows:

- A secret key for an affine space $\mathcal{S}(A, \vec{y}) \in \mathcal{K}_{SIE}$ is generated using the KeyGen algorithm of the CP-SE scheme by taking as input the Euclidean space $\mathcal{S}((A_1, 0), \ldots, (A_r, 0), (\vec{y}, 1))$.

- A message associated with an affine space $\mathcal{S}(A', \vec{y}') \in \mathcal{I}_{SIE}$ is encrypted under the monotone access structure in the form of $(r' + 1)$-out-of-$(r' + 1)$ secret sharing for all the attribute vectors $\{(A'_1, 0), \ldots, (A'_{r'}, 0), (\vec{y}', 1)\}$, where $A' \in \mathbb{Z}_q^{r' \times n}$.

Hierarchical ABE We generalize ABE such that each attribute possesses the delegation capability as

with HIBE, achieving hierarchical ABE (HABE) [115].

We first show how one can embed CP-ABE in CP-SE. Let us assume that the attribute universe is $\{x_1, \ldots, x_d\}$ and there exists a d-dimensional CP-SE scheme. Let $I_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ be a d-dimensional vector where the i-th scalar is 1. We then do the following:

- A secret key for an attribute set $\{x_i | i \in T\} \in \mathcal{K}_{CP-ABE}$ where $T \subseteq [d]$ is generated using the KeyGen algorithm of the CP-SE scheme by taking as input $\mathcal{S}(\{I_i | i \in T\})$.
- To encrypt a message associated with a non-monotone access structure $\mathcal{A} := (M, \rho) \in \mathcal{I}_{CP-ABE}$, we define a new non-monotone access structure $\mathcal{A}' := (M, \rho') \in \mathcal{I}_{CP-SE}$ such that:
  - if $\rho(j) = x_t$ then $\rho'(j) = I_t$;
  - if $\rho(j) = \neg x_t$ then $\rho'(j) = \neg I_t$.

To extend CP-ABE to the hierarchical setting, we need to make use of a CP-SE scheme that has a delegation mechanism and thus the derived HABE scheme may not supporting non-monotone access structure. We utilize multiple dimensions to deal with the corresponding hierarchical attribute instead of one dimension for each attribute in the above construction of CP-ABE scheme.

6.7 Discussion

Let CP-IPE$_{\mathrm{OT}}$ and d-CP-IPE$_{\mathrm{OT}}$ denote the CP-IPE scheme of [96] and the same scheme with key delegation, respectively. Let CP-SE$^*_{\mathrm{OT}}$ and d-CP-SE$^*_{\mathrm{OT}}$ denote the CP-SE schemes derived from CP-IPE$_{\mathrm{OT}}$ and d-CP-IPE$_{\mathrm{OT}}$, respectively. We let also n_μ denote the maximal value of $\{n_i : i \in [d]\}$ and ℓ denote the maximal number of varieties in an access structure. We now list the properties of the CP-IPE schemes and the resulting corresponding CP-SE schemes in Table 6.2. The properties include the size of public parameters PP, the size of secret keys sk, the size of ciphertexts c, the number # of pairing computation for decryption, and whether the scheme is fully secure, anonymous, has short ciphertexts c, in prime order bilinear groups and under a simple assumption. Here, the sizes are in the number of group elements. Moreover, the CP-IPE scheme are with structure $\vec{n} := (d; n_1, \ldots, n_d)$ and the CP-SE scheme are n-dimensional.

Table 6.2: Comparisons between CP-IPE and corresponding derived CP-SE schemes.

	CP-IPE$_{\mathrm{OT}}$	CP-SE$^*_{\mathrm{OT}}$	d-CP-IPE$_{\mathrm{OT}}$	d-CP-SE$^*_{\mathrm{OT}}$
size of PP	$O((n_\mu+\ell)^2 d)$	$O((n+\ell)^3)$	$O((n_\mu+\ell)^2 d)$	$O((n+\ell)^3)$
size of sk	$O((n_\mu+\ell)d)$	$O((n+\ell)^2)$	$O((n_\mu+\ell)^2 d)$	$O((n+\ell)^3)$
size of c	$O((n_\mu+\ell)\ell)$	$O((n+\ell)\ell)$	$O((n_\mu+\ell)\ell)$	$O((n+\ell)\ell)$
# pairings	$O((n_\mu+\ell)\ell)$	$O((n+\ell)\ell)$	$O((n_\mu+\ell)\ell)$	$O((n+\ell)\ell)$
fully secure	Yes	Yes	Yes	Yes
anonymous	No	No	No	No
short c	No	No	No	No
prime order	Yes	Yes	Yes	Yes
simple assumptions	Yes	Yes	Yes	Yes

We note, from Table 6.2, that the size of ciphertexts of both the CP-IPE schemes and the resulting CP-SE schemes are dependent on the number of attributes ℓ and the related dimensions n or n_μ. We leave how to construct CP-IPE and CP-SE schemes with shorter ciphertexts as open problems.

Chapter 7

Conclusions

Contents

This chapter summaries the thesis and gives some concluding remarks on the problems studied and the results achieved. We also give suggestions for future work in this area.

7.1 Concluding Remarks

The development of pairing-based cryptography and its application in various cryptographic primitives such as identity-based encryption (IBE), broadcast encryption (BE), attribute-based encryption (ABE) and spatial encryption (SE) etc are amongst today's most important technical innovations in the field of cryptology. Over the past decade, the development in this area has attracted not only the attention from academic circle, but also industrial organizations who transfer theoretical research into real applications.

In this thesis, we study four aspects of the pairing-based cryptography: key update in IBE, ciphertext update in BE, ciphertext compactness in BE and ABE, and last but not least, a generic construction of SE from inner-product encryption (IPE).

In Chapter 3, we investigate the concept of revocable IBE (RIBE). It is important to minimize the regular key update information in an IBE scheme when users have been revoked. The classical proposal of a selectively secure RIBE scheme by Boldyreva, Goyal, and Kumara, and a subsequent adaptively

secure construction by Libert and Vergnaud are all based on a technique called binary tree approach, such that their key update size is logarithmic in the number of users. We study the problem of key update size reduction and show that, indeed, the key update material can be made constant with only some small amount of auxiliary information, through a novel combination of the Lewko and Waters IBE scheme and the Camenisch, Kohlweiss, and Soriente pairing-based dynamic accumulator. In this chapter, we propose a concrete RIBE construction achieving this goal, together with a rigorous security analysis. We further extend our technique to a RIBE scheme supports forward secrecy and also a revocable ABE scheme. One immediate open problem would be to achieve adaptive security under more standard assumptions such as the Decisional Linear assumption, and to remove the auxiliary information to make the key update size truly constant. Also, it would be interesting to investigate the possibility of applying our accumulator-based key update technique to revocable storage ABE proposed by Sahai et al. [103] and other variants of functional encryption.

In the subsequent Chapter 4, we continue our study of user revocation from a different angle: update a ciphertext to prevent revoked users from accessing future content changes. We extend our sight to the encrypted file systems (EFS) where files are shared among recipients and revocation of users are frequent. In this chapter, we first define a list of desirable properties of an ideal EFS system. With subsequent analysis, we show potential development space for existing approaches which the key updates for legitimate users are at linear cost. We then propose a new primitive called updatable broadcast encryption (UBE), which could be used to achieve better efficiency for EFS systems. Through some novel techniques, we provide two concrete UBE constructions based on different broadcast encryption schemes which theoretically outperform existing approaches: they are near-ideal in the sense that the revocation cost is constant at the file owner, the server and the recipients sides, and the file header size is also constant size in an amortized manner (where the "near-ideal" comes from). One possible next step is to deploy our schemes into real-life applications, such as the cloud-based file sharing service Dropbox, and evaluate and compare its performance with existing solutions.

As a continuation of the study of Chapter 4, in the next chapter, we explore the possibility of saving the ciphertext overhead when a user encrypts, stores and shares her files at a cloud-based server. Two primitives, broadcast encryption and attribute-based encryption, are widely used for enforcing crypto-

graphic access control at different levels of granularity. However, most current BE and ABE approaches are largely designed for encryption of a single message. This, however, contradicts with real practice when it is often desirable to encrypt multiple messages simultaneously and to share them with potentially different sets of recipients. A straightforward approach is to apply any BE or ABE scheme to each individual message in parallel. However, the resulting ciphertexts overhead are likely to be large. This is due to the fact that each individual message is typically encrypted using a fresh, unique random value to have an appropriate level of security guarantee. In this chapter, we investigate the possibility of reusing random values across multiple messages targeting for different recipient sets during encryption. We propose two new primitives called multi-message broadcast encryption (MM-BE) and multi-message key-policy attribute-based encryption (MM-KP-ABE), and provide two concrete constructions. Our MM-BE scheme reduces the ciphertext size of the existing straightforward approach by almost half; while our MM-KP-ABE scheme cuts down the ciphertext size from quadratic to linear complexity. Two immediate open problems may be worth further investigation. The first is to further reduce the user private key size of our schemes while maintaining the current ciphertext size savings. The second is to achieve constant size ciphertexts, such as that of the multi-channel BE scheme proposed in [99] but under more general setting. That is, the encryptor can be any user in the system, and different messages can be encrypted under different (and potentially overlapping) access policies.

Last but not least, in Chapter 6, we turn our focus to a pure theoretical study of the relation between two cryptographic primitives. We investigate a variant of spatial encryption (SE) called ciphertext-policy SE (CP-SE). This primitive combines the properties of SE and those from ciphertext-policy attribute-based encryption (CP-ABE), and supports non-monotone access structure. In CP-SE, the decryptability of a ciphertext depends on whether the required attribute vectors are in the same affine space that also corresponds to the decryption key. This property rises many new applications, for example, SE supporting negation, hierarchical ABE (HABE) and forward-secure ABE. In this chapter, we present techniques for generic construction of CP-SE from ciphertext-policy inner product encryption (CP-IPE). Our techniques are property-preserving in the sense that if the CP-IPE scheme from which we derive our CP-SE scheme is fully secure, then so is the resulting CP-SE scheme. Moreover, interestingly, we show that it is possible for a reverse direction: a generic construction of a CP-IPE scheme from a CP-SE scheme.

We note that the size of ciphertexts of both the CP-IPE schemes and the resulting CP-SE schemes are dependent on the number of attributes and the related dimensions. We leave the construction of CP-IPE and CP-SE schemes with shorter ciphertexts as open problems.

Over the past decade, the proliferation of pairing-based cryptography has not only brought numerous theoretical advances in the world of cryptography, but also have attracted immense attention from industry. To name a few successful stories: pairing-based primitives have been implemented in sensor nodes, identity-based authentication shows its advantage in network session initiation, a company named Voltage Security Inc., which recently acquired by Hewlett-Packard, started from a commercial email protection solution based on identity-based encryption, and not to mention the powerfulness of attribute-based encryption in various access-control applications such as electronic health record access. Without any doubt, pairing-based primitives, especially those under functional encryption, will continue to play a more significant role in every aspect of the IT industry in the coming future. We admit those primitives discussed in this thesis, together with other pairing-based concepts, are still in the process of further development and insufficiencies still exist, for example the computational cost of pairing operation is orders of magnitude slower than symmetric key encryption operations. Still we believe, combined with traditional symmetric key techniques, such a hybrid approach will bring immense opportunities and energy for both scientific researchers and industrial practitioners.

Chapter 8

Bibliography

[1] Everything you need to work smarter. https://www.dropbox.com/business/pricing.

[2] William Aiello, Sachin Lodha, and Rafail Ostrovsky. Fast digital identity revocation. In *CRYPTO*, pages 137–152. Springer, 1998.

[3] Selim G Akl and Peter D Taylor. Cryptographic solution to a problem of access control in a hierarchy. *ACM Transactions on Computer Systems (TOCS)*, 1(3):239–248, 1983.

[4] Corey M Angst and Ritu Agarwal. Adoption of electronic health records in the presence of privacy concerns: The elaboration likelihood model and individual persuasion. *MIS quarterly*, 33(2):339–370, 2009.

[5] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. Provable data possession at untrusted stores. In *CCS*, pages 598–609. ACM, 2007.

[6] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security*, 9(1):1–30, 2006.

[7] Nuttapong Attrapadung, Javier Herranz, Fabien Laguillaumie, Benoît Libert, Elie De Panafieu, and Carla Ràfols. Attribute-based encryption schemes with constant-size ciphertexts. *Theoretical Computer Science*, 422:15–38, 2012.

[8] Nuttapong Attrapadung and Hideki Imai. Attribute-based encryption supporting direct/indirect revocation modes. In *IMA International Conference on Cryptography and Coding*, pages 278–300. Springer, 2009.

[9] Nuttapong Attrapadung and Hideki Imai. Conjunctive broadcast and attribute-based encryption. In *Pairing*, pages 248–265. Springer, 2009.

[10] Nuttapong Attrapadung and Benoît Libert. Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation. In *PKC*, pages 384–402. Springer, 2010.

[11] Nuttapong Attrapadung, Benoît Libert, and Elie De Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *PKC*, pages 90–108. Springer, 2011.

[12] Michael Backes, Dario Fiore, and Raphael M Reischuk. Verifiable delegation of computation on outsourced data. In *CCS*, pages 863–874. ACM, 2013.

[13] Joonsang Baek, Cheng-Kang Chu, and Jianying Zhou. On shortening ciphertexts: New constructions for compact public key and stateful encryption schemes. In *CT-RSA*, page 302. Springer, 2011.

[14] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Efficient multi-receiver identity-based encryption and its application to broadcast encryption. In *PKC*, pages 380–397. Springer, 2005.

[15] Joonsang Baek and Yuliang Zheng. Identity-based threshold decryption. In *PKC*, page 262. Springer, 2004.

[16] Joonsang Baek, Jianying Zhou, and Feng Bao. Generic constructions of stateful public key encryption and their applications. In *ACNS*, pages 75–93. Springer, 2008.

[17] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid. Recommendation for key management – part 1: General (revision 3). In *NIST Special Publication 800-57*, 2012.

[18] Amos Beimel. *Secure schemes for secret sharing and key distribution*. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.

[19] Mihir Bellare, Alexandra Boldyreva, Kaoru Kurosawa, and Jessica Staddon. Multi recipient encryption schemes: How to save on bandwidth and computation without sacrificing security. *IEEE Transactions on Information Theory*, 53(11):3927–3943, 2007.

[20] Mihir Bellare, Tadayoshi Kohno, and Victor Shoup. Stateful public-key cryptosystems: how to encrypt with one 160-bit exponentiation. In *CCS*, pages 380–389. ACM, 2006.

[21] Josh Benaloh and Michael de Mare. One-way accumulators: a decentralized alternative to digital signatures. In *EUROCRYPT*, pages 274–285. Springer, 1993.

[22] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *S&P*, pages 321–334. IEEE, 2007.

[23] Matt Blaze. A cryptographic file system for unix. In *CCS*, pages 9–16. ACM, 1993.

[24] Alexandra Boldyreva, Vipul Goyal, and Virendra Kumar. Identity-based encryption with efficient revocation. In *CCS*, pages 417–426. ACM, 2008.

[25] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT*, page 440. Springer, 2005.

[26] Dan Boneh, Xuhua Ding, Gene Tsudik, and Chi Ming Wong. A method for fast revocation of public key certificates and security capabilities. In *USENIX*, page 22. USENIX Association, 2001.

[27] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, page 213. Springer, 2001.

[28] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO*, pages 258–275. Springer, 2005.

[29] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *TCC*, pages 325–341. Springer, 2005.

[30] Dan Boneh and Michael Hamburg. Generalized identity based and broadcast encryption schemes. In *CRYPTO*, pages 455–470. Springer, 2008.

[31] Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In *CRYPTO 2013*, pages 410–428. Springer, 2013.

[32] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: definitions and challenges. In *TCC*, pages 253–273. Springer, 2011.

[33] Boxcryptor. `https://www.boxcryptor.com/`.

[34] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *PKC*, pages 481–500. Springer, 2009.

[35] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, pages 255–271. Springer, 2003.

[36] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *Eurocrypt*, pages 207–222. Springer, 2004.

[37] Giuseppe Cattaneo, Luigi Catuogno, Aniello Del Sorbo, and Pino Persiano. The design and implementation of a transparent cryptographic file system for UNIX. In *USENIX*, pages 199–212. USENIX Association, 2001.

[38] Jae Choon Cha and Jung Hee Cheon. An identity-based signature from gap diffie-hellman groups. In *PKC*, pages 18–30. Springer, 2003.

[39] Chin-Chen Chang, Ren-Junn Hwang, and Tzong-Chen Wu. Cryptographic key assignment scheme for access control in a hierarchy. *Information Systems*, 17(3):243–247, 1992.

[40] Jie Chen, Hoon Wei Lim, San Ling, Le Su, and Huaxiong Wang. Spatial encryption supporting non-monotone access structure. *Designs, Codes and Cryptography*, pages 1–16, 2013.

[41] Jie Chen, Hoon Wei Lim, San Ling, and Huaxiong Wang. The relation and transformation between hierarchical inner product encryption and spatial encryption. *Designs, codes and cryptography*, 71(2):347–364, 2014.

[42] Jie Chen, Hoon Wei Lim, San Ling, Huaxiong Wang, and Khoa Nguyen. Revocable identity-based encryption from lattices. In *ACISP*, pages 390–403. Springer, 2012.

[43] Richard Chow, Philippe Golle, Markus Jakobsson, Elaine Shi, Jessica Staddon, Ryusuke Masuoka, and Jesus Molina. Controlling data in the cloud: outsourcing computation without outsourcing control. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 85–90. ACM, 2009.

[44] Sherman Sze Ming. Chow. Removing escrow from identity-based encryption. In *PKC*, pages 256–276, 2009.

[45] Cheng-Kang Chu, Sherman SM Chow, Wen-Guey Tzeng, Jianying Zhou, and Robert H Deng. Key-aggregate cryptosystem for scalable data sharing in cloud storage. *Parallel and Distributed Systems, IEEE Transactions on*, pages 468–477, 2014.

[46] Cheng-Kang Chu, Jian Weng, Sherman SM Chow, Jianying Zhou, and Robert H Deng. Conditional proxy broadcast re-encryption. In *ACISP*, pages 327–342. Springer, 2009.

[47] Cloudfogger. `https://www.cloudfogger.com/`.

[48] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pages 360–363. Springer, 2001.

[49] OpenID Connect. `http://openid.net/connect/`.

[50] DataLocker. `http://datalocker.com/`.

[51] Cécile Delerablée. Identity-based broadcast encryption with constant size ciphertexts and private keys. In *ASIACRYPT*, pages 200–215. Springer, 2007.

[52] Cécile Delerablée, Pascal Paillier, and David Pointcheval. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In *Pairing*, pages 39–59. Springer, 2007.

[53] Xuhua Ding and Gene Tsudik. Simple identity-based cryptography with mediated RSA. In *CT-RSA*, pages 193–210. Springer, 2003.

[54] Carl Ellison and Bruce Schneier. Ten risks of PKI: What you're not being told about public key infrastructure. *Computer Security Journal*, 16(1):1–7, 2000.

[55] IKEA enables large-scale file sharing with Egnyte. `http://www.egnyte.com/blog/2012/11/`.

[56] Amos Fiat and Moni Naor. Broadcast encryption. In *CRYPTO*, pages 480–491. Springer, 1993.

[57] Matt Franklin. An introduction to identity-based encryption. In *NIST Identity-based Encryption Workshop*, 2008.

[58] John F Gantz and David Reinsel. The expanding digital universe: A forecast of worldwide information growth through 2010. IDC, 2007.

[59] Paolo Gasti and Alessio Merlo. On re-use of randomness in broadcast encryption. In *PST*, pages 36–43. IEEE, 2011.

[60] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In *ASIACRYPT*, pages 548–566. Springer, 2002.

[61] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In *EUROCRYPT*, pages 171–188. Springer, 2009.

[62] Eu-Jin Goh, Hovav Shacham, Nagendra Modadugu, and Dan Boneh. Sirius: Securing remote untrusted storage. In *NDSS*, pages 131–145, 2003.

[63] Juan Manuel González-Nieto, Mark Manulis, and Dongdong Sun. Fully private revocable predicate encryption. In *ACISP*, pages 350–363. Springer, 2012.

[64] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS*, pages 89–98. ACM, 2006.

[65] Matthew Green, Susan Hohenberger, and Brent Waters. Outsourcing the decryption of abe ciphertexts. In *USENIX*, pages 34–34. USENIX Association, 2011.

[66] Tracy D Gunter and Nicolas P Terry. The emergence of national electronic health record architectures in the united states and australia: models, costs, and questions. *Journal of Medical Internet Research*, 7(1), 2005.

[67] Shai Halevi, Danny Harnik, Benny Pinkas, and Alexandra Shulman-Peleg. Proofs of ownership in remote storage systems. In *CCS*, pages 491–500. ACM, 2011.

[68] Mike Hamburg. *Spatial Encryption*. PhD thesis, Stanford University, 2011.

[69] Yumiko Hanaoka, Goichiro Hanaoka, Junji Shikata, and Hideki Imai. Identity-based hierarchical strongly key-insulated encryption and its application. In *ASIACRYPT*, pages 495–514. Springer, 2005.

[70] Florian Hess. Efficient identity based signature schemes based on pairings. In *SAC*, pages 310–324. Springer, 2003.

[71] HighQ. `http://highq.com/industries/`.

[72] Hightail. `https://www.hightail.com/solutions/`.

[73] M Jason Hinek, Mo King Low, and Edlyn Teske. On some attacks on multi-prime RSA. In *SAC*, pages 385–404. Springer, 2003.

[74] Susan Hohenberger and Brent Waters. Attribute-based encryption with fast decryption. In *PKC*, pages 162–179. Springer, 2013.

[75] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In *EUROCRYPT*, pages 466–481. Springer, 2002.

[76] Russell Housley, W Polk, Warwick Ford, and David Solo. Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile, 2002.

[77] Anca-Andreea Ivan and Yevgeniy Dodis. Proxy cryptography revisited. In *NDSS*, 2003.

[78] Antoine Joux. A one round protocol for tripartite diffie–hellman. In *Algorithmic number theory*, pages 385–393. Springer, 2000.

[79] Ari Juels and Burton S Kaliski Jr. PORs: Proofs of retrievability for large files. In *CCS*, pages 584–597. ACM, 2007.

[80] Mahesh Kallahalla, Erik Riedel, Ram Swaminathan, Qian Wang, and Kevin Fu. Plutus: Scalable secure file sharing on untrusted storage. In *USENIX*, pages 29–42. USENIX Association, 2003.

[81] Seny Kamara and Kristin Lauter. Cryptographic cloud storage. In *Financial Cryptography and Data Security*, pages 136–149. Springer, 2010.

[82] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162. Springer, 2008.

[83] Kwangsu Lee, Seung Geol Choi, Dong Hoon Lee, Jong Hwan Park, and Moti Yung. Self-updatable encryption: Time constrained access control with hidden attributes and better efficiency. In *ASIACRYPT*, pages 235–254. Springer, 2013.

[84] Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91. Springer, 2010.

[85] Allison Lewko, Amit Sahai, and Brent Waters. Revocation systems with very small private keys. In *S&P*, pages 273–285, 2010.

[86] Allison Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *TCC*, page 455. Springer, 2010.

[87] Benoît Libert, Kenneth G Paterson, and Elizabeth A Quaglia. Anonymous broadcast encryption: adaptive security and efficient constructions in the standard model. In *PKC*, pages 206–224. Springer, 2012.

[88] Benoît Libert and Jean-Jacques Quisquater. Efficient revocation and threshold pairing based cryptosystems. In *Proceedings of the 22nd Annual Symposium on Principles of Distributed Computing*, pages 163–171. ACM, 2003.

[89] Benoît Libert and Damien Vergnaud. Adaptive-ID secure revocable identity-based encryption. In *CT-RSA*, pages 1–15. Springer, 2009.

[90] Stephen J. MacKinnon, Peter D. Taylor, Henk Meijer, and Selim G. Akl. An optimal algorithm for assigning cryptographic keys to control access in a hierarchy. *Computers, IEEE Transactions on*, 100(9):797–802, 1985.

[91] Ethan L Miller, William E Freeman, Darrell DE Long, and Benjamin C Reed. Strong security for network-attached storage. In *USENIX FAST*, pages 1–13, 2002.

[92] Randy Muller. How it works: Encrypting file systems, May 2006.

[93] Michael Myers, Rich Ankney, Ambarish Malpani, Slava Galperin, and Carlisle Adams. X.509 internet public key infrastructure online certificate status protocol-OCSP. Technical report, 1999.

[94] Moni Naor and Kobbi Nissim. Certificate revocation and certificate update. In *USENIX*, page 17. USENIX Association, 1998.

[95] Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In *ASIACRYPT*, page 214. Springer, 2009.

[96] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, page 191. Springer, 2010.

[97] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *CCS*, pages 195–203. ACM, 2007.

[98] Duong-Hieu Phan, David Pointcheval, Siamak F Shahandashti, and Mario Strefler. Adaptive CCA broadcast encryption with constant-size secret keys and ciphertexts. In *ACISP*, pages 308–321. Springer, 2012.

[99] Duong Hieu Phan, David Pointcheval, and Viet Cuong Trinh. Multi-channel broadcast encryption. In *ASIACCS*, pages 277–286. ACM, 2013.

[100] Matthew Pirretti, Patrick Traynor, Patrick McDaniel, and Brent Waters. Secure attribute-based systems. *Journal of Computer Security*, 18(5):799–837, 2010.

[101] Geraint Price. Public key infrastructures: A research agenda. *Journal of Computer Security*, 14(5):391–417, 2006.

[102] Scientific Data Sharing Project. `http://scientificdatasharing.com/`.

[103] Amit Sahai, Hakan Seyalioglu, and Brent Waters. Dynamic credentials and ciphertext delegation for attribute-based encryption. In *CRYPTO*, pages 199–217. Springer, 2012.

[104] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473. Springer, 2005.

[105] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing. In *The 2000 Symposium on Cryptography and Information Security*, pages 135–148, 2000.

[106] Ravinderpal S Sandhu. Cryptographic implementation of a tree hierarchy for access control. *Information Processing Letters*, 27(2):95–98, 1988.

[107] Jae Hong Seo and Keita Emura. Revocable identity-based encryption revisited: Security model and construction. In *PKC*, pages 216–234. Springer, 2013.

[108] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, page 47. Springer, 1984.

[109] SkyCrypt. `http://www.skycrypt.com/`.

[110] Le Su, Hoon Wei Lim, San Ling, and Huaxiong Wang. Revocable IBE systems with almost constant-size key update. In *Pairing*, pages 168–185. Springer, 2014.

[111] Windows Encrypting File System. `http://support.microsoft.com/kb/223316`.

[112] Hassan Takabi, James BD Joshi, and Gail-Joon Ahn. Security and privacy challenges in cloud computing environments. In *S&P*, pages 24–31. IEEE, 2010.

[113] Wen-Guey Tzeng. A time-bound cryptographic key assignment scheme for access control in a hierarchy. *Knowledge and Data Engineering, IEEE Transactions on*, 14(1):182–188, 2002.

[114] Viivo. `http://www.viivo.com/`.

[115] Guojun Wang, Qin Liu, and Jie Wu. Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In *CCS*, pages 735–737. ACM, 2010.

[116] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *CRYPTO*, pages 619–636. Springer, 2009.

[117] Brent Waters. Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In *PKC*, pages 53–70. Springer, 2011.

[118] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *INFOCOM*, pages 1–9. IEEE, 2010.

[119] E. Zadok, I. Badulescu, and A. Shender. Cryptfs: A stackable vnode level encryption file system. Technical Report CUCS-021-98, Columbia University, 1998.