THEORY AND APPLICATIONS OF OUTSIDER ANONYMITY IN BROADCAST ENCRYPTION

by

IRIPPUGE DESHAN MILINDA PERERA

A dissertation submitted to the Graduate Faculty in Computer Science
in partial fulfillment of the requirements for the degree of Doctor of Philosophy,
The City University of New York

2015

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirements for the degree of Doctor of Philosophy.

Dr. Nelly Fazio

April 3, 2015 — Date

[illegible] — Chair of Examining Committee

Dr. Robert Haralick

April 3, 2015 — Date

[illegible] — Executive Officer

Dr. Rosario Gennaro

Dr. Antonio Nicolosi

Dr. William E. Skeith III

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

Theory and Applications of Outsider Anonymity in Broadcast Encryption

by

Irippuge Deshan Milinda Perera

Adviser: Dr. Nelly Fazio

Broadcast encryption (BE) allows efficient one-to-many secret communication of data over a broadcast channel. In the standard setting of BE, information about receivers is transmitted in the clear together with ciphertexts. This could be a serious violation of recipient privacy since the identities of the users authorized to access the secret content in certain broadcast scenarios are as sensitive as the content itself. *Anonymous broadcast encryption* (AnoBE) prevents this leakage of recipient identities from ciphertexts but at a cost of a *linear* lower bound (in the number of receivers) on the length of ciphertexts. A linear ciphertext length is a highly undesirable bottleneck in any large-scale broadcast application. In this thesis, we propose a less stringent yet very meaningful notion of anonymity for broadcast encryption called *outsider-anonymous broadcast encryption* (oABE) that allows the creation of ciphertexts that are sublinear in the number of receivers. We construct several oABE schemes with varying security guarantees and levels of efficiency. We also present two very interesting cryptographic applications afforded by the efficiency of our oABE schemes. The first is *broadcast steganography* (BS), the extension of the state of the art setting of point-to-point steganography to the multi-recipient setting. The second is *oblivious group storage* (OGS), the introduction of fine-grained data access control policies to the setting of *multi-client oblivious cloud storage* protocols.

Acknowledgments

I express my deepest gratitude to Dr. Nelly Fazio for being such a wonderful Ph.D. adviser and friend. Without Nelly, this dissertation would never have been possible. She introduced me to the field of cryptography during the senior year of my undergraduate studies, and advised me to pursue a doctoral degree in computer science. For that I'm truly grateful. During my Ph.D. years, she also made sure that I'm financially fit to devote as much time as possible on doctoral research by providing me with continuous funding. I appreciate her inspiration, guidance, and encouragement that made my life as a graduate student both enjoyable and productive. Nelly, it has been a privilege and a great honor to be your first Ph.D. student!

I thank my supervisory committee, Dr. Rosario Gennaro, Dr. Antonio Nicolosi, and Dr. William E. Skeith III, for providing helpful suggestions and constructive criticisms. A special thanks goes to Rosario for organizing weekly reading groups that exposed my colleagues and me to cutting-edge research results in cryptography.

I express my appreciation to the Graduate Center of CUNY for awarding me the Enhanced Chancellor's Fellowship to support my doctoral studies. I also thank Dr. Theodore Brown for admitting me to the doctoral program, and Dr. Robert Haralick for his academic advisement. Lots of thanks go to Lina Garcia and Dilvania Rodriguez for all their help with administrative matters.

Kudos to all my friends and family who helped me in a multitude of ways to reach this stage in my life. My heartfelt appreciation goes to my parents who have instilled within me a love for intellectual pursuits. I'm grateful to my father for sharing with me the stories of hardships he had to overcome during his childhood. Though I found them dull when I was a child, they later gave me a strong sense of perseverance, especially during the last five-year trek of Ph.D. studies. No words are enough to convey my gratitude to my loving wife and best friend, QiQi. She is the best thing that ever happened (and continues to happen) in my life!

To QiQi . . .

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The origins of cryptography are rooted almost exclusively in diplomatic, military, and government contexts, but in the last couple of decades, cryptography has rapidly moved into consumer applications. One does not have to look far to notice the widespread use of cryptography in our everyday lives. Whenever we unlock a garage door or a car using a remote-control device, connect to a WiFi network, pay for meals at a restaurant or order takeout on the Internet using a credit or debit card, make a video call via voice-over-IP, install a software update, or pay for public transportation using a transit pass we witness the omnipresence of cryptography. These ubiquitous applications of cryptography are fueled by our increasing reliance on digital technologies and our general necessity to prevent these technologies from being abused.

Historically, cryptography was considered the art of writing and solving codes. Therefore, proving the advertised security guarantees of the early cryptosystems such as Caesar cipher, Vigenère cipher, and Enigma machine was never an integral part of their design [73]. Consequently, all these early cryptosystems have been badly broken.[1] The first step in treating cryptography as a science was taken by Claude Shannon in his seminal paper published in 1949 under the title "Communication Theory of Secrecy

[1]Here we mean that a cryptosystems is broken when it is possible for an adversary to recover the message embedded in a ciphertext without knowing the secret key with overwhelming advantage.

Systems" [101]. It was in this paper that the first rigorous mathematical analysis of *private-key encryption* was presented. A couple of decades later, Whitfield Diffie and Martin Hellman published a paper titled "New Directions in Cryptography" [39] marking a pivotal point in the scientific study of cryptography. The novelty of Diffie-Hellman result was that it demonstrated how two people can securely communicate over an insecure channel without having a previously-agreed common shared key. This paper combined with the introduction of RSA cryptosystem [96] in 1978 by Ronald L. Rivest, Adi Shamir, and Leonard Adleman opened doors to the realm of *public-key cryptography.* Since then, the field of cryptography has observed a multitude of advances in terms of more specialized protocols, refined mathematical foundations, stronger security guarantees, *etc.* from the research community. Today, cryptography is a major research discipline with thousands of researchers and dozens of conferences.

Public-key encryption, which is about the secure point-to-point communication of messages, is undoubtedly the most popular application domain of public-key cryptography. The basic security property provided by a public-key encryption scheme is *data secrecy*, the guarantee that no ciphertxt reveals any non-trivial information regarding its encrypted message. This notion of data secrecy has been formalized under the terms semantic security [57] and non-malleability [44]. It has also been extended to more specialized settings such as *identity-based encryption* (IBE) [16], *hierarchical identity-based encryption* (HIBE) [53, 68], *broadcast encryption* (BE) [12, 49], *attribute-based encryption* (ABE) [63, 97], *etc.*

With the increased concerns over the privacy of the users in the recent years, another security property for public-key encryption came into existence under the term *anonymity* [11]. This security property guarantees that a ciphertext does not leak the identity of the user for whom it was encrypted. As of now, most of the existing settings of encryption have been extended to provide both data secrecy and anonymity properties. Examples of such extensions include *key-private public-key encryption*

[11], *anonymous (hierarchical) identity-based encryption* (A(H)IBE) [2,16,24], and *anonymous broadcast encryption* (AnoBE) [10].

In addition to preserving the user privacy, anonymous encryption schemes also allow additional cryptographic applications. One such application is *public-key encryption with keyword search* (PEKS) as Abdalla *et al.* noticed in [2]. In a PEKS scheme, each ciphertext is associated with a keyword with the requirement that the ciphertext does not leak any information regarding the keyword. A user is given along with his decryption key a trapdoor for each keyword he is authorized to use. Later, given a batch of ciphertexts stored in a remote database, a user can delegate the task of finding the ciphertexts associated to one of his keywords to an honest-but-curious database administrator by providing the corresponding trapdoor. Then, the database administrator can find all the ciphertexts associated with the keyword corresponding to the given trapdoor without knowing what the keyword is or which messages are encrypted in the ciphertexts. In [2], Abdalla *et al.* presented a generic framework that can construct a PEKS scheme by using an AIBE scheme as an underlying primitive. Clearly, the efficiency of the underlying anonymous encryption scheme plays an important role in these cryptographic applications of anonymity.

A major drawback for the efficiency of the state of the art anonymous broadcast encryption schemes is that their ciphertext lengths are linear in the number of receivers. As Kiayias and Samari showed in [76], this linear ciphertext length is a lower bound that is unfortunately unavoidable. A close examination of the result in [76] indicates that this highly inefficient overhead on the ciphertext length is a direct consequence of the very restrictive notion of anonymity that the existing anonymous broadcast encryption schemes are required to provide. Since any overhead on the ciphertext length directly translates to a communication overhead, the existing definition of anonytmity for broadcast encryption also narrows the range of its applications, especially when it comes to large-scale broadcast systems.

In this dissertation, we propose a less stringent yet very meaningful notion of anonymity for broadcast encryption called *outsider-anonymous broadcast encryption* (oABE). Since our notion of *oursider anonymity* is not subject to the lower bound presented in [76], our oABE schemes enjoy much more efficient, sublinear ciphertext lengths in the number of receivers. We also present two very interesting cryptographic applications allowed by the efficiency of our oABE schemes. The first extends the state of the art setting of point-to-point steganography to the multi-recipient setting. The second introduces fine-grained data access control policies to the existing setting of multi-client oblivious cloud storage protocols.

1.1 Organization of the Dissertation

The rest of this dissertation is organized into four chapters, namely Preliminaries (Chapter 2), Outsider-Anonymous Broadcast Encryption (Chapter 3), Broadcast Steganography (Chapter 4), and Oblivious Group Storage (Chapter 5). A brief summary of each of these chapters is given below.

In Chapter 2, we present the background information prerequisite for the rest of this dissertation. The material presented in this chapter include mathematical notations, Diffie-Hellman assumptions, hash functions, cryptographic primitives, and a summary of the subset cover framework [89].

In Chapter 3, we formally define the notion of outsider-anonymous broadcast encryption and construct several oABE schemes with varying security guarantees and levels of efficiency. Each of our oABE constructions is also accompanied by a rigorous mathematical proof of security.

Chapter 4 contains our first application of the notion of outsider anonymity: the extension of point-to-point model of steganography to the broadcast setting. After formally defining the setting and the security model of this new type of steganography that we call *broadcast steganography* (BS), we also present two provably secure constructions of BS along with their proofs of security. *En route* to our BS constructions, we also construct and prove the security of a new kind of an oABE scheme that produces *pseudorandom ciphertexts*.

The last chapter (Chapter 5) presents our second application of the notion of outsider anonymity. In this application, we extend the state of the art multi-client oblivious cloud storage model to provide shared data access with fine-grained data access control policies. In the first part of this chapter, we formally define the setting and the security model of our new type of an oblivious cloud storage protocol that we term *oblivious group storage* (OGS). The second part consists of our new OGS construction and its proof of security.

Chapter 2

Preliminaries

2.1 Notations

Given below are some notations that we are using throughout this dissertation.

- $:=$ denotes a definition or a deterministic assignment where the right hand side (RHS) is the definiens and the left hand side (LHS) is the definiendum.
- $\leftarrow$ denotes a random assignment where the RHS is a probabilistic polynomial time (PPT) algorithm. The random tape of this execution of the PPT algorithm is chosen uniformly at random.
- $\leftarrow_{\$}$ denotes a random assignment where the RHS is a finite set. The element assigned to the LHS from this set is picked uniformly at random.
- $\{0,1\}^n$ denotes the set of all bit-strings of length n.
- $\perp$ denotes a special output of an algorithm indicating a failure.
- $\mathbb{G}$ denotes a finite group, $\mathbb{Z}$ denotes the set of integers.
- For $n \in \mathbb{Z}$, $\mathbb{Z}_n$ denotes the additive group of integers modulo n and $\mathbb{Z}_n^*$ denotes the multiplicative group of integers modulo n.

- $\|$ denotes the string concatenation operation. Specifically, if s_1 and s_2 are two bit-strings, then $s_1\|s_2$ denotes the concatenation of s_1 and s_2.
- For a given vector $\vec{a}$ and an element b, we denote by $\vec{a} : b$ the vector obtained by appending b at the end of the vector $\vec{a}$.
- $\mathsf{Prfx}(\vec{a})$ denotes the set of all prefix vectors of $\vec{a}$ with non-zero length.

2.2 Diffie-Hellman Assumptions

Let $\mathbb{G} = \langle g \rangle$ be a group with a generator g and order q. Define the function dh as

$$\begin{aligned} \mathsf{dh} : \quad & \mathbb{G}^2 \to \mathbb{G} \\ & \mathsf{dh}(X, Y) := Z, \end{aligned} \tag{2.1}$$

where $X = g^x, Y = g^y$, and $Z = g^{xy}$ for $x, y \in \mathbb{Z}_q$.

2.2.1 Computational Diffie-Hellman Assumption

The computational Diffie-Hellman (CDH) problem is to compute $\mathsf{dh}(X, Y)$ given two random group elements $X, Y \in \mathbb{G}$. Formally, we say that the CDH problem is (t, ϵ)-hard relative to the group $\mathbb{G}$ if for all t-time adversaries $\mathcal{A}$, we have

$$\left|\Pr\left[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y) = \mathsf{dh}(g^x, g^y)\right]\right| \leq \epsilon,$$

where $x, y \leftarrow_\$ \mathbb{Z}_q$ and the probability is computed over the random coins used to generate the exponents x, y and by $\mathcal{A}$. The CDH assumption was first introduced by Diffie and Hellman in [39].

2.2.2 Decisional Diffie-Hellman Assumption

The decisional Diffie-Hellman (DDH) problem is to distinguish the two distributions $(X, Y, \mathsf{dh}(X, Y))$ and (X, Y, Z) for three random group elements $X, Y, Z \in \mathbb{G}$. Formally, we say that the DDH problem is (t, ϵ)-hard relative to the group $\mathbb{G}$ if for all t-time adversaries $\mathcal{A}$, we have

$$\left|\Pr\left[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1\right] - \Pr[\mathcal{A}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1]\right| \leq \epsilon,$$

where $x, y, z \leftarrow\$ \mathbb{Z}_q$ and the probability is computed over the random coins used to generate the exponents x, y, z and by $\mathcal{A}$. The DDH assumption was also introduced by Diffie and Hellman in [39].

2.2.3 Strong Twin Computational Diffie-Hellman Assumption

Let $\mathbb{G} = \langle g \rangle$ be a group with a generator g and order q. Also, let the function dh be defined as in Equation (2.1). Define the function $\mathsf{2dh}$ as

$$\begin{aligned} \mathsf{2dh}: \quad & \mathbb{G}^3 \to \mathbb{G}^2 \\ & \mathsf{2dh}(X_1, X_2, Y) \coloneqq (\mathsf{dh}(X_1, Y), \mathsf{dh}(X_2, Y)). \end{aligned}$$

For two fixed group elements $X_1, X_2 \in \mathbb{G}$, also define the predicate $\mathsf{2dhp}$ as follows.

$$\begin{aligned} \mathsf{2dhp}: \quad & \mathbb{G}^3 \to \{\mathsf{True}, \mathsf{False}\} \\ & \mathsf{2dhp}(X_1, X_2, \hat{Y}, \hat{Z}_1, \hat{Z}_2) \coloneqq \mathsf{2dh}(X_1, X_2, \hat{Y}) \stackrel{?}{=} (\hat{Z}_1, \hat{Z}_2) \end{aligned}$$

The strong twin computational Diffie-Hellman problem (s2CDH) problem is to compute $\mathsf{2dh}(X_1, X_2, Y)$ given random group elements $X_1, X_2, Y \in \mathbb{G}$ along with access to an oracle that provides answers to the predicate $\mathsf{2dhp}(X_1, X_2, \cdot, \cdot, \cdot)$. Formally, We say that the s2CDH problem is (t, ϵ)-hard relative to $\mathbb{G}$ if for all t-time adversaries $\mathcal{A}$, it is the case that

$$\left|\Pr\left[\mathcal{A}^{\mathsf{2dhp}(g^{x_1}, g^{x_2}, \cdot, \cdot, \cdot)}(g^{x_1}, g^{x_2}, g^y) = \mathsf{2dh}(g^{x_1}, g^{x_2}, g^y)\right]\right| \leq \epsilon,$$

where $x_1, x_2, y \leftarrow_{\$} \mathbb{Z}_q$ and the probability is computed over the randomness used to sample the exponents x_1, x_2, y and by $\mathcal{A}$. The s2CDH problem was introduced by Cach *et al.* in [29]. The authors also showed in [29] that the s2CDH assumption is equivalent to the CDH assumption.

2.3 Hashing Functions

2.3.1 Entropy-Smoothing Hashing

A family of hash functions $\mathcal{H}_{es} = \{H : X \to Y\}$ is "entropy-smoothing" [69] if it is hard to distinguish $(H, H(x))$ from (H, y), where H is a random element of $\mathcal{H}_{es}$, x is a random element of X, and y is a random element of Y. More formally, $\mathcal{H}_{es}$ is called (t, ϵ)-entropy-smoothing if for every t-time adversary $\mathcal{A}$,

$$\Big|\Pr[\mathcal{A}(H,H(x)) = 1 \mid H \leftarrow_{\$} \mathcal{H}_{es}, x \leftarrow_{\$} X] \\ - \Pr[\mathcal{A}(H, y) = 1 \mid H \leftarrow_{\$} \mathcal{H}_{es}, y \leftarrow_{\$} Y]\Big| \leq \epsilon,$$

where the probability is over the choice of H, x, y and over the random coins used by $\mathcal{A}$. Entropy smoothing is related to strong randomness extraction [120], but it is a much less stringent (and hence easier to realize) notion, as it seeks only computational (rather than information-theoretic) guarantees, and it is specific to *one* entropy source (the uniform distribution over the domain X), whereas strong extractors are applicable to any source of a given min-entropy.

2.3.2 Strong 2-Universal Hashing

A family of hash functions $\mathcal{H}_{s2u} = \{H : X \to Y\}$ is strong 2-universal [113] if for all distinct $x_1, x_2 \in X$ and all (not necessarily distinct) $y_1, y_2 \in Y$, exactly $\frac{|\mathcal{H}_{s2u}|}{|Y|^2}$ functions of $\mathcal{H}_{s2u}$ map x_1 to y_1 and x_2 to y_2. More formally,

$$\Pr\Big[H(x_1) = y_1 \wedge H(x_2) = y_2 \mid H \leftarrow_{\$} \mathcal{H}_{es}; \\ x_1, x_2 \leftarrow_{\$} X; x_1 \neq x_2; y_1, y_2 \leftarrow_{\$} Y\Big] = \frac{1}{|Y|^2}.$$

2.4 Cryptographic Primitives

2.4.1 Encapsulation Mechanism

An encapsulation mechanism can be thought of as a “relaxed” commitment scheme. While a commitment scheme allows the sender to commit to any message of his choosing, an encapsulation mechanism forces the sender to commit to a random bit-string. This notion was originally proposed by Boneh and Katz [20].

Definition 2.4.1 (Encapsulation Mechanism Setting): The formal setting of an encapsulation mechanisms consists a tuple of algorithms $\Pi = (\mathsf{SetupCom}, \mathsf{Commit}, \mathsf{Open})$ defined as follows.

$\mathsf{PK} \leftarrow \mathsf{SetupCom}(1^\lambda)$: $\mathsf{SetupCom}$ algorithm takes the security parameter λ as an input and produces a commitment public key PK.

$(\hat{r}, \mathsf{com}, \mathsf{decom}) \leftarrow \mathsf{Commit}(\mathsf{PK})$: Commit algorithm takes as input the public key PK. After sampling a random bit-string $\hat{r} \in \{0,1\}^\lambda$ together with associated commitment and decommitment values com and decom, Commit algorithm outputs the tuple $(\hat{r}, \mathsf{com}, \mathsf{decom})$.

$\hat{r}/\bot := \mathsf{Open}(\mathsf{PK}, \mathsf{com}, \mathsf{decom})$: Given a public key PK, a commitment value com, and a decommitment value decom, the Open algorithm either outputs $\hat{r} \in \{0,1\}^\lambda$ or the failure symbol $\bot$. We assume that $\mathsf{Decrypt}$ is deterministic.

Correctness. If PK is a commitment public key output by $\mathsf{SetupCom}(1^\lambda)$ and $(\hat{r}, \mathsf{com}, \mathsf{decom}) \leftarrow \mathsf{Commit}(\mathsf{PK})$, then it must be the case that

$$\mathsf{Open}(\mathsf{PK}, \mathsf{com}, \mathsf{decom}) = \hat{r},$$

except with negligible probability in the security parameter λ. $\Diamond$

A secure encapsulation mechanism provides two aspects of security: *hiding* and *relaxed binding*. The hiding aspect of security guarantees that the triples of the form $(\mathsf{PK}, \mathsf{com}, \hat{r})$ are statistically indistinguishable from those of the form $(\mathsf{PK}, \mathsf{com}, r)$ for random $r \in \{0,1\}^\lambda$. The relaxed binding property assures that given an output $(\hat{r}, \mathsf{com}, \mathsf{decom})$ of $\mathsf{Commit}(\mathsf{PK})$, it is hard to produce a decommitment value decom' such that $\mathsf{Open}(\mathsf{PK}, \mathsf{com}, \mathsf{decom}') \notin \{\hat{r}, \bot\}$.

2.4.2 Strong Existentially Unforgeable One-Time Signature

A one-time signature scheme is a digital signature scheme that allows at most one message to be signed per key pair. Signing more than one message per key pair breaks the security of the scheme. One-time signatures are an old notion, originally proposed in 1979 by Lamport [79].

Definition 2.4.2 (One-time Signature Setting): The formal setting of one-time signatures consists of a message space $\mathcal{MSP}$, a signature space $\mathcal{SSP}$, and a tuple of algorithms $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ defined as follows.

$(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$: The key-generation algorithm Gen takes as input the security parameter λ and outputs a key pair $(\mathsf{VK}, \mathsf{SK})$, where SK is used to sign a message and VK is used to later verify that message/signature pair.

$\sigma \leftarrow \mathsf{Sign}(\mathsf{SK}, m)$: The signing algorithm Sign takes as input a signing key SK and a message $m \in \mathcal{MSP}$ and outputs a signature $\sigma \in \mathcal{SSP}$.

$\{0,1\} := \mathsf{Vrfy}(\mathsf{VK}, \sigma, m)$: Given a verification key VK, a signature $\sigma \in \mathcal{SSP}$, and a message $m \in \mathcal{MSP}$, the verification algorithm Vrfy either outputs 0 or 1. We assume that Vrfy is deterministic.

Correctness. For every $m \in \mathcal{MSP}$, if $(\mathsf{VK}, \mathsf{SK})$ is a signature/verification key pair output by $\mathsf{Gen}(1^\lambda)$, then it must be the case that

$$\mathsf{Vrfy}(\mathsf{VK}, \mathsf{Sign}(\mathsf{SK}, m), m) = 1,$$

except with negligible probability in the security parameter λ. $\Diamond$

The definition of security for one-time signature schemes we present here is termed *strong existential unforgeability against a chosen-message attack* or SIG-SEU for short. An existentially unforgeable signature scheme ensures that a PPT adversary who is given the signatures for a few messages of his choosing is not be able to produce a signature for a new message. Strong existential unforgeability guarantees that such an adversary cannot even produce a new signature for a previously signed message. The notion of strong existential unforgeability can be formalized as the following game between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

Definition 2.4.3 (SIG-SEU Game): For a given one-time signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$, the strong existential unforgeability game, which is played between a PPT challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, is defined as follows.

Setup: $\mathcal{C}$ executes the Gen algorithm Q_S times and returns the generated verification keys $\mathsf{VK}_1, \ldots, \mathsf{VK}_{Q_S}$ to $\mathcal{A}$. $\mathcal{C}$ also initializes the set of returned queries $R_Q := \emptyset$.

Query: $\mathcal{A}$ adaptively requests Q_S signatures, at most one per verification key, from $\mathcal{C}$. For each such request, $\mathcal{A}$ sends a verification key index i and a message m_i of his choosing. $\mathcal{C}$ computes the signature $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{SK}_i, m_i)$, adds (m_i, σ_i, i) to R_Q, and returns σ_i to $\mathcal{A}$.

Response: $\mathcal{A}$ returns a tuple (m^*, σ^*, i^*) to $\mathcal{C}$. We say that $\mathcal{A}$ wins the SIG-SEU game if $\mathsf{Vrfy}(\mathsf{VK}_{i^*}, \sigma^*, m^*) = 1$ and $(m^*, \sigma^*, i^*) \notin R_Q$.

We refer to such an adversary $\mathcal{A}$ as a SIG-SEU adversary. The advantage of $\mathcal{A}$ winning the above game is defined as,

$$\mathsf{Adv}_{\mathcal{A},\Sigma}^{\text{SIG-SEU}} := \Pr[\mathsf{Vrfy}(\mathsf{VK}_{i^*}, \sigma^*, m^*) = 1 \wedge (m^*, \sigma^*, i^*) \notin R_Q],$$

where the probability is over the random bits used by $\mathcal{A}$ and $\mathcal{C}$. ◊

Definition 2.4.4 (SIG-SEU Security): A signature scheme Σ is (t, Q_S, ϵ)-SIG-SEU-secure if for any t-time SIG-SEU adversary making at most Q_S adaptive signature queries, we have that $\mathsf{Adv}_{\mathcal{A},\Sigma}^{\text{SIG-SEU}} \leq \epsilon$. ◊

2.4.3 Anonymous (Hierarchical) Identity-Based Encryption

Identity-Based Encryption. *Identity-based encryption* is a public-key encryption scheme where the public key of a user can be an arbitrary bit-string such as an email address. A *system initialization authority* (SIA) possessing a master secret key produces a secret key corresponding to a given identity. IBE greatly simplifies the problem of key distribution and management in public-key encryption since now the users don't have to worry about exchanging long and unintelligible public keys. Although this concept was suggested by Shamir in 1984 [100], an efficient and provably secure construction was not proposed until the work of Boneh and Franklin in 2001 [16]. Since then, there have been several IBE constructions (*e.g.*, [13, 17, 112]).

Hierarchical Identity-Based Encryption. Since having a single key generation authority is undesirable due to the computational overhead in a large network of users, the notion of *hierarchical identity-based encryption* was proposed in [53, 68]. HIBE is a generalized version of IBE that allows delegation of keys in a hierarchical structure. At the top of this hierarchy is the system initialization authority. At the following level are several sub-authorities. Each of these sub-authorities holds a delegation key that allows it to decrypt the ciphertexts destined to it as well as to the users below it in the hierarchy. Each user belonging to a sub-authority is also allowed to extend the hierarchy by becoming a sub-sub-authority, and these levels of delegation can extend further. Since the introduction, there have been several HIBE constructions in the literature (*e.g.*, [14, 52, 82, 85, 112]).

Anonymous (Hierarchical) Identity-Based Encryption. The notion of *anonymous (hierarchical) identity-based encryption* is a direct extension of the notion of key privacy in public-key encryption [11] to the setting of (hierarchical) identity-based encryption. Specifically, an A(H)IBE scheme is anonymous if its ciphertexts do not leak the identities of the recipients. This concept was originally proposed by Abdalla *et al.* in [2] where they investigated public-key encryption with keyword search (PEKS) [15]. PEKS is a system where each ciphertext is associated with a keyword with the requirement that the ciphertext does not leak any information regarding the keyword. A user is given along with his decryption key a trapdoor for each keyword he is authorized to use. Now, given a batch of ciphertexts (*e.g.*, stored in a remote database), a user can delegate the task of finding the ciphertexts associated to one of his keywords to an honest-but-curious third party (*e.g.*, database administrator) by giving that third party the corresponding trapdoor. Then, the third party can find all the ciphertexts associated with the keyword corresponding to the given trapdoor without knowing what the keyword is or which messages are encrypted in the ciphertexts. In [2], Abdalla *et al.* presented a framework that constructed a

PEKS scheme by using an AIBE scheme as an underlying primitive.

Although the authors in [2] introduced the notion of A(H)IBE, they didn't provide any concrete constructions. They did, however, notice that the very first IBE scheme by Boneh and Franklin [16] was indeed anonymous in the random oracle model. The first AIBE scheme secure in the standard model was proposed by Gentry in [51] and the first AHIBE scheme also secure in the standard model was proposed by Boyen and Waters in [24]. Since then, there have been several other constructions with various useful properties and improvements [4,8,28,37,46,81,94,98,99,103]. Given below are the definitions of the AIBE and AHIBE schemes.

Definition 2.4.5 (AIBE Setting): An anonymous identity-based encryption scheme, associated with an identity space $\mathcal{ISP}$, a message space $\mathcal{MSP}$, and a ciphertext space $\mathcal{CSP}$, is a tuple of algorithms $(\mathsf{Setup}, \mathsf{Extract}, \mathsf{Encrypt}, \mathsf{Decrypt})$ defined as follows.

$(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$: Setup takes the security parameter 1^λ as input and outputs the master public key MPK and the master secret key MSK.

$sk_I \leftarrow \mathsf{Extract}(\mathsf{MPK}, \mathsf{MSK}, I)$: $\mathsf{Extract}$ takes the master public key MPK, the master secret key MSK, and an identity $I \in \mathcal{ISP}$ as inputs and outputs a secret key sk_I for the identity I.

$c \leftarrow \mathsf{Encrypt}(\mathsf{MPK}, I, m)$: $\mathsf{Encrypt}$ takes the master public key MPK, an identity $I \in \mathcal{ISP}$, and a message $m \in \mathcal{MSP}$ as inputs and outputs a ciphertext $c \in \mathcal{CSP}$.

$m/\perp := \mathsf{Decrypt}(\mathsf{MPK}, sk_I, c)$: Given the master public key MPK, a secret key sk_I, and a ciphertext $c \in \mathcal{CSP}$, $\mathsf{Decrypt}$ either outputs a message $m \in \mathcal{MSP}$ or the failure symbol $\perp$. $\mathsf{Decrypt}$ is assumed to be deterministic.

Correctness. For every $I \in \mathcal{ISP}$, and every $m \in \mathcal{MSP}$, if sk_I is the secret key output by $\mathsf{Extract}(\mathsf{MPK}, \mathsf{MSK}, I)$, then

$$\mathsf{Decrypt}(\mathsf{MPK}, sk_I, \mathsf{Encrypt}(\mathsf{MPK}, I, m)) = m,$$

except with negligible probability in the security parameter λ. $\Diamond$

Definition 2.4.6 (AHIBE Setting): An anonymous hierarchical identity-based encryption scheme, associated with an identity space $\mathcal{ISP}$, a message space $\mathcal{MSP}$, and a ciphertext space $\mathcal{CSP}$, is a tuple of algorithms ($\mathsf{Setup}, \mathsf{Extract}, \mathsf{Delegate}, \mathsf{Encrypt}$, $\mathsf{Decrypt}$) defined as follows.

$(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, \ell)$: Setup takes the security parameter 1^λ and the maximum depth of the hierarchy ℓ and outputs the master public key MPK and the master secret key MSK.

$sk_{\vec{I}} \leftarrow \mathsf{Extract}(\mathsf{MPK}, \mathsf{MSK}, \vec{I})$: $\mathsf{Extract}$ takes the master public key MPK, the master secret key MSK, and an identity vector $\vec{I} \in \mathcal{ISP}^d$ such that $1 \leq d \leq \ell$ as inputs and outputs a secret key $sk_{\vec{I}}$ for the identity vector $\vec{I}$.

$sk_{\vec{I}:I'} \leftarrow \mathsf{Delegate}(\mathsf{MPK}, sk_{\vec{I}}, I')$: $\mathsf{Delegate}$ takes the master public key MPK, a secret key $sk_{\vec{I}}$ for the identity vector $\vec{I} \in \mathcal{ISP}^d$ such that $1 \leq d < \ell$, and an identity $I' \in \mathcal{ISP}$ as inputs and outputs a secret key $sk_{\vec{I}:I'}$ for the identity vector $\vec{I} : I' \in \mathcal{ISP}^{d+1}$.

$c \leftarrow \mathsf{Encrypt}(\mathsf{MPK}, \vec{I}, m)$: $\mathsf{Encrypt}$ takes the master public key MPK, an identity vector $\vec{I} \in \mathcal{ISP}^d$ such that $1 \leq d \leq \ell$, and a message $m \in \mathcal{MSP}$ as inputs and outputs a ciphertext $c \in \mathcal{CSP}$.

$m/\perp := \mathsf{Decrypt}(\mathsf{MPK}, sk_{\vec{I}}, c)$: Given the master public key MPK, a secret key $sk_{\vec{I}}$, and a ciphertext $c \in \mathcal{CSP}$, $\mathsf{Decrypt}$ either outputs a message $m \in \mathcal{MSP}$ or the failure symbol $\perp$. $\mathsf{Decrypt}$ is assumed to be deterministic.

Correctness. For every $\vec{I} \in \mathcal{ISP}^d$ such that $1 \leq d \leq \ell$, and every $m \in \mathcal{MSP}$, if $sk_{\vec{I}}$ is the secret key properly generated for the identity $\vec{I}$ (*i.e.*, either executing $\mathsf{Extract}$ or $\mathsf{Delegate}$), then

$$\mathsf{Decrypt}(\mathsf{MPK}, sk_{\vec{I}}, \mathsf{Encrypt}(\mathsf{MPK}, \vec{I}, m)) = m,$$

except with negligible probability in the security parameter λ. $\Diamond$

We now review the formal security models related to A(H)IBE schemes. Following the work in the literature [4, 46, 81, 98], we present these security models as games played between a PPT challenger and an adversary. In a nutshell, the goal of the adversary in these games is to tell apart two ciphertexts generated under two different identities of which he does not own the corresponding secret keys. Depending on the game in question, the adversary is also granted some privileges.

We follow a unified approach in the presentation of these games. As such, we first present the games related to the X-IND-CCA notions of security where $\mathrm{X} \in \{\mathrm{AIBE}, \mathrm{AHIBE}\}$. In the material that follows, we show how to tweak these games to obtain the X-IND-CPA variations.

Definition 2.4.7 (X-IND-CCA Game for $\mathbf{X \in \{AIBE, AHIBE\}}$): For a given A(H)IBE scheme, the X-IND-CCA game for $\mathrm{X} \in \{\mathrm{AIBE}, \mathrm{AHIBE}\}$ played between a PPT challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, is defined as follows.

Setup: $\mathcal{C}$ runs

$$(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \begin{cases} \mathsf{Setup}(1^\lambda) & \text{if X} = \text{AIBE} \\ \mathsf{Setup}(1^\lambda, \ell) & \text{otherwise} \end{cases}$$

and gives $\mathcal{A}$ the resulting master public key MPK, keeping the master secret key MSK to itself. $\mathcal{C}$ also initializes the set of revoked identities $R_I := \emptyset$.

Phase 1: $\mathcal{A}$ adaptively issues queries of the following types.

Secret-key query $\vec{I}$: $\mathcal{A}$ requests the secret key of the identity $\vec{I} \in \mathcal{ISP}^d$ where $d = 1$ if $\mathrm{X} = \mathrm{AIBE}$, $1 \leq d \leq \ell$ otherwise. $\mathcal{C}$ runs $sk_{\vec{I}} \leftarrow \mathsf{Extract}(\mathsf{MPK}, \mathsf{MSK}, \vec{I})$, adds $\vec{I}$ to R, and sends $sk_{\vec{I}}$ to $\mathcal{A}$.

Decryption query $(\vec{I}, c)$: $\mathcal{A}$ issues a decryption query on an identity $\vec{I} \in \mathcal{ISP}^d$, where $d = 1$ if $\mathrm{X} = \mathrm{AIBE}$, $1 \leq d \leq \ell$ otherwise, and a ciphertext $c \in \mathcal{CSP}$. $\mathcal{C}$ computes $sk_{\vec{I}} \leftarrow \mathsf{Extract}(\mathsf{MPK}, \mathsf{MSK}, \vec{I})$, runs $\mathsf{Decrypt}(\mathsf{MPK}, sk_{\vec{I}}, c)$, and gives the result to $\mathcal{A}$.

Challenge: $\mathcal{A}$ gives $\mathcal{C}$ two *equal* length messages $m_0^*, m_1^* \in \mathcal{MSP}$ and two identities $\vec{I}_0^*, \vec{I}_1^* \in \mathcal{ISP}^d$ with the following restrictions:

1. $(\mathsf{Prfx}(\vec{I}_0^*) \cup \mathsf{Prfx}(\vec{I}_1^*)) \cap R = \emptyset$.
2. $d = 1$ if X = AIBE, $1 \leq d \leq \ell$ otherwise.

$\mathcal{C}$ picks $b^* \leftarrow\$ \{0,1\}$, computes $c^* \leftarrow \mathsf{Encrypt}(\mathsf{MPK}, \vec{I}_{b^*}^*, m_{b^*}^*)$, and sends c^* to $\mathcal{A}$.

Phase 2: The interaction between $\mathcal{A}$ and $\mathcal{C}$ in this phase is similar to *Phase 1* with two restrictions as given below.

Secret-key query $\vec{I}$: $\vec{I} \notin (\mathsf{Prfx}(\vec{I}_0^*) \cup \mathsf{Prfx}(\vec{I}_1^*))$.

Decryption query $(\vec{I}, c)$: If $\vec{I} \in (\mathsf{Prfx}(\vec{I}_0^*) \cup \mathsf{Prfx}(\vec{I}_1^*))$, then $c \neq c^*$.

Guess: $\mathcal{A}$ outputs a guess $b \in \{0,1\}$ and wins if $b = b^*$.

The adversary $\mathcal{A}$ in this game is called an X-IND-CCA adversary and $\mathcal{A}$'s advantage is defined as

$$\mathsf{Adv}_{\mathcal{A},\Pi}^{\text{X-IND-CCA}} := \left|\Pr[b = b^*] - \tfrac{1}{2}\right|,$$

where the probability is over the random coins used by $\mathcal{C}$ and $\mathcal{A}$. $\Diamond$

Remark 2.4.8. In [27], Canetti *et al.* proposed a weaker notion of security called *selective-ID* security. In contrast to the *full* security game that we presented in Definition 2.4.7 above, the adversary in the selective-ID security game is required to output the challenge identities $\vec{I}_0^*, \vec{I}_1^*$ before the public parameters are generated by the challenger. This weakened notion of security has allowed the realization of early AHIBE constructions [4, 8, 24, 46, 81, 99, 103].

Definition 2.4.9: An A(H)IBE scheme Π is (t, Q_I, Q_D, ϵ)-X-IND-CCA-secure for X $\in$ $\{\text{AIBE}, \text{AHIBE}\}$ if for any t-time X-IND-CCA adversary making at most Q_I adaptive secret-key queries and Q_D adaptive decryption queries, we have $\mathsf{Adv}^{\text{X-IND-CCA}}_{\mathcal{A},\Pi} \leq \epsilon$. $\Diamond$

By not allowing the adversary to submit any decryption queries in *Phase 1* and *Phase 2* of the X-IND-CCA game, one obtains the X-IND-CPA game.

Definition 2.4.10: An A(H)IBE scheme Π is (t, Q_I, ϵ)-X-IND-CPA-secure for X $\in$ $\{\text{AIBE}, \text{AHIBE}\}$ if Π is $(t, Q_I, 0, \epsilon)$-X-IND-CCA-secure. $\Diamond$

Weakly Robust Anonymous Identity-Based Encryption. The *robust encryption*, formalized by Abdalla *et al.* [3], requires that it is hard to produce a ciphertext that is valid for two different users. In [3], the authors define two types of robustness, strong and weak. Informally, an AIBE scheme is called *weakly robust*, if any adversary has negligible advantage in producing two identities I_0, I_1 and a message m such that the encryption of m under I_0 can be decrypted with the private key associated with I_1 leading to a non-$\perp$ result. The authors also provide a transformation algorithm which makes possible to obtain a weakly robust AIBE scheme from a regular AIBE one.

2.4.4 Broadcast Encryption

Conventional encryption schemes provide the means for secret transmission of data in point-to-point communication. The setting of *broadcast encryption* [12, 49], instead, consists of a *sender*, an insecure unidirectional *broadcast channel*, and a universe of *receivers*. When the sender wants to transmit some confidential data, it specifies the set of authorized receivers and creates an encrypted version of the content. A secure BE scheme enables legitimate receivers to recover the original content, while ensuring that excluded users just obtain meaningless data, even in the face of collusions. The formal setting of broadcast encryption is given below.

Definition 2.4.11 (BE Setting): A broadcast encryption scheme, associated with a universe of users $U = [1, N]$, a message space $\mathcal{MSP}$, and a ciphertext space $\mathcal{CSP}$, is a tuple of algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ defined as follows.

$(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, N)$: The Setup algorithm takes as input the security parameter 1^λ and the number of users in the system N. It outputs the master public key MPK and the master secret key MSK of the system.

$sk_i \leftarrow \mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)$: The key generation algorithm KeyGen takes as input the master public key MPK, the master secret key MSK, and a user $i \in U$. It outputs the secret key sk_i of the user i.

$c \leftarrow \mathsf{Encrypt}(\mathsf{MPK}, S, m)$: The $\mathsf{Encrypt}$ algorithm takes as input the master public key MPK, the set of receivers $S \subseteq U$, and a message $m \in \mathcal{MSP}$. It then outputs a ciphertext $c \in \mathcal{CSP}$.

$m/\perp := \mathsf{Decrypt}(\mathsf{MPK}, S, sk_i, c)$: Given the master public key MPK, a secret key sk_i, the set of receivers $S \subseteq U$, and a ciphertext $c \in \mathcal{CSP}$, the $\mathsf{Decrypt}$ algorithm either outputs a message $m \in \mathcal{MSP}$ or the failure symbol $\perp$. We assume that $\mathsf{Decrypt}$ is deterministic.

Correctness. For every $S \subseteq U$, every $i \in S$, and every $m \in \mathcal{MSP}$, if sk_i is the secret key output by $\mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)$, then it must be the case that

$$\mathsf{Decrypt}(\mathsf{MPK}, S, sk_i, \mathsf{Encrypt}(\mathsf{MPK}, S, m)) = m,$$

except with negligible probability in the security parameter λ. $\Diamond$

There are two main models of security for BE schemes: security against chosen plaintext attack (BE-IND-CPA) and security against chosen ciphertext attack (BE-IND-CCA). These security models are defined as games between a PPT adversary and a challenger. For completeness, we present these definitions below.

Definition 2.4.12 (BE-IND-CCA Game): For a given BE scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$, the BE-IND-CCA game is defined between a PPT adversary $\mathcal{A}$, and a challenger $\mathcal{C}$ as follows.

Setup: $\mathcal{C}$ runs $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, N)$ and gives $\mathcal{A}$ the resulting master public key MPK, keeping the master secret key MSK to itself. $\mathcal{C}$ also initializes the set of revoked users R_U to be empty.

Phase 1: $\mathcal{A}$ adaptively issues queries of the following types.

Secret-key query $\boldsymbol{i}$: $\mathcal{A}$ requests the secret key of the user $i \in U$. $\mathcal{C}$ runs $sk_i \leftarrow \mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)$ to generate the secret key sk_i of the user i, adds i to R_U, and sends sk_i to $\mathcal{A}$.

Decryption query $\boldsymbol{(S, i, c)}$: $\mathcal{A}$ issues a decryption query where $S \subseteq U$, $i \in U$, and $c \in \mathcal{CSP}$. First, $\mathcal{C}$ runs $sk_i \leftarrow \mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)$ to generate the secret key sk_i of user i. Then, it runs $\mathsf{Decrypt}(\mathsf{MPK}, S, sk_i, c)$ and returns the output to $\mathcal{A}$.

Challenge: $\mathcal{A}$ provides $\mathcal{C}$ with two equal length messages $m_0^*, m_1^* \in \mathcal{MSP}$. $\mathcal{C}$ sets $S^* := U \setminus R_U$, picks a random bit $b^* \in \{0,1\}$, runs $c^* \leftarrow \mathsf{Encrypt}(\mathsf{MPK}, S^*, m_{b^*}^*)$, and returns c^* to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ adaptively issues queries of the following type.

Decryption query $\boldsymbol{(S, i, c)}$: This type of query is handled by $\mathcal{C}$ as in *Phase 1* with one exception: if $i \in S^*$ and $c = c^*$, $\mathcal{C}$ rejects the query.

Guess: $\mathcal{A}$ outputs a guess $b \in \{0,1\}$ and wins if $b = b^*$.

We refer to such an adversary $\mathcal{A}$ as a BE-IND-CCA adversary. The advantage of $\mathcal{A}$ winning the above game is defined as,

$$\mathsf{Adv}_{\mathcal{A},\Pi}^{\text{BE-IND-CCA}} := \left|\Pr[b = b^*] - \tfrac{1}{2}\right|,$$

where the probability is over the random bits used by $\mathcal{A}$ and $\mathcal{C}$. $\Diamond$

Definition 2.4.13 (BE-IND-CCA Security): A BE scheme Π is (t, Q_U, Q_D, ϵ)-BE-IND-CCA-secure if for any t-time BE-IND-CCA adversary making at most Q_U adaptive secret-key queries and at most Q_D chosen decryption queries, we have that $\mathsf{Adv}^{\text{BE-IND-CCA}}_{\mathcal{A},\Pi} \leq \epsilon$. $\Diamond$

Definition 2.4.14 (BE-IND-CPA Security): A BE scheme Π is (t, Q_U, ϵ)-BE-IND-CPA-secure if Π is $(t, Q_U, 0, \epsilon)$-BE-IND-CCA-secure. $\Diamond$

2.4.5 Steganography

The point-to-point encryption schemes are effective at concealing the *meaning* of the communication between two parties. If the parties additionally require that the very *existence* of their communication over a public communication medium remains concealed, then the required tool is *steganography* [104]. Conventional steganography allows *two* parties to communicate covertly, even in the presence of an adversary, by *hiding* the intended content within seemingly harmless messages, also known as the *communication channel*.

In the following, we present the formal definition of a steganography scheme, including the existing formal models of security defined for such a scheme. A crucial aspect of any steganographic protocol is the underlying communication channel in which the secret messages are sent. We also present the formal definitions of related to this stegotext-transmission medium below.

Definition 2.4.15 (Documents & Covertexts): Let $\Sigma = \{0,1\}^\sigma$ be a finite set of bit-strings with length σ. Denote by Σ^* the set of sequences of finite length over Σ. We call the strings $u \in \Sigma$ *documents* and the strings $s \in \Sigma^*$ *covertexts*. $\Diamond$

Definition 2.4.16 (Channels): A *channel* $\mathfrak{C}_h$ is a function that takes as input a *channel history* $h \in \Sigma^*$ and produces a probability distribution on Σ. A channel history

$h = s_1\| \ldots \| s_l \in \Sigma^*$ is called *legal* if for all $i \in [1, l]$, $\Pr_{\mathfrak{C}_{s_1\|\ldots\|s_{i-1}}}[s_i] > 0$. A sampling of l documents in succession from a channel is denoted by $s = s_1\| \ldots \| s_l \leftarrow \mathfrak{C}_h^l$ (shorthand notation for $s_1 \leftarrow \mathfrak{C}_h, s_2 \leftarrow \mathfrak{C}_{h\|s_1}, \ldots, s_l \leftarrow \mathfrak{C}_{h\|s_1\|\ldots\|s_{l-1}}$). A channel is called *always informative* if for every legal history $h \in \Sigma^*$, $H_\infty(\mathfrak{C}_h^l) = \Omega(l)$, where H_∞ is the min-entropy function. $\Diamond$

A channel can be modeled either as an oracle or as an efficiently computable randomized function $\mathsf{Channel}(h; r)$ (where r denotes the random coins). While the latter is a stronger assumption on the channel, [66] shows it to be necessary for secure steganography. Efficiently computable channels also enable broadcast steganographic constructions with stronger security guarantees as we show in Section 4.5.

Definition 2.4.17 (Steganography Setting): A steganography scheme, associated with a message space $\mathcal{MSP}$, and a channel $\mathfrak{C}_h$ on a set of documents Σ, is a tuple of algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encode}, \mathsf{Decode})$ defined as follows.

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$: KeyGen takes the security parameter 1^λ as input and outputs a public/secret key pair (pk, sk).

$s \leftarrow \mathsf{Encode}(pk, h, m)$: Encode takes a public key pk, a channel history $h \in \Sigma^*$, and a message $m \in \mathcal{MSP}$ as inputs and outputs a stegotext $s \in \Sigma^*$ from the support of $\mathfrak{C}_h^l$ for some $l = poly(|m|)$.

$m/\bot := \mathsf{Decode}(sk, h, s)$: Given a secret key sk, a channel history $h \in \Sigma^*$, and a stegotext $s \in \Sigma^*$, Decode either outputs a message $m \in \mathcal{MSP}$ or the failure symbol $\bot$. We assume that Decode is deterministic.

Correctness. For every $m \in \mathcal{MSP}$ and legal channel history $h \in \Sigma^*$, if (pk, sk) is output by $\mathsf{KeyGen}(1^\lambda)$, then it must be the case that

$$\mathsf{Decode}(sk, h, \mathsf{Encode}(pk, h, m)) = m,$$

except with negligible probability in the security parameter λ. $\Diamond$

There are three models of security defined for steganography schemes. They are, in the strongest to weakest order, steganographic secrecy against adaptive chosen-covertext attacks (SS-IND-CCA), steganographic secrecy against publicly-detectable, replayable adaptive chosen-covertext attacks (SS-IND-PDR-CCA), and steganographic secrecy against adaptive chosen-hiddentext attacks (SS-IND-CHA). These models of security are defined as games played between a PPT challenger and an adversary. Below we define the SS-IND-CCA game. Next, we show how to modify the SS-IND-CCA game to obtain the SS-IND-PDR-CCA and SS-IND-CHA games.

Definition 2.4.18 (SS-IND-CCA Game): For a given steganography scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encode}, \mathsf{Decode})$, the SS-IND-CCA game, which is played between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, is defined as follows.

Setup: $\mathcal{C}$ runs $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and gives $\mathcal{A}$ the resulting public key pk, keeping the secret key sk to itself.

Phase 1: $\mathcal{A}$ adaptively issues queries of the following type.

> **Decoding query (h, s):** $\mathcal{A}$ issues a decoding query on a channel history $h \in \Sigma^*$ and a covertext $s \in \Sigma^*$. $\mathcal{C}$ sends the result of $\mathsf{Decode}(sk, h, s)$ to $\mathcal{A}$.

Challenge: $\mathcal{A}$ gives $\mathcal{C}$ a message $m^* \in \mathcal{MSP}$ and a legal history $h \in \Sigma^*$. $\mathcal{C}$ picks a random bit $b^* \in \{0, 1\}$ and generates the challenge s^* depending on it as follows. If $b^* = 0$, then $\mathcal{C}$ encodes m^* into a stegotext s^*, *i.e.*, $s^* \leftarrow \mathsf{Encode}(pk, h, m^*)$. Otherwise, $\mathcal{C}$ sample s^* as a covertext of equal length, *i.e.*, $s^* \leftarrow_\$ \mathfrak{C}_h^{l^*}$ for $l^* = |\mathsf{Encode}(pk, h, m^*)|/\sigma$. At the end, $\mathcal{C}$ gives s^* to $\mathcal{A}$.

Phase 2: The interaction between $\mathcal{A}$ and $\mathcal{C}$ in this phase is similar to *Phase 1* with one restriction as given below.

Decoding query (h, s): $s \neq s^*$.

Guess: $\mathcal{A}$ outputs a guess $b \in \{0,1\}$ and wins if $b = b^*$.

The adversary $\mathcal{A}$ is called a SS-IND-CCA adversary and $\mathcal{A}$'s advantage is defined as

$$\mathsf{Adv}^{\text{SS-IND-CCA}}_{\mathcal{A},\Pi} := \left|\Pr[b = b^*] - \tfrac{1}{2}\right|,$$

where the probability is over the random coins used by $\mathcal{A}$ and $\mathcal{C}$. $\Diamond$

Definition 2.4.19 (SS-IND-CCA Security): A steganography scheme Π is (t, Q_D, ϵ)-SS-IND-CCA-secure if for any t-time SS-IND-CCA adversary making at most Q_D adaptive decoding queries, it must be the case that $\mathsf{Adv}^{\text{SS-IND-CCA}}_{\mathcal{A},\Pi} \leq \epsilon$. $\Diamond$

We obtain the SS-IND-PDR-CCA game by restricting the kind of decoding queries allowed in *Phase 2* of the SS-IND-CCA game. Specifically, the adversary now cannot issue any decoding query (h, s) such that $s \equiv_{pk} s^*$ for some SS compatible relation $\equiv_{pk}$, which is defined in Definition 2.4.20 below. The adversary $\mathcal{A}$'s advantage in the SS-IND-PDR-CCA game is defined as,

$$\mathsf{Adv}^{\text{SS-IND-PDR-CCA}}_{\mathcal{A},\Pi} := \left|\Pr[b = b^*] - \tfrac{1}{2}\right|,$$

where the probability is over the random coins used by $\mathcal{A}$ and $\mathcal{C}$.

Definition 2.4.20 (SS Compatible Relation): Let $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encode}, \mathsf{Decode})$ be a steganography scheme. Let (pk, sk) be a valid public/secret key pair generated by the KeyGen algorithm. A binary relation on stegotexts of Π induced by the public key pk is called a *SS compatible relation* (denoted by $\equiv_{pk}$) if for any two channel history/stegotext pairs $(h_1, s_1), (h_2, s_2)$ where s_1 and s_2 are encoded under the public key pk, the following requirements are satisfied.

1. If $s_1 \equiv_{pk} s_2$, then it must be the case that $\mathsf{Decode}(sk, h_1, s_1) = \mathsf{Decode}(sk, h_2, s_2)$ except with negligible probability in the security parameter λ.

2. There exists a PPT algorithm that only takes pk, (h_1, s_1), and (h_2, s_2) and determines whether $s_1 \equiv_{pk} s_2$.

Definition 2.4.21 (SS-IND-PDR-CCA Security): A steganography scheme Π is (t, Q_D, ϵ)-SS-IND-PDR-CCA-secure with respect to some SS compatible relation $\equiv_{pk}$ if for any t-time SS-IND-PDR-CCA adversary making at most Q_D adaptive decoding queries, it holds that $\mathsf{Adv}^{\text{SS-IND-PDR-CCA}}_{\mathcal{A},\Pi} \leq \epsilon$. $\diamondsuit$

The SS-IND-CHA game is defined similar to the SS-IND-CCA game, with the restriction that the adversary is not allowed to issue any decoding queries during *Phase 1* and *Phase 2*.

Definition 2.4.22 (SS-IND-CHA Security): A steganography scheme Π is (t, ϵ)-SS-IND-CHA-secure if Π is $(t, 0, \epsilon)$-SS-IND-CCA-secure. $\diamondsuit$

2.4.6 Multi-User Oblivious Random Access Machine

Multi-user oblivious random access machine (M-ORAM), which was originally proposed by Jinsheng *et al.* [72], is a protocol that allows a set of clients to obliviously share the storage at a cloud storage provider. A M-ORAM protocol is executed between four types of parties: a system initialization authority, a set of clients, a set of anonymizers, and a cloud storage provider $\mathcal{S}$. The SIA's job is to initialize the system by generating the required keys, and authorize the addition of new clients to the system. The set of anonymizers facilitate the communication between the clients and $\mathcal{S}$ while also preserving the oblivious data access and data secrecy guarantees. The formal definition of the setting of M-ORAM protocols is given below.

Definition 2.4.23 (M-ORAM Setting): A M-ORAM scheme, associated with an SIA, a set of clients $C = [1, N]$, a set of anonymizers $A = [1, M]$, a server $\mathcal{S}$, a position space $\mathcal{PSP}$, and a message space $\mathcal{MSP}$, is a tuple of algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Write}, \mathsf{Read})$ defined as follows.

$\boldsymbol{(\mathsf{MPK}, \mathsf{MSK}, ak_1, \ldots, ak_M, st_0) \leftarrow \mathsf{Setup}(1^\lambda, M)}$**:** Setup is a non-interactive algorithm executed by the SIA. This algorithm takes the security parameter 1^λ and the number of anonymizers in the system M as inputs and outputs the master public key MPK, the master secret key MSK, the anonymizers' secret keys $ak_1, \ldots, ak_M$, and the initial state of the server storage st_0. At the end of this algorithm, the SIA places MPK in a publicly accessible location, keeps MSK to herself, provides the anonymizers' secret keys to the corresponding anonymizers, and places st_0 in the storage provided by $\mathcal{S}$.

$\boldsymbol{ck_i \leftarrow \mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)}$**:** KeyGen is also a non-interactive algorithm executed by the SIA. This algorithm takes the master public key MPK, the master secret key MSK, and a client $i \in C$ as inputs and outputs a secret key ck_i for client i. At the end of this algorithm, the SIA gives ck_i to client i.

$\boldsymbol{(ak'_1, \ldots, ak'_M, st_{t+1}) \leftarrow \mathsf{Write}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_i, p, m, st_t)}$**:** Write is an interactive algorithm executed between a client $i \in C$, the M anonymizers, and $\mathcal{S}$. The anonymizers supply their secret keys $ak_1, \ldots, ak_M$. The client i supplies his secret key ck_i, a position $p \in \mathcal{PSP}$, and a message $m \in \mathcal{MSP}$. $\mathcal{S}$ provides the current server state st_t. At the end of this algorithm, the server state transforms from st_t to st_{t+1} and the old anonymizers' secret keys get replaced by the new secret keys $ak'_1, \ldots, ak'_M$.

$\boldsymbol{(m/\perp, ak'_1, \ldots, ak'_M, st_{t+1}) \leftarrow \mathsf{Read}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_i, p, st_t)}$**:** Read is also an interactive algorithm executed between a client $i \in C$, the M anonymizers, and $\mathcal{S}$. The anonymizers supply their secret keys $ak_1, \ldots, ak_M$. The client i

provides the secret key ck_i and a position p. $\mathcal{S}$ provides the current server state st_t. At the end of this algorithm, the client i learns a message m or the failure symbol $\perp$, the server state transforms from st_t to st_{t+1}, and the old anonymizers' secret keys get replaced by the new secret keys $ak'_1, \ldots, ak'_M$.

Correctness. Let MPK, $ak_1, \ldots, ak_M$, and st_t be a valid master public key, a sequence of M anonymizers' valid secret keys, and a valid server state, respectively. For every $h, i \in C$, $p \in \mathcal{PSP}$, and $m \in \mathcal{MSP}$, if ck_h, ck_i are the secret keys of the clients h, i, respectively, $\mathsf{Write}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_h, p, m, st_t) = (ak'_1, \ldots, ak'_M, st_{t+1})$, and $ak'_1, \ldots, ak'_M$ and st_{t+1} transform into $\overline{ak}'_1, \ldots, \overline{ak}'_M$ and $\overline{st}_{t+1}$ after zero or more executions of Read and Write algorithms that do not modify the message at position p, then $\mathsf{Read}(\overline{ak}'_1, \ldots, \overline{ak}'_M, ck_i, p, \overline{st}_{t+1})$ yields m, except with negligible probability in the security parameter λ. $\Diamond$

The formal definition of security of a M-ORAM protocol is defined in an *honest-but-curious* (OBC) adversarial model, where the adversary is allowed to gain as much information as he can with the requirement that he follow the M-ORAM protocol as specified in Definition 2.4.23. Similar to the security models presented earlier in this section, this M-ORAM security model is also defined as a game, termed M-ORAM-IND-OBC. In this game, the challenger simulates the entire M-ORAM protocol while giving the adversary read access to the private states of the cloud storage provider, the revoked clients, and the compromised anonymizers.

Definition 2.4.24 (M-ORAM-IND-OBC Game): For a given multi-user oblivious RAM scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Write}, \mathsf{Read})$, the M-ORAM-IND-OBC game, played between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, is defined as follows.

Setup: $\mathcal{C}$ runs $(\mathsf{MPK}, \mathsf{MSK}, ak_1, \ldots, ak_M, st_0) \leftarrow \mathsf{Setup}(1^\lambda, N)$, gives $\mathcal{A}$ the resulting master public key MPK and the initial server state st_0, and keeps the master secret key MSK and the anonymizer keys $ak_1, \ldots, ak_M$ to herself. $\mathcal{C}$ also initializes the sets of revoked clients R_C and compromised anonymizers R_A as empty sets.

Phase 1: $\mathcal{A}$ adaptively issues queries of the following types.

Client secret-key query $\boldsymbol{i}$: $\mathcal{A}$ requests the secret key of a client $i \in C$. $\mathcal{C}$ runs $ck_i \leftarrow \mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)$, adds i to R_C, and gives ck_i to $\mathcal{A}$.

Anonymizer secret-key query $\boldsymbol{j}$: $\mathcal{A}$ requests the secret key of an anonymizer $j \in A$. $\mathcal{C}$ adds j to R_A and gives ak_j, the most recent secret key of the anonymizer j, to $\mathcal{A}$. It is required that at least one anonymizer remain uncompromised for the duration of the M-ORAM-IND-OBC game.

Revoked client write query $(\boldsymbol{i}, \boldsymbol{p}, \boldsymbol{m})$: $\mathcal{A}$ inquires $\mathcal{C}$ to execute the Write algorithm on behalf of a revoked client $i \in R_C$ with a position $p \in \mathcal{PSP}$, and a message $m \in \mathcal{MSP}$ as inputs. Then, using the secret key of client i, the most recent anonymizer keys, and the server state, $\mathcal{C}$ simulates the interactive algorithm $\mathsf{Write}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_i, p, m, st_t)$ and gives the *adversarial view*[1] of this execution of Write algorithm to $\mathcal{A}$.

Revoked client read query $(\boldsymbol{i}, \boldsymbol{p})$: $\mathcal{A}$ asks $\mathcal{C}$ to run the Read algorithm on a position $p \in \mathcal{PSP}$ on behalf of a revoked client $i \in R_C$. $\mathcal{C}$ simulates $\mathsf{Read}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_i, p, st_t)$ using the client i's secret key, the most recent anonymier keys, and the server state; and gives the adversarial view of this execution of Read to $\mathcal{A}$.

Honest client write query $(\boldsymbol{i}, \boldsymbol{m})$: $\mathcal{A}$ asks $\mathcal{C}$ to run the Write algorithm for an honest client $i \in C \setminus R_C$ with a message $m \in \mathcal{MSP}$. $\mathcal{C}$, however, does not allow $\mathcal{A}$ to provide a position for this query. Instead, $\mathcal{C}$ picks

[1]The adversarial view includes the list of all the publicly visible requests exchanged among the client i, the anonymizers, and $\mathcal{S}$, changes made to the server state, as well as the private states and the secret keys of the entities controlled by $\mathcal{A}$ (such as the client i and the compromised anonymizers) during an episode of an interactive execution of Write or Read in the chronological order.

$p \leftarrow\!\$\ \mathcal{PSP}$ uniformly at random and assigns to it a sequential identifier id. After running $\mathsf{Write}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_i, p, m, st_t)$ using the client i's secret key, the most recent anonymizer keys, and the server state, $\mathcal{C}$ gives the adversarial view of this execution of Write and id to $\mathcal{A}$.

Honest client read query (i, id): $\mathcal{A}$ inquires $\mathcal{C}$ to run the Read algorithm for an honest client $i \in C \setminus R_C$ on the position associated with the identifier id. $\mathcal{C}$ looks up the position p that she associated with id during a previous honest client write query, simulates $\mathsf{Read}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_i, p, st_t)$ using the client i's secret key, the most recent anonymier keys, and the server state; and provides the adversarial view of this execution of Read to $\mathcal{A}$.

Pre-Challenge: $\mathcal{A}$ gives $\mathcal{C}$ two messages $m_0^*, m_1^* \in \mathcal{MSP}$, and a client $i^* \in C \setminus R_C$. $\mathcal{C}$ picks two positions $p_0^*, p_1^* \leftarrow\!\$\ \mathcal{PSP}$ and assigns sequential identifiers id_0^*, id_1^* to them, respectively. Next, it runs $\mathsf{Write}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_{i^*}, p_0^*, m_0^*, st_t)$ and $\mathsf{Write}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_{i^*}, , p_1^*, m_1^*, st_{t+1})$ using the client i^*'s secret key, the most recent anonymizer keys, and the server state; and gives the adversarial views of these executions of Write and the position identifiers id_0^*, id_1^* to $\mathcal{A}$.

Phase 2: This phase is similar to *Phase 1* with the exception that $\mathcal{A}$ is not allowed to corrupt the client i^*. Note that $\mathcal{A}$ may submit honest client read queries on the position identifiers id_0^*, id_1^*.

Challenge: $\mathcal{C}$ picks a random bit $b^* \in \{0, 1\}$ and simulates the execution of $\mathsf{Read}($ $\mathsf{MPK}, ak_1, \ldots, ak_M, ck_{i^*}, p_{b^*}^*, st_t)$ using the client i^*'s secret key, the most recent anonymier keys, and the server state. $\mathcal{C}$ also assigns two new sequential position identifiers id_2^*, id_3^* to p_b^*, p_{1-b}^*, respectively. Finally, $\mathcal{C}$ gives the adversarial view of this execution of Read and the two position identifiers id_2^*, id_3^* to the adversary.

Phase 3: This phase is similar to *Phase 2* with the exception that $\mathcal{A}$ is not allowed to submit any honest client read queries on the position identifiers id_0^*, id_1^*. However,

$\mathcal{A}$ is allowed to submit such queries on the position identifiers id_2^*, id_3^*

Guess: $\mathcal{A}$ outputs a guess $b \in \{0,1\}$ and wins if $b = b^*$.

$\mathcal{A}$ is called an M-ORAM-IND-OBC adversary and $\mathcal{A}$'s advantage in winning the above game is defined as

$$\mathsf{Adv}_{\mathcal{A},\Pi}^{\text{M-ORAM-IND-OBC}} := \left|\Pr[b = b^*] - \tfrac{1}{2}\right|,$$

where the probability is over the random coins used by $\mathcal{A}$ and $\mathcal{C}$. $\Diamond$

Definition 2.4.25 (M-ORAM-IND-OBC Security): An M-ORAM scheme Π is $(t, Q_C, Q_A, Q_D, \epsilon)$-M-ORAM-IND-OBC-secure if for any t-time M-ORAM-IND-OBC adversary making at most Q_C, Q_A, and Q_D adaptive client secret-key, anonymizer secret-key, and data access queries, respectively, we have $\mathsf{Adv}_{\mathcal{A},\Pi}^{\text{M-ORAM-IND-OBC}} \leq \epsilon$. $\Diamond$

2.5 Subset Cover Framework

The *subset cover* (SC) framework proposed by Naor *et al.* [89] is a system that abstracts a variety of broadcast encryption schemes in the private-key setting where only the Center can transmit encrypted messages. A broadcast encryption scheme belonging to the SC framework defines a family of subsets Fam of the universe of users $U = [1, N]$ in the system. The Center assigns to each subset $F_j \in \mathsf{Fam}$ a long-lived private key k_j. When generating the private key of a user $i \in U$, the Center runs the following steps.

1. Find the set of all the subsets $\mathsf{Usr} \subseteq \mathsf{Fam}$ that user i is a member of.

2. Give to user i the long-lived private keys associated with each subset in Usr.

When the Center wants to broadcast a message m to a subset of users $S \subseteq U$, it executes the following steps.

1. Generate a short-lived session key $\widehat{k}$.
2. Encrypt m under $\widehat{k}$ using a semantically secure private-key encryption protocol.
3. Find a set of disjoint subsets $\mathsf{Cov} \subseteq \mathsf{Fam}$ that contains or "covers" all the users belonging to S.
4. Encrypt $\widehat{k}$ using each long-lived private key corresponding to the subsets in Cov.
5. Broadcast the encryption of m under $\widehat{k}$ and the encryptions of $\widehat{k}$ to all the users in the system.

Upon receiving a broadcast ciphertext, a user can successfully decrypt and obtain m if and only if that user is part of the authorized set (*i.e.*, the user possesses a long-lived private key corresponding to some subset in Cov).

The authors in [89] also presented two concrete revocation schemes, namely the *complete subtree* (CS) method and the *subset difference* (SD) method. In the CS method, which is the simplest of the two, the ciphertext length is $O\Big(r \log\Big(\frac{N}{r}\Big)\Big)$ and the private key length at a receiver is $O(\log N)$, where r is the number of revoked users in a broadcast ciphertext. In the SD method, the one with more involved computations, the ciphertext length reduces to $O(r)$ while the private key length increases to $O\Big(\log^2 N\Big)$. Another crucial difference between the two schemes is that the assignment of the long-lived private keys in the former is information-theoretic, whereas in the latter it is computational. Below we provide a short description of the CS method, and we refer the reader to [89] for further details on the SD method.

2.5.1 Complete Subtree Method

In this scheme, the N users are represented as the leaves of a perfect binary tree $\mathcal{T}$ and the collection of subsets Fam contains all possible subtrees of $\mathcal{T}$. In case N is not a power of 2, some dummy users are added to the system. During the key generation phase, every subtree in Fam is assigned a long-lived private key which is also made available to all the users belonging to that subtree. Since every user is a member of all the subtrees rooted at each node in the path from the root of $\mathcal{T}$ down to the leaf corresponding to that user, the secret key length at a user is $O(\log N)$. The ciphertext length becomes $O\Big(r \log\Big(\frac{N}{r}\Big)\Big)$ due to the fact that it requires on average a logarithmic number of subtrees to revoke r users (see [89] for a formal analysis).

2.5.2 Extension of the Subset Cover Framework to the Public-Key Setting

The original subset cover framework was defined only for the private-key setting. In [40], Dodis and Fazio extended the SC framework to the public-key setting by combining a novel assignment of hierarchical identifiers (HIDs) to the nodes in $\mathcal{T}$ with (hierarchical) identity-based encryption. For completeness, we only explain below the extension of the CS method. We refer the reader to [40] for the details regarding the extension of the SD method.

The assignment of HIDs to the nodes in $\mathcal{T}$ goes as follows. First, the root of $\mathcal{T}$ is assigned a special identifier denoted by Root. Next, each edge e of $\mathcal{T}$ is assigned the identifier $\mathsf{ID}_e \in \{0,1\}$ depending on whether the edge connects to the left child or to the right child. Then, the hierarchical identifier HID_v of any node v can be computed by concatenating all the identifiers starting from the root of $\mathcal{T}$ down to v (*i.e.*, $\mathsf{HID}_v := \mathsf{Root}\|\mathsf{ID}_{e_1}\|\dots\|\mathsf{ID}_{e_{\log N}}$). It is important to note that any prefix of HID_v represents a valid HID of a parent node of v.

Once the HIDs of the nodes are assigned, the Dodis and Fazio extends the CS method to the public-key setting by utilizing an IBE scheme as follows. First, each of the long-lived private keys corresponding to the subtrees in Fam of the original CS method now gets replaced by a public/private key pair. For a given subtree t in Fam the public key becomes the hierarchical identifier $\mathsf{HID}_{t_{\mathsf{root}}}$, where t_{root} denotes the root of t; and the private key becomes the IBE secret key corresponding to $\mathsf{HID}_{t_{\mathsf{root}}}$ when viewed as a bit-string. The key generation algorithm of the public-key CS method, when executed on behalf of a user i, goes as follows.

1. Find the set of all the subtrees $\mathsf{Usr} \subseteq \mathsf{Fam}$ that user i (when viewed as a leaf of $\mathcal{T}$) is a member of.

2. Extract the IBE secret keys corresponding to the HIDs of the root nodes of each subtree belonging to Usr.

3. Give those IBE secret keys to user i.

Since the structure of the $\mathcal{T}$ and the assignment of HIDs to the nodes of $\mathcal{T}$ are publicly known to all the users, any user in the system can be a sender as well as a receiver. When a user want to broadcast a message to a subset of users $S \subseteq U$, it executes the following steps.

1. Generate a short-lived session key $\widehat{k}$.

2. Encrypt m under $\widehat{k}$ using a semantically secure private-key encryption protocol.

3. Find a set of disjoint subtrees $\mathsf{Cov} \subseteq \mathsf{Fam}$ covering all the users in S.

4. Using the IBE scheme, encrypt the session key $\widehat{k}$ under each of the HIDs associated with the root nodes of the subtrees in Cov.

5. Broadcast the encryption of the message m under $\widehat{k}$ and the IBE encryptions of $\widehat{k}$ to all the users in the system.

Upon receiving a broadcast ciphertext, a user can decrypt if and only if that user possesses an IBE secret key corresponding any of the valid root node HIDs associated with the cover set of that ciphertext. In the public-key setting, the Center becomes the system initialization authority that establishes the broadcast encryption system and provides each user with the required IBE keys.

Chapter 3

Outsider-Anonymous Broadcast Encryption

3.1 Introduction

The intrinsic access control capabilities of BE schemes make them a useful tool for many natural applications, spanning from protecting copyrighted content distributed as stored media [1], to managing digital subscriptions to satellite TV, to controlling access in encrypted file systems [19]. Thanks to its wide variety of applications, BE has received a lot of attention from the crypto research community in recent years (*e.g.*, [18, 22, 23, 40–42, 50, 54, 64, 89, 118]). The quest in these works, however, has been for ever more efficient solutions in terms of sender-oriented properties such as less encryption/decryption running time and more compact broadcast ciphertexts and key storage. And, in these respects, the constructions proposed in [18, 54] can be considered as being nearly optimal. Little attention, instead, has been devoted to the exploration of refined BE security models that accurately account for the requirements inherent in multi-recipient communication. More specifically, the focus has been on providing attractive solutions for senders, while overlooking the security and privacy concerns of the receivers.

For instance, the formal definition of broadcast encryption explicitly requires that whenever some digital content is encrypted and sent in broadcast, information about the set of authorized receivers is necessary to decrypt it correctly. Therefore, the set of authorized receivers is transmitted as part of the ciphertext. This in particular implies that an eavesdropper, even if unable to recover the message, can still easily discover the identities of the actual receivers of the content. One way to address the privacy implications that result from explicitly specifying the set of authorized receivers in the broadcast is to use ephemeral IDs and to keep secret the table that associates such IDs with the actual receivers. This simple solution, however, would at best result in a pseudonym system, in which it is still possible to link pseudonyms across transmissions and determine whether the same entity is an authorized receiver for two different broadcasts. Consequently, the state of the art BE schemes are inherently incapable of preserving *any* notion of receiver anonymity.

In certain broadcast applications, protecting the privacy of the receivers is just as important as preserving the confidentiality of the broadcast messages. For example, suppose a satellite TV provider employs a BE scheme to securely broadcast sensitive information over a channel to its subscribers. Now, if the BE scheme does not provide any privacy guarantees of the users, a subscriber decrypting the channel using his secret key might also learn who else has subscribed for that channel. This is a serious violation of the privacy of the subscribers.

Anonymous Broadcast Encryption. The first work in the cryptographic literature to consider recipient privacy in broadcast encryption was put forth by Barth *et al.* [10]. Motivated by the privacy requirements in encrypted file systems, the authors therein introduced the notion of *private* broadcast encryption, which later came to be known as *anonymous broadcast encryption*, that aims to prevent the leakage of the identities of the receivers. As a proof-of-concept, they also suggested two generic public-key constructions, one with linear decryption time in the number of legitimate recipients and another with constant decryption time, that do not leak any information about the set of authorized receivers of a broadcast ciphertext and are secure in the standard model and in the random oracle model, respectively. In [83], Libert *et al.* suggested a technique to prove the security of a variant of the second construction of [10] without reliance on random oracles, thus attaining an AnoBE construction with efficient decryption in the standard model.

Krzywiecki *et al.* presented a private public-key broadcast encryption scheme with communication complexity proportional to the number of revoked users [77]. The security analysis of the proposed solution is rather informal, however, so the security guarantees are at best heuristic.

In [93], Yu *et al.* presented the first *secret-key multicast* scheme with membership anonymity and communication complexity independent of the number of receivers. The proposed scheme not only hides the *identities* of the receives, but also *the number* of users allowed to receive the content. A shortcoming is that only a single user can be revoked for each broadcast.

A promising research line toward practical anonymous broadcast encryption was initiated by Jarecki and Liu [71]. The authors propose the first construction of an efficient unlinkable secret handshake scheme, which is an authenticated key exchange protocol providing *affiliation/policy hiding* (*i.e.*, the transmission hides the affiliation and the identities of all parties) and *unlinkability* (*i.e.*, it is impossible to link any two instances of the secret handshake protocol). The proposed construction can be seen as a *stateful* version of a public-key broadcast encryption scheme, with the additional property of protecting the receivers' identities. Statefulness, however, implies that the key used to encrypt the broadcasts changes for each transmission, and receivers need to keep track of the changes to be able to recover the content. An interesting trait of the of construction of [71] is that it trades some degree of anonymity for better efficiency: while the receiver's identities are hidden from outsiders, the scheme still allows authorized users to learn information about other members of the receiver set.

A major drawback of the state of the art AnoBE constructions mentioned above is that their ciphertexts have linear length in the number of authorized receivers. A linear ciphertext length is a highly undesirable property in any large-scale broadcast application. Furthermore, as Kiayias and Samari recently showed in [76], this drawback is unfortunately unavoidable. In [76], the authors presented the lower bounds on the ciphertext length of AnoBE schemes and showed that fully anonymous broadcast encryption schemes with atomic ciphertexts (*e.g.*, the schemes of [10, 83]) must have $\Omega(s\,\lambda)$ ciphertext length, where s is the number of authorized receivers and λ is the security parameter. This lower bound highlights the cost of achieving full anonymity of the receivers in the setting of broadcast encryption.

3.2 Contributions

There are yet other broadcast applications where the anonymity of the authorized receivers must be protected only from the outsiders (or the unauthorized users). As a simple example, imagine that a group of scientists working for a top secret government project wants to broadcast documents among themselves. Since their identities and the documents they share are equally sensitive, they decide to employ an AnoBE scheme for the transmissions. Now, since the scientists already know one another, the full anonymity provided by AnoBE is not really necessary. What they really need is a secure broadcast encryption scheme that prevents the leakage of their identities to the outsiders. Also, as shown in [76], full anonymity comes at a cost of ciphertext length being linear in the number of authorized receivers.

Outsider-Anonymous Broadcast Encryption. We formalize the above notion of anonymity in this chapter under the term *outsider-anonymous broadcast encryption* (oABE).[1] We identify that the notion of outsider anonymity lies in between the complete lack of anonymity that characterizes traditional broadcast encryption schemes, and the full anonymity provided by AnoBE schemes. Taking advantage of this relaxation of anonymity, we also present modular oABE constructions whose ciphertext lengths are sublinear in the number of legitimate receivers. It should be noted that oABE is the first broadcast encryption scheme to achieve sublinear ciphertext lengths while also guaranteeing a useful degree of anonymity for the authorized receivers. In summary, our contributions are as follows.

1. First, we formally present the definition of Outsider-Anonymous Broadcast Encryption. Compared to the definition of regular broadcast encryption, our definition does not require the set of receivers to be an input to the decryption algorithm. As we noticed, this modification to the decryption algorithm is a necessary step toward any notion of anonymity for broadcast encryption.

2. Second, we put forth two security models of oABE: security with respect to chosen plaintext attack and security with respect to chosen ciphertext attack. These security models are presented as games played between a probabilistic polynomial time adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. We say that a particular oABE construction is secure with respect to one of these games if for all probabilistic polynomial time adversaries, the advantage of winning the game is negligible in the security parameter of the construction.

3. Then, we present several secure oABE constructions with different optimization criterion. Table 3.1 summarizes the parameters of these constructions. This table also compares our constructions to the state of the art broadcast encryption schemes with anonymity guarantees. Notice that all of our oABE constructions attain sublinear ciphertext length in the number of receivers by trading some

[1]This result has also been published at the 15th International Conference on Practice and Theory in Public Key Cryptography—PKC 2012 [48].

Table 3.1: Comparison of the parameters of public-key (outsider-)anonymous broadcast encryption schemes. The second half of the table shows the schemes with a tagging mechanism to allow one decryption attempt from the underlying crptosystem per ciphertext. N denotes the total number of users in the system. r denotes the number of revoked users of a ciphertext.

Scheme	**MPK Length**	***sk* Length**	***c* Length**	**Security Model**	**Anonymity**	**Decrypt Attempts**
AnoBE Cons. 1 of [10]	$O(N)$	$O(1)$	$O(N-r)$	CCA, Static $\mathcal{A}$	Full	$(N-r)/2$
AnoBE Cons. 1 of [83]	$O(N)$	$O(1)$	$O(N)$	CCA, Adaptive $\mathcal{A}$	Full	1
AnoBE Cons. 2 of [83]	$O(N)$	$O(1)$	$O(N-r)$	CCA, Adaptive $\mathcal{A}$	Full	$(N-r)/2$
AnoBE Cons. 3 of [83]	$O(N)$	$O(1)$	$O(N-r)$	CCA, Adaptive $\mathcal{A}$	Full	$(N-r)/2$
oABE Cons. of Sect. 3.4.1	$O(1)$	$O(\log N)$	$O\big(r\log(\frac{N}{r})\big)$	CPA, Adaptive $\mathcal{A}$	Outsider	$\big(\lfloor r\log(\frac{N}{r})\rfloor \log N\big)/2$
oABE Cons. of Sect. 3.4.2	$O(1)$	$O(\log N)$	$O\big(r\log(\frac{N}{r})\big)$	CCA, Adaptive $\mathcal{A}$	Outsider	$\big(\lfloor r\log(\frac{N}{r})\rfloor \log N\big)/2$
AnoBE Cons. 2 of [10]	$O(N)$	$O(1)$	$O(N-r)$	CCA, Static $\mathcal{A}$, ROM	Full	1
AnoBE Cons. 4 of [83]	$O(N)$	$O(1)$	$O(N-r)$	CCA, Adaptive $\mathcal{A}$	Full	1
oABE Cons. of Sect. 3.4.3	$O(N)$	$O(\log N)$	$O\big(r\log(\frac{N}{r})\big)$	CCA, Adaptive $\mathcal{A}$, ROM	Outsider	1
oABE Cons. of Sect. 3.4.4	$O(N)$	$O(\log N)$	$O\big(r\log(\frac{N}{r})\big)$	CCA, Adaptive $\mathcal{A}$	Outsider	1
oABE Cons. of Sect. 3.4.5	$O(N\log N)$	$O(N)$	$O(r)$	CCA, Adaptive $\mathcal{A}$	Outsider	1

degree of anonymity. As we mentioned earlier, in certain applications of broadcast communication, this level of anonymity provided by our oABE constructions do suffice. In addition to sublinear ciphertext length, some of our constructions also enjoy enhanced decryption, where for a given oABE ciphertext, the decryption algorithm executes a single decryption operation of the underlying cryptosystem.

4. Finally, we prove that our constructions are secure with respect to their corresponding security models. At the heart of our proofs is a set of reduction arguments comprising multiple intermediate hybrid games. Specifically, we show how to use any adversary that can distinguish between any pair of two consecutive games to break the security of one of the underlying cryptosystems of our oABE constructions or falsify one of the Diffie-Hellman assumptions that we presented earlier in Section 2.2.

Organization. The setting and the security models of outsider-anonymous broadcast encryption is introduced in Section 3.3. In Section 3.4, we first present a generic oABE construction in the standard model that achieve outsider anonymity under adaptive corruptions in the chosen-plaintext (Section 3.4.1) and chosen-ciphertext (Section 3.4.2) settings. Next, we describe a CCA-secure oABE construction with enhanced decryption that is secure under the gap Diffie-Hellman assumption in the random oracle model (Section 3.4.3), and also extend it to the standard model (Section 3.4.4), using the twin Diffie-Hellman-based techniques of [29]. In Section 3.4.5 we also present a variant of the scheme in Section 3.4.4 with even shorter ciphertexts, at a price on the other parameters, most notably user storage and decryption complexity. Finally, we outline an optimization for the private-key setting to accommodate storage-constrained systems and attain constant key storage at the Center, while maintaining efficient decryption and logarithmic storage at the receivers (Section 3.4.6).

3.3 Formal Model

3.3.1 Setting of oABE

The setting of outsider-anonymous broadcast encryption is analogous to that of broadcast encryption with one important distinction: the oABE decryption algorithm does not require the set of recipients as an input. We stress that is the starting point for providing any level of anonymity in a broadcast encryption scheme.

Definition 3.3.1 (oABE Setting): An outsider-anonymous broadcast encryption scheme, associated with a universe of users $U = [1, N]$, a message space $\mathcal{MSP}$, and a ciphertext space $\mathcal{CSP}$, is a tuple of algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ defined as follows.

$\boldsymbol{(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, N)}$**:** The Setup algorithm takes as input the security parameter 1^λ and the number of users in the system N. It outputs the master public key MPK and the master secret key MSK of the system.

$\boldsymbol{sk_i \leftarrow \mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)}$**:** The key generation algorithm KeyGen takes as input the master public key MPK, the master secret key MSK, and a user $i \in U$. It outputs the secret key sk_i of the user i.

$\boldsymbol{c \leftarrow \mathsf{Encrypt}(\mathsf{MPK}, S, m)}$**:** The $\mathsf{Encrypt}$ algorithm takes as input the master public key MPK, the set of receivers $S \subseteq U$, and a message $m \in \mathcal{MSP}$. It then outputs a ciphertext $c \in \mathcal{CSP}$.

$\boldsymbol{m/\perp := \mathsf{Decrypt}(\mathsf{MPK}, sk_i, c)}$**:** Given the master public key MPK, a secret key sk_i, and a ciphertext $c \in \mathcal{CSP}$, the $\mathsf{Decrypt}$ algorithm either outputs a message $m \in \mathcal{MSP}$ or the failure symbol $\perp$. We assume that $\mathsf{Decrypt}$ is deterministic.

Correctness. For every $S \subseteq U$, every $i \in S$, and every $m \in \mathcal{MSP}$, if sk_i is the secret key output by $\mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)$, then it must be the case that

$$\mathsf{Decrypt}(\mathsf{MPK}, sk_i, \mathsf{Encrypt}(\mathsf{MPK}, S, m)) = m,$$

except with negligible probability in the security parameter λ. $\Diamond$

3.3.2 Security of oABE

The degree of recipient-set anonymity captured in the security models of oABE, which we call *outsider anonymity*, lies between the complete lack of protection that characterizes traditional broadcast encryption schemes as introduced in [12, 49], and the full anonymity provided in schemes such as [10, 83]. In an oABE scheme, when the adversary receives a ciphertext of which she is not a legal recipient, she will be unable to learn anything about the identities of the legal recipients (let alone the contents of the ciphertext). Still, for those ciphertexts for which the adversary is in the authorized set of recipients, she might also learn the identities of some the other legal recipients. This seems a natural relaxation, since often the *contents* of the communication already reveals something about the recipient set. At the same time, our new intermediate definitions of security allow the constructions of more efficient anonymous broadcast encryption schemes; for example, in Section 3.4 we describe the first broadcast encryption schemes with sublinear ciphertexts that attain some meaningful recipient-set anonymity guarantees. Specifically, we define two models of security for oABE schemes, namely, chosen-ciphertext attack security (oABE-IND-CCA) and chosen-plaintext attack security (oABE-IND-CPA).

oABE-IND-CCA Security. This is the strongest notion of security of an oABE scheme, and it is related to the BE-IND-CCA model of security of BE schemes. Formally, we define the oABE-IND-CCA security model as a game played between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. For a given oABE scheme to be secure in this model of security, any $\mathcal{A}$'s advantage in winning the oABE-IND-CCA game must be negligible. The high-level idea of this game is that for any two sets of recipients $S_0, S_1 \in U$, any $\mathcal{A}$ should not be able to distinguish between a ciphertext intended for the recipient set S_0 and a ciphertext intended for the recipient set S_1 given the fact that the $\mathcal{A}$ does not possess the secret key of any user in $S_0 \cup S_1$. We require the two sets S_0, S_1 be the same size in order to avoid trivial ciphertext length-based attacks. The formal definition of the oABE-IND-CCA model of security is given below.

Definition 3.3.2 (oABE-IND-CCA Game): The oABE-IND-CCA game defined for an oABE scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$, a PPT adversary $\mathcal{A}$, and a challenger $\mathcal{C}$ is as follows.

Setup: $\mathcal{C}$ runs $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, N)$ and gives $\mathcal{A}$ the resulting master public key MPK, keeping the master secret key MSK to itself. $\mathcal{C}$ also initializes the set of revoked users R_U to be empty.

Phase 1: $\mathcal{A}$ adaptively issues queries of the following types.

Secret-key query $\boldsymbol{i}$: $\mathcal{A}$ requests the secret key of the user $i \in U$. $\mathcal{C}$ runs $sk_i \leftarrow \mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)$ to generate the secret key sk_i of the user i, adds i to R_U, and sends sk_i to $\mathcal{A}$.

Decryption query $(\boldsymbol{i}, \boldsymbol{c})$: $\mathcal{A}$ issues a decryption query where $i \in U$ and $c \in \mathcal{CSP}$. First, $\mathcal{C}$ runs $sk_i \leftarrow \mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)$ to generate the secret key sk_i of user i. Then, it runs $\mathsf{Decrypt}(\mathsf{MPK}, sk_i, c)$ and gives the output to $\mathcal{A}$.

Challenge: $\mathcal{A}$ gives $\mathcal{C}$ two equal length messages $m_0^*, m_1^* \in \mathcal{MSP}$ and two equal length sets of user identities $S_0^*, S_1^* \subseteq U$ with the restriction that $R_U \cap (S_0^* \cup S_1^*) = \emptyset$. $\mathcal{C}$ picks a random bit $b^* \in \{0, 1\}$, runs $c^* \leftarrow \mathsf{Encrypt}(\mathsf{MPK}, S_{b^*}^*, m_{b^*}^*)$, and returns c^* to $\mathcal{A}$.

Phase 2: The interaction between $\mathcal{A}$ and $\mathcal{C}$ in this phase is similar to *Phase 1* with two restrictions as given below.

Secret-key query $\boldsymbol{i}$: $i \notin S_0^* \cup S_1^*$.

Decryption query $(\boldsymbol{i}, \boldsymbol{c})$: If $i \in S_0^* \cup S_1^*$, then $c \neq c^*$.

Guess: $\mathcal{A}$ outputs a guess $b \in \{0,1\}$ and wins if $b = b^*$.

We refer to such an adversary $\mathcal{A}$ as an oABE-IND-CCA adversary. The advantage of $\mathcal{A}$ winning the above game is defined as,

$$\mathsf{Adv}_{\mathcal{A},\Pi}^{\text{oABE-IND-CCA}} := \left|\Pr[b = b^*] - \tfrac{1}{2}\right|,$$

where the probability is over the random bits used by $\mathcal{A}$ and $\mathcal{C}$. $\diamondsuit$

Definition 3.3.3 (oABE-IND-CCA Security): An oABE scheme Π is (t, Q_U, Q_D, ϵ)-oABE-IND-CCA-secure if for any t-time oABE-IND-CCA adversary making at most Q_U adaptive secret-key queries and at most Q_D chosen decryption queries, we have that $\mathsf{Adv}_{\mathcal{A},\Pi}^{\text{oABE-IND-CCA}} \leq \epsilon$. $\diamondsuit$

oABE-IND-CPA Security. This model of security is defined similar to the oABE-IND-CCA game with the restriction that the adversary is not allowed to issue any decryption queries during *Phase 1* and *Phase 2*. The adversary is still allowed to issue secret-key queries. Also note that oABE-IND-CPA security model is analogous to the BE-IND-CPA security model of BE schemes.

Definition 3.3.4 (oABE-IND-CPA Security): An oABE scheme Π is (t, Q_U, ϵ)-oABE-IND-CPA-secure if Π is $(t, Q_U, 0, \epsilon)$-oABE-IND-CCA-secure. $\diamondsuit$

Remark 3.3.5. Any of our definitions of security of outsider-anonymous broadcast encryption schemes given above can be easily transformed to a corresponding definition of security of a fully anonymous broadcast encryption scheme by changing the restriction in the *Challenge* step, which is currently $R_U \cap (S_0^* \cup S_1^*) = \emptyset$, to $R_U \cap (S_0^* \triangle S_1^*) = \emptyset$.[2]

3.4 Constructions

We now present our constructions of outsider-anonymous broadcast encryption schemes. In a nutshell, the key point of our constructions is to combine an anonymized version of the public-key extension by Dodis and Fazio [40] of the complete subtree method of the subset cover framework by Naor *et al.* [89] with a fully secure weakly robust AIBE scheme such as [51]. Reviews of the SC framework and the CS method are given in Section 2.5. Notice that our approach can be seen as a framework for achieving an oABE scheme by using any weakly robust AIBE scheme as an underlying primitive.

The ciphertext length in all constructions is $O\Big(r \log\Big(\frac{N}{r}\Big)\Big)$ times the ciphertext length of the underlying AIBE scheme, and the user secret key length is $O(\log N)$ times the user secret key length of the underlying AIBE scheme, where r is the number of revoked users and N is the total number of users in the system.

We start with two generic oABE constructions: an oABE-IND-CPA-secure construction in Section 3.4.1 and an oABE-IND-CCA-secure construction in Section 3.4.2. A considerable limitation with each of these constructions is that on average, the $\mathsf{Decrypt}$ algorithm attempts $\Big(\Big\lfloor r \log\Big(\frac{N}{r}\Big)\Big\rfloor \log N\Big)/2$ decryption operations of the underlying AIBE scheme for each decrypted oABE ciphertext. In Section 3.4.3, we present an enhanced oABE-IND-CCA-secure construction in which for a given oABE ciphertext, the $\mathsf{Decrypt}$ algorithm executes a single AIBE decryption operation. A drawback of this construction is that its security can only be proven in the random oracle model. In Section 3.4.4, we present another enhanced oABE-IND-CCA-secure construction whose security can be proven in the standard model. In Section 3.4.5 we present a variant of the scheme in Section 3.4.4 attaining even shorter ciphertexts, at

[2]For any two sets S_0, S_1, their symmetric difference is denoted by $S_0 \triangle S_1$.

a price on the other parameters, most notably, user storage and decryption complexity. Finally, in Section 3.4.6, we outline an optimization for the private-key setting to attain constant key storage at the Center, while maintaining efficient decryption and logarithmic storage at the receivers.

For the simplicity of exposition, our constructions encrypt the actual message m. The ciphertext length could be further reduced by using a hybrid encryption where m is encrypted using a private-key encryption algorithm with a secret key k, and then k is encrypted using the oABE scheme.

In all constructions, $\mathcal{T}$ denotes the binary tree of N users in the system with respect to the CS method. For simplicity, we assume that $N = 2^n$.

3.4.1 A Generic oABE-IND-CPA-Secure Public-Key Construction

Given a weakly robust AIBE-IND-CPA-secure anonymous identity-based encryption scheme $\Pi' = (\mathsf{Setup}', \mathsf{Extract}', \mathsf{Encrypt}', \mathsf{Decrypt}')$, we construct an oABE-IND-CPA-secure outsider-anonymous broadcast encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as shown below.

$\mathsf{Setup}(1^\lambda, N)$: Obtain $(\mathsf{MPK}', \mathsf{MSK}') \leftarrow \mathsf{Setup}'(1^\lambda)$. Output MPK and MSK as

$$\mathsf{MPK} \coloneqq (\mathsf{MPK}', N) \quad \mathsf{MSK} \coloneqq \mathsf{MSK}'.$$

$\mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)$: Let $\mathsf{HID}_i \coloneqq (\mathsf{Root}, \mathsf{ID}_1, \ldots, \mathsf{ID}_n)$ be the hierarchical identifier associated with user i in the binary tree $\mathcal{T}$. For $k \coloneqq 1$ to $n+1$, compute $sk_{i,k} \leftarrow \mathsf{Extract}'(\mathsf{MPK}', \mathsf{MSK}', \mathsf{HID}_{i|k})$. Output the secret key sk_i of user i as

$$sk_i \coloneqq (sk_{i,1}, \ldots, sk_{i,n+1}) .$$

$\mathsf{Encrypt}(\mathsf{MPK}, S, m)$: Let Cover be the family of subtrees covering the set of receivers S according to the CS method. For each subtree T_j in Cover, let HID_j be the hierarchical identifier associated with the root of T_j. Let $l \coloneqq |\mathsf{Cover}|$, $r \coloneqq N - |S|$ and $L \coloneqq \left\lfloor r \log\left(\frac{N}{r}\right)\right\rfloor$. For $1 \leq j \leq l$, compute $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}_j, m)$. Set $\widetilde{m} \leftarrow_{\$} \{0,1\}^{|m|}$. For $l+1 \leq j \leq L$, compute $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$, where $\mathtt{dummy}$ is a special identifier used to obtain padding ciphertext components. Output the ciphertext c as

$$c \coloneqq \left(c_{\pi(1)}, \ldots, c_{\pi(L)}\right),$$

where $\pi : \{1, \ldots, L\} \to \{1, \ldots, L\}$ is a random permutation.

$\mathsf{Decrypt}(\mathsf{MPK}, sk_i, c)$: Parse the secret key sk_i as the tuple $(sk_{i,1}, \ldots, sk_{i,n+1})$ and the ciphertext c as the tuple $(c_1, \ldots, c_L)$.

```
1. For $k \coloneqq 1$ to $n+1$,
    a. For $j \coloneqq 1$ to $L$,
        i. Compute $m \coloneqq \mathsf{Decrypt}'(\mathsf{MPK}', sk_{i,k}, c_j)$.
        ii. If $m \neq \perp$, return $m$. Otherwise, continue to next $j$.
    b. If $k = n+1$, return $\perp$. Otherwise, continue to next $k$.
```

Parameters. When the above construction is instantiated with Gentry's fully secure AIBE scheme in the CPA setting [51], we obtain the following parameter lengths. Let $\overline{\mathbb{G}}$ and $\overline{\mathbb{G}}_{\mathrm{T}}$ be the two groups with prime order $\overline{q}$ in Gentry's construction. MSK is just one element in $\mathbb{Z}_{\overline{q}}$ and the integer N. MPK is only 3 group elements in $\overline{\mathbb{G}}$. The user secret key consists of $(\log N + 1)$ elements in $\mathbb{Z}_{\overline{q}}$ and $(\log N + 1)$ elements in $\overline{\mathbb{G}}$. The ciphertext consists of $\left\lfloor r \log\left(\frac{N}{r}\right)\right\rfloor$ elements in $\overline{\mathbb{G}}$ and $2\left\lfloor r \log\left(\frac{N}{r}\right)\right\rfloor$ elements in $\overline{\mathbb{G}}_{\mathrm{T}}$. Also notice that the $\mathsf{Encrypt}$ algorithm in Gentry's AIBE-IND-CPA-secure scheme does not require any pairing computations since they can be precomputed.

The correctness of the above generic public-key construction in the CPA setting follows from the correctness of the underlying AIBE scheme. In Theorem 3.4.1 given below, we establish the security of this construction based on the security of the underlying AIBE scheme.

Theorem 3.4.1: *If the AIBE scheme* $\Pi' = (\mathsf{Setup}', \mathsf{Extract}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ *is* (t, Q_U, ϵ)*-AIBE-IND-CPA-secure, then the above construction is* $\left(t, Q_U, 2\,\epsilon\, r \log\left(\frac{N}{r}\right)\right)$*-oABE-IND-CPA-secure.* □

Proof. Let r be the number of revoked users in the challenge ciphertext, let $L := \left\lfloor r \log\left(\frac{N}{r}\right)\right\rfloor$, and let $\pi : \{1, \ldots, L\} \to \{1, \ldots, L\}$ be a random permutation. For $b \in \{0, 1\}$, let S_b^* be the set of authorized receivers chosen by the adversary in the *Challenge* step, and let Cover_b denote the family of subtrees covering the set S_b^* according to the complete subtree method of the public-key extension of the subset cover framework. Let $l_b := |\mathsf{Cover}_b|$. For each $1 \leq j \leq l_b$ and for each subtree T_j^b in Cover_b, let HID_j^b be the hierarchical identifier associated with the root of T_j^b.

We organize our proof as a sequence of games, $\text{Game}_0^0, \ldots, \text{Game}_{l_0}^0 \equiv \text{Game}_{l_1}^1, \ldots,$ Game_0^1, between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$. During the *Challenge* step of the first game (Game_0^0), $\mathcal{A}$ receives an encryption of m_0^* for S_0^* and in the last game (Game_0^1), $\mathcal{A}$ receives an encryption of m_1^* for S_1^*.

Game$_\mathbf{0}^\mathbf{0}$: This game corresponds to the game given in Definition 3.3.4 when the challenge bit b^* is fixed to 0. The interaction between $\mathcal{A}$ and $\mathcal{C}$ during *Setup*, *Phase 1*, and *Phase 2* steps follow exactly as specified in Definition 3.3.4. During *Challenge* step, $\mathcal{A}$ gives $\mathcal{C}$ two equal length messages $m_0^*, m_1^* \in \mathcal{MSP}$ and two equal length sets of user identities $S_0^*, S_1^* \subseteq U$ with the restriction that $R_U \cap (S_0^* \cup S_1^*) = \emptyset$, where R_U is the set of users that $\mathcal{A}$ has corrupted during *Phase 1*. $\mathcal{C}$ computes the challenge ciphertext c^*, which will subsequently be sent to $\mathcal{A}$, as follows.

1. For $j \coloneqq 1$ to l_0, compute $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}^0_j, m^*_0)$.
2. Set $\widetilde{m} \leftarrow_\$ \{0,1\}^{|m^*_0|}$.
3. For $j \coloneqq l_0 + 1$ to L, compute $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$.
4. Set $c^* \coloneqq \big(c_{\pi(1)}, \ldots, c_{\pi(L)}\big)$.

Eventually, $\mathcal{A}$ outputs a bit b and wins if $b = 0$.

Game^0_h(1 ≤ h ≤ l_0): This games is similar to Game$^0_{h-1}$, but $\mathcal{C}$ computes the challenge ciphertext c^* as follows.

1. For $j \coloneqq 1$ to $l_0 - h$, compute $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}^0_j, m^*_0)$.
2. Set $\widetilde{m} \leftarrow_\$ \{0,1\}^{|m^*_0|}$.
3. For $j \coloneqq l_0 - h + 1$ to L, compute $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$.
4. Set $c^* \coloneqq \big(c_{\pi(1)}, \ldots, c_{\pi(L)}\big)$.

At the end, $\mathcal{A}$ outputs a bit b and wins if $b = 0$.

Game$^1_{l_1}$: This game is identical to Game$^0_{l_0}$.

Game^1_k(0 ≤ k < l_1): This game is similar to Game$^1_{k+1}$, but the challenge ciphertext c^* is now computed by $\mathcal{C}$ as follows.

1. For $j \coloneqq 1$ to $l_1 - k$, compute $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}^1_j, m^*_1)$.
2. Set $\widetilde{m} \leftarrow_\$ \{0,1\}^{|m^*_1|}$.
3. For $j \coloneqq l_1 - k + 1$ to L, compute $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$.
4. Set $c^* \coloneqq \big(c_{\pi(1)}, \ldots, c_{\pi(L)}\big)$.

Finally, $\mathcal{A}$ outputs a bit b and wins if $b = 0$.

For $0 \leq i \leq l_0$ and $0 \leq j \leq l_1$, let $\mathsf{Adv}^{0,i}_{\mathcal{A},\Pi}$ and $\mathsf{Adv}^{1,j}_{\mathcal{A},\Pi}$ denote $\mathcal{A}$'s advantage in winning Game^0_i and Game^1_j, respectively. In Lemma 3.4.2, we show that if the underlying AIBE scheme is (t, Q_U, ϵ)-AIBE-IND-CPA secure, then $\mathcal{A}$'s advantage in distinguishing Game^0_{h-1} from Game^0_h is at most ϵ. Similarly, Lemma 3.4.3 states that under similar conditions $\mathcal{A}$'s advantage in distinguishing Game^1_{k+1} from Game^1_k is at most ϵ. Therefore,

$$\begin{aligned}\left|\mathsf{Adv}^{0,0}_{\mathcal{A},\Pi} - \mathsf{Adv}^{1,0}_{\mathcal{A},\Pi}\right| &\le \epsilon\,(l_0 + l_1)\\ &\le 2\,\epsilon\,L\\ &\le 2\,\epsilon\,r\log\left(\frac{N}{r}\right). \qquad \blacksquare\end{aligned}$$

Lemma 3.4.2: *For $1 \le h \le l_0$, if the underlying anonymous identity-based encryption scheme Π' is (t, Q_U, ϵ)-AIBE-IND-CPA-secure, then $\mathcal{A}$'s advantage in distinguishing Game$^0_{h-1}$ from Game^0_h is at most ϵ. In other words,*

$$\left|\mathsf{Adv}^{0,h-1}_{\mathcal{A},\Pi} - \mathsf{Adv}^{0,h}_{\mathcal{A},\Pi}\right| \le \epsilon. \qquad \square$$

Proof. We build a PPT adversary $\mathcal{B}$ that runs the AIBE-IND-CPA game with its challenger $\mathcal{C}'$ as follows. First, $\mathcal{B}$ receives the master public key MPK' of the AIBE scheme from $\mathcal{C}'$. Next, $\mathcal{B}$ internally executes the oABE-IND-CPA game with $\mathcal{A}$ in order to gain advantage in the AIBE-IND-CPA game. The details of the interaction between $\mathcal{C}'$, $\mathcal{B}$, and $\mathcal{A}$ are given below.

Setup: $\mathcal{B}$ forwards MPK' to $\mathcal{A}$. $\mathcal{B}$ also initializes the set of revoked users $R_U := \emptyset$.

Phase 1: When $\mathcal{A}$ invokes a secret-key query for user i, first, $\mathcal{B}$ computes HID_i, which is the hierarchical identifier associated with the user i in the binary tree $\mathcal{T}$. Next, for $k := 1$ to $n+1$, $\mathcal{B}$ obtains the secret key $sk_{i,k}$ of the identity $\mathsf{HID}_{i|k}$ from its challenger $\mathcal{C}'$. After adding i to R_U, $\mathcal{B}$ sends to $\mathcal{A}$ the secret key of the user i as $sk_i := (sk_{i,1}, \ldots, sk_{i,n+1})$.

Challenge: $\mathcal{B}$ receives from $\mathcal{A}$ two equal length messages $m_0^*, m_1^* \in \mathcal{MSP}$ and two equal length sets of user identities $S_0^*, S_1^* \subseteq U$ with the restriction that $R_U \cap (S_0^* \cup S_1^*) = \emptyset$. $\mathcal{B}$ draws $\widetilde{m} \leftarrow\$ \{0,1\}^{|m_0^*|}$ and computes the components of its challenge query as follows.

$$id_0' = \mathsf{HID}_{l_0-h+1}^0, \quad id_1' = \mathtt{dummy}, \quad m_0' = m_0^*, \quad m_1' = \widetilde{m}$$

Observe that the condition $R_U \cap (S_0^* \cup S_1^*) = \emptyset$, together with the key assignment strategy of the CS method guarantees that the identity id_0' hadn't been queried to $\mathcal{B}$'s key generation oracle, and thus this is a valid challenge query to $\mathcal{C}'$.

Next, $\mathcal{B}$ sends the two identities id_0', id_1' and the two messages m_0', m_1' as the challenge query to $\mathcal{C}'$. $\mathcal{C}'$ picks a random bit $b' \in \{0,1\}$ and sends $c' \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', id_{b'}', m_{b'}')$ to $\mathcal{B}$. Finally, $\mathcal{B}$ computes the challenge ciphertext c^*, which is eventually sent to $\mathcal{A}$, as follows.

1. For $j \coloneqq 1$ to $l_0 - h$, compute $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}_j^0, m_0^*)$.
2. Set $c_{l_0-h+1} \coloneqq c'$.
3. For $j \coloneqq l_0 - h + 2$ to L, compute $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$.
4. Set $c^* \coloneqq \left(c_{\pi(1)}, \ldots, c_{\pi(L)}\right)$.

Phase 2: This phase is handled similarly to *Phase 1* with the usual restriction that $\mathcal{A}$ does not invoke a secret-key query i such that $i \in S_0^* \cup S_1^*$.

Guess: $\mathcal{A}$ outputs a guess b and $\mathcal{B}$ passes this bit as its guess for b' to $\mathcal{C}'$.

Observe that, by construction, it holds that if $\mathcal{C}'$ chooses $b' = 0$, then $\mathcal{B}$ is playing Game_{h-1}^0, whereas if $b' = 1$, then $\mathcal{B}$ is playing Game_h^0. Therefore, $\mathcal{B}$'s AIBE-IND-CPA advantage is equivalent to $\mathcal{A}$'s advantage in distinguishing Game_{h-1}^0 from Game_h^0. More formally,

$$\left|\mathsf{Adv}_{\mathcal{A},\Pi}^{0,h-1} - \mathsf{Adv}_{\mathcal{A},\Pi}^{0,h}\right| = \mathsf{Adv}_{\mathcal{B},\Pi'}^{\text{AIBE-IND-CPA}} \leq \epsilon.$$ ■

Lemma 3.4.3: *For $0 \leq k < l_1$, if the underlying anonymous identity-based encryption scheme Π' is (t, Q_U, ϵ)-AIBE-IND-CPA-secure, then $\mathcal{A}$'s advantage in distinguishing Game$^1_{k+1}$ from Game^1_k is at most ϵ. More precisely,*

$$\left|\mathsf{Adv}^{1,k+1}_{\mathcal{A},\Pi} - \mathsf{Adv}^{1,k}_{\mathcal{A},\Pi}\right| \leq \epsilon. \qquad \square$$

Proof. The argument is analogous to the proof of Lemma 3.4.2. ■

3.4.2 A Generic oABE-IND-CCA-Secure Public-Key Construction

Given a weakly robust AIBE-IND-CCA-secure anonymous identity-based encryption scheme $\Pi' = (\mathsf{Setup}', \mathsf{Extract}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ and a strong existentially unforgeable one-time signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$, we construct an oABE-IND-CCA-secure outsider-anonymous broadcast encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as given below. The definition of strong existential unforgeability for one-time signature schemes is given in Section 2.4.2.

Setup$(1^\lambda, N)$: Obtain $(\mathsf{MPK}', \mathsf{MSK}') \leftarrow \mathsf{Setup}'(1^\lambda)$. Output MPK and MSK as

$$\mathsf{MPK} \coloneqq (\mathsf{MPK}', N) \quad \mathsf{MSK} \coloneqq \mathsf{MSK}'.$$

KeyGen(MPK, MSK, i): Let $\mathsf{HID}_i \coloneqq (\mathsf{Root}, \mathsf{ID}_1, \ldots, \mathsf{ID}_n)$ be the hierarchical identifier associated with user i in the binary tree $\mathcal{T}$. For $k \coloneqq 1$ to $n+1$, compute $sk_{i,k} \leftarrow \mathsf{Extract}'(\mathsf{MPK}', \mathsf{MSK}', \mathsf{HID}_{i|k})$. Output the secret key sk_i of user i as

$$sk_i \coloneqq (sk_{i,1}, \ldots, sk_{i,n+1}).$$

Encrypt(MPK, S, m): Generate $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$. Let Cover be the family of subtrees covering the set of receivers S according to the CS method. For each subtree T_j in Cover, let HID_j be the hierarchical identifier associated with the root of T_j. Let $l \coloneqq |\mathsf{Cover}|$, $r \coloneqq N - |S|$ and $L \coloneqq \left\lfloor r \log\left(\frac{N}{r}\right)\right\rfloor$. For $1 \leq j \leq l$, compute $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}_j, \mathsf{VK}\|m)$. Set $\widetilde{m} \leftarrow_{\$} \{0,1\}^{|\mathsf{VK}\|m|}$. For $l+1 \leq j \leq L$, compute $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$, where $\mathtt{dummy}$ is a special identifier used to obtain padding ciphertext components. Compute $\widehat{c}$ as

$$\widehat{c} \coloneqq \left(c_{\pi(1)}, \ldots, c_{\pi(L)}\right),$$

where $\pi : \{1, \ldots, L\} \rightarrow \{1, \ldots, L\}$ is a random permutation.

Generate $\sigma \leftarrow \mathsf{Sign}(\mathsf{SK}, \mathsf{VK}\|\widehat{c})$, and output $c \coloneqq (\sigma, \mathsf{VK}, \widehat{c})$.

Decrypt(MPK, sk_i, c): Parse the secret key sk_i as the tuple $(sk_{i,1}, \ldots, sk_{i,n+1})$ and the ciphertext c as $(\sigma, \mathsf{VK}, \widehat{c} = (c_1, \ldots, c_L))$.

```
1. For $k \coloneqq 1$ to $n+1$,
    a. For $j \coloneqq 1$ to $L$,
        i. Compute $m' \coloneqq \mathsf{Decrypt}'(\mathsf{MPK}', sk_{i,k}, c_j)$.
        ii. If $m' \neq \perp$, parse $m'$ as $\mathsf{VK}\|m$ and return $m$ if
            $\mathsf{Vrfy}(\mathsf{VK}, \sigma, \mathsf{VK}\|\widehat{c}) = 1$. Otherwise, continue to next $j$.
    b. If $k = n+1$, return $\perp$. Otherwise, continue to next $k$.
```

Parameters. The parameter lengths of the above construction when instantiated with Gentry's Fully Secure AIBE scheme in the CCA setting [51] are as follows. Let $\overline{\mathbb{G}}$ and $\overline{\mathbb{G}}_{\mathrm{T}}$ be the two groups with prime order $\overline{q}$ in Gentry's construction. MSK is one element in $\mathbb{Z}_{\overline{q}}$ and the integer N. MPK consists of 5 group elements in $\overline{\mathbb{G}}$ and the definition of a hash function $\overline{H}$ from a family of universal one-way hash functions. The user secret key consists of $3(\log N + 1)$ elements in $\mathbb{Z}_{\overline{q}}$ and $3(\log N + 1)$ elements in $\overline{\mathbb{G}}$. The ciphertext consists of $\left\lfloor r \log\left(\frac{N}{r}\right)\right\rfloor$ elements in $\overline{\mathbb{G}}$ and $3\left\lfloor r \log\left(\frac{N}{r}\right)\right\rfloor$ elements in $\overline{\mathbb{G}}_{\mathrm{T}}$.

Similar to Gentry's AIBE-IND-CPA-secure construction, the $\mathsf{Encrypt}$ algorithm in his AIBE-IND-CCA-secure construction does not require any pairing computations since they can be precomputed.

The correctness of the above generic public-key construction in the CCA setting follows from the correctness of the underlying signature and AIBE schemes. The security of this construction is established in Theorem 3.4.4 below.

Theorem 3.4.4: *If the one-time signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is (t, Q_D, ϵ_1)-SIG-SEU-secure and the AIBE scheme $\Pi' = (\mathsf{Setup}', \mathsf{Extract}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ is $(t, Q_U, Q_D, \epsilon_2)$-AIBE-IND-CCA-secure, then the construction given above is $\Big(t, Q_U, Q_D, 2\,(\epsilon_1 + \epsilon_2)\, r \log\big(\frac{N}{r}\big)\Big)$-oABE-IND-CCA-secure.* □

Proof. Let r, L, π, S_b^*, Cover_b, l_b, T_j^b, and HID_j^b be as defined in the proof of Theorem 3.4.1. We organize our proof as a sequence of games, $\text{Game}_0^0, \ldots, \text{Game}_{l_0}^0 \equiv \text{Game}_{l_1}^1, \ldots,\ \text{Game}_0^1$, between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$. During the *Challenge* step of the first game (Game_0^0), $\mathcal{A}$ receives an encryption of m_0^* for S_0^* and in the last game (Game_0^1), $\mathcal{A}$ receives an encryption of m_1^* for S_1^*.

Game_0^0: This game corresponds to the game given in Definition 3.3.3 when the challenge bit b^* is fixed to 0. The interaction between $\mathcal{A}$ and $\mathcal{C}$ during *Setup*, *Phase 1*, and *Phase 2* steps follow exactly as specified in Definition 3.3.3. During *Challenge* step, $\mathcal{A}$ gives $\mathcal{C}$ two equal length messages $m_0^*, m_1^* \in \mathcal{MSP}$ and two equal length sets of user identities $S_0^*, S_1^* \subseteq U$ with the restriction that $R_U \cap (S_0^* \cup S_1^*) = \emptyset$, where R_U is the set of users that $\mathcal{A}$ corrupted during *Phase 1*. $\mathcal{C}$ computes the challenge ciphertext c^*, which will subsequently be sent to $\mathcal{A}$, as follows.

1. Generate $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$.
2. For $j := 1$ to l_0, compute $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}_j^0, \mathsf{VK}\|m_0^*)$.
3. Set $\widetilde{m} \leftarrow_\$ \{0,1\}^{|\mathsf{VK}\|m_0^*|}$.

4. For $j := l_0 + 1$ to L, compute $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$.
5. Set $\widehat{c} := \left(c_{\pi(1)}, \ldots, c_{\pi(L)}\right)$.
6. Generate $\sigma \leftarrow \mathsf{Sign}(\mathsf{SK}, \mathsf{VK} \| \widehat{c})$, and set $c^* := (\sigma, \mathsf{VK}, \widehat{c})$.

Eventually, $\mathcal{A}$ outputs a bit b and wins if $b = 0$.

Game$_h^0$(1 ≤ h ≤ l_0): This game is similar to Game$_{h-1}^0$, but $\mathcal{C}$ computes the challenge ciphertext c^* as follows.

1. Generate $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$.
2. For $j := 1$ to $l_0 - h$, compute $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}_j^0, \mathsf{VK} \| m_0^*)$.
3. Set $\widetilde{m} \leftarrow\$ \{0,1\}^{|\mathsf{VK} \| m_0^*|}$.
4. For $j := l_0 - h + 1$ to L, compute $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$.
5. Set $\widehat{c} := \left(c_{\pi(1)}, \ldots, c_{\pi(L)}\right)$.
6. Generate $\sigma \leftarrow \mathsf{Sign}(\mathsf{SK}, \mathsf{VK} \| \widehat{c})$, and set $c^* := (\sigma, \mathsf{VK}, \widehat{c})$.

At the end, $\mathcal{A}$ outputs a bit b and wins if $b = 0$.

Game$_{l_1}^1$: This game is identical to Game$_{l_0}^0$.

Game$_k^1$(0 ≤ k < l_1): This game is similar to Game$_{k+1}^1$, but c^* is now computed by $\mathcal{C}$ as follows.

1. Generate $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$.
2. For $j := 1$ to $l_1 - k$, compute $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}_j^1, \mathsf{VK} \| m_1^*)$.
3. Set $\widetilde{m} \leftarrow\$ \{0,1\}^{|\mathsf{VK} \| m_1^*|}$.
4. For $j := l_1 - k + 1$ to L, compute $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$.
5. Set $\widehat{c} := \left(c_{\pi(1)}, \ldots, c_{\pi(L)}\right)$.
6. Generate $\sigma \leftarrow \mathsf{Sign}(\mathsf{SK}, \mathsf{VK} \| \widehat{c})$, and set $c^* := (\sigma, \mathsf{VK}, \widehat{c})$.

Finally, $\mathcal{A}$ outputs a bit b and wins if $b = 0$.

For $0 \leq i \leq l_0$ and $0 \leq j \leq l_1$, let $\mathsf{Adv}^{0,i}_{\mathcal{A},\Pi}$ and $\mathsf{Adv}^{1,j}_{\mathcal{A},\Pi}$ denote $\mathcal{A}$'s advantage of winning Game^0_i and Game^1_j, respectively. In Lemma 3.4.5, we show that if the underlying one-time signature scheme and AIBE scheme are, respectively, (t, Q_D, ϵ_1)-SIG-SEU-secure and $(t, Q_U, Q_D, \epsilon_2)$-AIBE-IND-CCA-secure, then $\mathcal{A}$'s advantage of distinguishing Game$^0_{h-1}$ from Game^0_h is at most $\epsilon_1 + \epsilon_2$. Similarly, Lemma 3.4.6 states that under analogous conditions $\mathcal{A}$'s advantage of distinguishing Game$^1_{k+1}$ from Game^1_k is again at most $\epsilon_1 + \epsilon_2$. Therefore,

$$\begin{aligned}\left|\mathsf{Adv}^{0,0}_{\mathcal{A},\Pi} - \mathsf{Adv}^{1,0}_{\mathcal{A},\Pi}\right| &\leq (\epsilon_1 + \epsilon_2)\,(l_0 + l_1) \\ &\leq 2\,(\epsilon_1 + \epsilon_2)\,L \\ &\leq 2\,(\epsilon_1 + \epsilon_2)\,r \log\left(\frac{N}{r}\right). \qquad \blacksquare\end{aligned}$$

Lemma 3.4.5: *For $1 \leq h \leq l_0$, if the underlying one-time signature scheme Σ is (t, Q_D, ϵ_1)-SIG-SEU-secure and the anonymous identity-based encryption scheme Π' is $(t, Q_U, Q_D, \epsilon_2)$-AIBE-IND-CCA-secure, then $\mathcal{A}$'s advantage of distinguishing Game$^0_{h-1}$ from Game^0_h is at most $\epsilon_1 + \epsilon_2$. In other words,*

$$\left|\mathsf{Adv}^{0,h-1}_{\mathcal{A},\Pi} - \mathsf{Adv}^{0,h}_{\mathcal{A},\Pi}\right| \leq (\epsilon_1 + \epsilon_2). \qquad \square$$

Proof. We build a PPT adversary $\mathcal{B}$ that runs the AIBE-IND-CCA game with its challenger $\mathcal{C}'$ as follows. First, $\mathcal{B}$ receives the master public key MPK' of the AIBE scheme from $\mathcal{C}'$. Next, $\mathcal{B}$ internally executes the oABE-IND-CCA game with $\mathcal{A}$ in order to gain advantage in the AIBE-IND-CCA game. The details of the interaction between $\mathcal{C}'$, $\mathcal{B}$, and $\mathcal{A}$ are given below.

Setup: $\mathcal{B}$ forwards MPK' to $\mathcal{A}$. $\mathcal{B}$ also initializes the set of revoked users $R_U := \emptyset$.

Phase 1: $\mathcal{B}$ replies to $\mathcal{A}$'s queries as follows.

Secret-key query i: First, $\mathcal{B}$ computes HID_i, which is the hierarchical identifier associated with the user i in the binary tree $\mathcal{T}$. Next, for $k := 1$ to $n+1$, $\mathcal{B}$ obtains the secret key $sk_{i,k}$ of the identity $\mathsf{HID}_{i|k}$ from its challenger $\mathcal{C}'$. After adding i to R_U, $\mathcal{B}$ sends to $\mathcal{A}$ the secret key of the user i as $sk_i := (sk_{i,1}, \ldots, sk_{i,n+1})$.

Decryption query (i, c): First, $\mathcal{B}$ parses c as $(\sigma, \mathsf{VK}, \widehat{c} = (c_1, \ldots, c_L))$. Then, $\mathcal{B}$ computes HID_i, and for each $k := 1$ to $n+1$, proceeds as follows.

- If $\mathcal{B}$ obtained the secret key $sk_{i,k}$ corresponding to the identity $\mathsf{HID}_{i|k}$ in the process of responding to a previous secret-key query, then $\mathcal{B}$ attempts to decrypt in turn all ciphertext components $c_1, \ldots, c_L$ in $\widehat{c}$ using the secret key $sk_{i,k}$. If any of these decryption attempts yield a non-$\perp$ value $\mathsf{VK}\|m$, then $\mathcal{B}$ returns m to $\mathcal{A}$ if $\mathsf{Vrfy}(\mathsf{VK}, \sigma, \mathsf{VK}\|\widehat{c}) = 1$. Otherwise, $\mathcal{B}$ continues to next k.
- If $\mathcal{B}$ did not obtain the secret key $sk_{i,k}$ of the identity $\mathsf{HID}_{i|k}$ from an earlier secret-key query, then $\mathcal{B}$ makes L decryption queries to its challenger $\mathcal{C}'$, one for each ciphertext component $c_1, \ldots, c_L$, all under identity $\mathsf{HID}_{i|k}$. If any of these decryption queries return a non-$\perp$ value $\mathsf{VK}\|m$, then $\mathcal{B}$ returns m to $\mathcal{A}$ if $\mathsf{Vrfy}(\mathsf{VK}, \sigma, \mathsf{VK}\|\widehat{c}) = 1$. Otherwise, $\mathcal{B}$ continues to next j.

If all the above decryption attempts return $\perp$, then $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$.

Challenge: $\mathcal{B}$ receives from $\mathcal{A}$ two equal length messages $m_0^*, m_1^* \in \mathcal{MSP}$ and two equal length sets of user identities $S_0^*, S_1^* \subseteq U$ with the restriction that $R_U \cap (S_0^* \cup S_1^*) = \emptyset$. $\mathcal{B}$ generates $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$, selects a random string

$\widetilde{m} \leftarrow_{\$} \{0,1\}^{|\mathsf{VK}\|m_0^*|}$, and sets

$$id_0' = \mathsf{HID}^0_{l_0-h+1}, \quad id_1' = \mathtt{dummy}, \quad m_0' = \mathsf{VK}\|m_0^*, \quad m_1' = \widetilde{m}.$$

Next, $\mathcal{B}$ sends the two identities id_0', id_1' and the two messages m_0', m_1' as the challenge query to $\mathcal{C}'$. $\mathcal{C}'$ picks a random bit $b' \in \{0,1\}$ and responds to $\mathcal{B}$ with $c' \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', id_{b'}', m_{b'}')$. Finally, $\mathcal{B}$ computes the challenge ciphertext c^*, which is eventually sent to $\mathcal{A}$, as follows,

1. For $j \coloneqq 1$ to $l_0 - h$, compute $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}^0_j, \mathsf{VK}\|m_0^*)$.
2. Set $c_{l_0-h+1} \coloneqq c'$.
3. For $j \coloneqq l_0 - h + 2$ to L, compute $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$.
4. Set $\widehat{c} \coloneqq (c_{\pi(1)}, \ldots, c_{\pi(L)})$.
5. Generate $\sigma \leftarrow \mathsf{Sign}(\mathsf{SK}, \mathsf{VK}\|\widehat{c})$, and set $c^* \coloneqq (\sigma, \mathsf{VK}, \widehat{c})$.

Phase 2: $\mathcal{B}$ replies to $\mathcal{A}$'s queries as follows.

Secret-key query i: These queries are handled similarly to *Phase 1*, with the usual restriction that $\mathcal{A}$ does not invoke a secret-key query i such that $i \in S_0^* \cup S_1^*$.

Decryption query (i, c): $\mathcal{B}$ parses c as $(\sigma, \mathsf{VK}, \widehat{c} = (c_1, \ldots, c_L))$ and replies according to one of the following cases.

- If $c = c^*$ and $i \notin S_0^* \cup S_1^*$, then $\mathcal{B}$ proceeds as in *Phase 1*. (Note that in this case $\mathcal{B}$'s output will be $\perp$, as it should be.)
- If $c = c^*$, and $i \in S_0^* \cup S_1^*$, $\mathcal{B}$ just rejects, since $\mathcal{A}$ is submitting an invalid query.
- If $c \neq c^*$ and $i \notin S_0^*$, then $\mathcal{B}$ proceeds as in *Phase 1*.
- If $c \neq c^*$ and $i \in S_0^*$, then $\mathcal{B}$ computes HID_i, and proceeds as follows:

- ◇ If for all $k := 1$ to $n+1$, it is the case that $\mathsf{HID}_{i|k} \neq \mathsf{HID}^0_{l_0-h+1}$, then $\mathcal{B}$ proceeds as in *Phase 1*. Observe that the condition $\forall k \in [1, n+1] : \mathsf{HID}_{i|k} \neq \mathsf{HID}^0_{l_0-h+1}$ ensures that all the decryption queries that $\mathcal{B}$ will make to its challenger $\mathcal{C}'$ in the process of responding to $\mathcal{A}$'s queries are allowed.
- ◇ If $\exists\ k \in [1, n+1]$ such that $\mathsf{HID}_{i|k} = \mathsf{HID}^0_{l_0-h+1}$, and c' does not appear among the ciphertext components of $\widehat{c}$, then again $\mathcal{B}$ proceeds as in *Phase 1*. Observe that the condition that $\widehat{c}$ does not contain c' ensures that also in this case all the decryption queries that $\mathcal{B}$ will make to its challenger $\mathcal{C}'$ in the process of responding to $\mathcal{A}$'s queries are allowed.
- ◇ If $\exists\ k \in [1, n+1]$ such that $\mathsf{HID}_{i|k} = \mathsf{HID}^0_{l_0-h+1}$, but c' appears among the ciphertext components of $\widehat{c}$, then $\mathcal{B}$ outputs $\perp$. To see that $\perp$ is the correct reply, observe that in the real oABE-IND-CCA game, a decryption query (i, c) of this type will trigger decryption of the c' component. Since by construction c' is the encryption of $\mathsf{VK}\|m^*_0$, and $c \neq c^*$, by the strong existential unforgeability of the underlying one-time signature scheme, the verification test of the decryption algorithm would fail, thus yielding $\perp$ as output.

Guess: $\mathcal{A}$ outputs a guess b and $\mathcal{B}$ passes this bit as its guess for b' to $\mathcal{C}'$.

Observe that, by construction, it holds that if $\mathcal{C}'$ chooses $b' = 0$, then $\mathcal{B}$ is playing Game^0_{h-1}, whereas if $b' = 1$, then $\mathcal{B}$ is playing Game^0_h. Therefore, up to forgeries of the underlying one-time signature scheme, $\mathcal{B}$'s AIBE-IND-CCA advantage is essentially $\mathcal{A}$'s advantage in distinguishing Game^0_{h-1} from Game^0_h. Therefore,

$$\left|\mathsf{Adv}^{0,h-1}_{\mathcal{A},\Pi} - \mathsf{Adv}^{0,h}_{\mathcal{A},\Pi}\right| \leq (\epsilon_1 + \epsilon_2). \qquad \blacksquare$$

Lemma 3.4.6: *For $0 \leq k < l_1$, if the underlying one-time signature scheme Σ is (t, Q_D, ϵ_1)-SIG-SEU-secure and the underlying anonymous identity-based encryption scheme Π' is $(t, Q_U, Q_D, \epsilon_2)$-AIBE-IND-CCA-secure, then $\mathcal{A}$'s advantage of distinguishing Game$^1_{k+1}$ from Game^1_k is at most ϵ. More precisely,*

$$\left|\mathsf{Adv}^{1,k+1}_{\mathcal{A},\Pi} - \mathsf{Adv}^{1,k}_{\mathcal{A},\Pi}\right| \leq (\epsilon_1 + \epsilon_2). \qquad \square$$

Proof. The argument is analogous to the proof of Lemma 3.4.5. ■

3.4.3 An Enhanced oABE-IND-CCA-Secure Public-Key Construction in the Random Oracle Model

The main limitation of our generic public-key constructions is the running time of the decryption algorithm. As described in the opening paragraphs of Section 3.4, decryption amounts to performing $\left(\left\lfloor r \log\left(\frac{N}{r}\right)\right\rfloor \log N\right)/2$ AIBE decryption attempts on average. The root cause behind this limitation is the decryption process's inability to identify the correct AIBE ciphertext component efficiently. In this section, we describe an enhancement of our generic public-key construction under the computational Diffie-Hellman assumption, in the random oracle model. The definition of the CDH problem is given in Section 2.2.1.

The main idea of this enhancement is to adapt the techniques of [10] to the structure of our ciphertexts and attach a unique tag to each AIBE ciphertext component of a given oABE ciphertext. With this optimization, the $\mathsf{Decrypt}$ algorithm is able to identify the correct AIBE ciphertext component via a linear search through the whole oABE ciphertext components, at which point a single AIBE decryption operation suffices to recover the original plaintext. This yields an asymptotic decryption time of $O\left(r \log\left(\frac{N}{r}\right) \log N\right)$, but in fact this is in a sense an overestimate, since the cost of searching for the correct ciphertext component is much less than carrying out multiple decryption attempts.

Given a weakly robust AIBE-IND-CCA-secure anonymous identity-based encryption scheme $\Pi' = (\mathsf{Setup}', \mathsf{Extract}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ and a strong existentially unforgeable one-time signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$, we construct an oABE-IND-CCA-secure outsider-anonymous broadcast encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with enhanced decryption as follows. In this construction, $\mathbb{G} = \langle g \rangle$ denotes a group with prime order $q > 2^\lambda$ in which computational Diffie-Hellman problem is hard, the decisional Diffie-Hellamn problem is easy, and g is a group generator. $H : \mathbb{G} \to \{0,1\}^\lambda$ is a cryptographic hash function that will be modeled as a random oracle in the security analysis of this construction.

$\mathsf{Setup}(1^\lambda, N)$: Obtain $(\mathsf{MPK}', \mathsf{MSK}') \leftarrow \mathsf{Setup}'(1^\lambda)$. For each node (with the hierarchical identifier HID) in $\mathcal{T}$, draw $a_{\mathsf{HID}} \leftarrow\!\$\ \mathbb{Z}_q$, and compute $A_{\mathsf{HID}} := g^{a_{\mathsf{HID}}}$. Output MPK and MSK as

$$\mathsf{MPK} := \Big(\mathsf{MPK}', N, \mathbb{G}, q, g, \{A_{\mathsf{HID}}\}_{\mathsf{HID} \in \mathcal{T}}\Big), \quad \mathsf{MSK} := \Big(\mathsf{MSK}', \{a_{\mathsf{HID}}\}_{\mathsf{HID} \in \mathcal{T}}\Big).$$

$\mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)$: Let $\mathsf{HID}_i := (\mathsf{Root}, \mathsf{ID}_1, \dots, \mathsf{ID}_n)$ be the hierarchical identifier associated with user i in the binary tree $\mathcal{T}$. For $k := 1$ to $n+1$, set $\overline{sk}_{i,k} := a_{\mathsf{HID}_{i|k}}$, and compute $sk_{i,k} \leftarrow \mathsf{Extract}'(\mathsf{MPK}', \mathsf{MSK}', \mathsf{HID}_{i|k})$. Output the secret key sk_i of user i as

$$sk_i := \Big(\big(\overline{sk}_{i,1}, sk_{i,1}\big), \dots, \big(\overline{sk}_{i,n+1}, sk_{i,n+1}\big)\Big).$$

$\mathsf{Encrypt}(\mathsf{MPK}, S, m)$: Generate $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$. Let Cover be the family of subtrees covering the set of receivers S according to the CS method. For each subtree T_j in Cover, let HID_j be the hierarchical identifier associated with the root of T_j. Let $l := |\mathsf{Cover}|$, $r := N - |S|$ and $L := \Big\lfloor r \log\Big(\frac{N}{r}\Big)\Big\rfloor$. Draw

$s \leftarrow\$ \mathbb{Z}_q$, and compute $\bar{c}_0 := g^s$. For $1 \le j \le l$, compute $\bar{c}_j := H(A^s_{\mathsf{HID}_j})$, $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}_j, \mathsf{VK}\|A^s_{\mathsf{HID}_j}\|m)$. Set $\widetilde{m} \leftarrow\$ \{0,1\}^{|\mathsf{VK}\|\bar{c}_0\|m|}$. For $l+1 \le j \le L$, set $s_j \leftarrow\$ \mathbb{Z}_q$, and compute $\bar{c}_j := H(g^{s_j})$, $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$, where $\mathtt{dummy}$ is a special identifier used to obtain padding ciphertext components. Compute $\widehat{c}$ as

$$\widehat{c} := \left(\bar{c}_0, \left(\bar{c}_{\pi(1)}, c_{\pi(1)}\right), \ldots, \left(\bar{c}_{\pi(L)}, c_{\pi(L)}\right)\right),$$

where $\pi : \{1, \ldots, L\} \to \{1, \ldots, L\}$ is a random permutation.

Generate $\sigma \leftarrow \mathsf{Sign}(\mathsf{SK}, \mathsf{VK}\|\widehat{c}\,)$, and output $c := (\sigma, \mathsf{VK}, \widehat{c}\,)$.

Decrypt(MPK, sk_i, c): First, parse the secret key sk_i as the tuple $\left(\left(\overline{sk}_{i,1}, sk_{i,1}\right), \ldots, \left(\overline{sk}_{i,n+1}, sk_{i,n+1}\right)\right)$ and the ciphertext c as $(\sigma, \mathsf{VK}, \widehat{c} = (\bar{c}_0, (\bar{c}_1, c_1), \ldots, (\bar{c}_L, c_L)))$.

```
1. For $k := 1$ to $n+1$,
    a. Compute $tag_k := H(\bar{c}_0^{\,\overline{sk}_{i,k}})$
2. Check whether $\exists k \in [1, n+1]\ \exists j \in [1, L]$ such that $tag_k = \bar{c}_j$
    a. If suitable $k, j$ exist, compute $m' := \mathsf{Dec}(\mathsf{MPK}', sk_{i,k}, c_j)$.
    b. If $m'$ can be parsed as $\mathsf{VK}\|\bar{c}_0^{\,\overline{sk}_{i,k}}\|m$ and
       $\mathsf{Vrfy}(\mathsf{VK}, \sigma, \mathsf{VK}\|\widehat{c}\,) = 1$, return $m$.
    c. Otherwise, return $\perp$.
```

Remark 3.4.7. Notice that the check in Step 2 of the $\mathsf{Decrypt}$ algorithm can be performed in expected time $O(n+L) = O(L)$, *e.g.*, using a hash table $\mathcal{H}$ to compute the intersection between $\{tag_k\}_{k\in[1,n+1]}$ and $\{\bar{c}_j\}_{j\in[1,L]}$ as follows.

```
1. Initialize $\mathcal{H}$ to be empty.
2. For $k := 1$ to $n+1$
    a. Insert $(tag_k, k)$ in $\mathcal{H}$.
```

3. For $j \coloneqq 1$ to L

   a. Look up an entry of the form $(\overline{c}_j, k)$ in $\mathcal{H}$. If found, return k.

The correctness of the above enhanced public-key construction in the CCA setting follows from the algebraic properties of the group $\mathbb{G}$ and the correctness of the underlying signature and AIBE schemes. The security of this construction is established in Theorem 3.4.8 below.

Theorem 3.4.8: *If the one-time signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is (t, Q_D, ϵ_1)-SIG-SEU-secure, the AIBE scheme $\Pi' = (\mathsf{Setup}', \mathsf{Extract}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ is $(t, Q_U, Q_D, \epsilon_2)$-AIBE-IND-CCA-secure, and CDH problem is (t, ϵ_3)-hard in $\mathbb{G}$ and DDH problem is efficiently computable in $\mathbb{G}$, then the above construction is $\left(t, Q_U, Q_D, 2\,(\epsilon_1 + \epsilon_2 + \epsilon_3)\, r \log\left(\frac{N}{r}\right)\right)$-oABE-IND-CCA-secure, in the random oracle model.* □

Proof. Let r, L, π, S_b^*, Cover_b, l_b, T_j^b, and HID_j^b be as defined in the proof of Theorem 3.4.1. We organize our proof as a sequence of games between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$ as follows.

$$\mathrm{Game}_0^0, \overline{\mathrm{Game}}_1^0, \mathrm{Game}_1^0, \ldots, \overline{\mathrm{Game}}_{l_0}^0, \mathrm{Game}_{l_0}^0 \equiv$$
$$\mathrm{Game}_{l_1}^1, \overline{\mathrm{Game}}_{l_1}^1, \ldots, \mathrm{Game}_1^1, \overline{\mathrm{Game}}_1^1, \mathrm{Game}_0^1$$

During the *Challenge* step of the first game (Game_0^0), $\mathcal{A}$ receives an encryption of m_0^* for S_0^* and in the last game (Game_0^1), $\mathcal{A}$ receives an encryption of m_1^* for S_1^*.

$\mathbf{Game_0^0}$: This game corresponds to the game given in Definition 3.3.3 when the challenge bit b^* is fixed to 0. The interaction between $\mathcal{A}$ and $\mathcal{C}$ during *Setup*, *Phase 1*, and *Phase 2* steps follow exactly as specified in Definition 3.3.3. During *Challenge* step, $\mathcal{A}$ gives $\mathcal{C}$ two equal length messages $m_0^*, m_1^* \in \mathcal{MSP}$ and two equal length sets of user identities $S_0^*, S_1^* \subseteq U$ with the restriction that

$R_U \cap (S_0^* \cup S_1^*) = \emptyset$, where R_U is the set of users that $\mathcal{A}$ corrupted during *Phase 1*. $\mathcal{C}$ computes the challenge ciphertext c^*, which will subsequently be sent to $\mathcal{A}$, as follows.

1. Generate $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$.
2. Draw $s \leftarrow\$ \mathbb{Z}_q$, and compute $\overline{c}_0 := g^s$.
3. For $j := 1$ to l_0, compute $\overline{c}_j := H\Big(A^s_{\mathsf{HID}^0_j}\Big)$,
   $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}^0_j, \mathsf{VK} \| A^s_{\mathsf{HID}^0_j} \| m_0^*)$.
4. Set $\widetilde{m} \leftarrow\$ \{0,1\}^{|\mathsf{VK}\|\overline{c}_0\|m_0^*|}$.
5. For $j := l_0 + 1$ to L, set $s_j \leftarrow\$ \mathbb{Z}_q$, and compute $\overline{c}_j := H(g^{s_j})$,
   $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$.
6. Set $\widehat{c} := \Big(\overline{c}_0, \Big(\overline{c}_{\pi(1)}, c_{\pi(1)}\Big), \ldots, \Big(\overline{c}_{\pi(L)}, c_{\pi(L)}\Big)\Big)$.
7. Generate $\sigma \leftarrow \mathsf{Sign}(\mathsf{SK}, \mathsf{VK} \| \widehat{c}\,)$, and set $c^* := (\sigma, \mathsf{VK}, \widehat{c}\,)$.

Eventually, $\mathcal{A}$ outputs a bit b and wins if $b = 0$.

$\overline{\mathbf{Game}}{}^{\mathbf{0}}_{\boldsymbol{h}}\boldsymbol{(1 \leq h \leq l_0)}$**:** This game is similar to Game^0_{h-1}, but $\mathcal{C}$ computes the challenge ciphertext c^* as follows.

1. Generate $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$.
2. Draw $s \leftarrow\$ \mathbb{Z}_q$, and compute $\overline{c}_0 := g^s$.
3. For $j := 1$ to $l_0 - h$, compute $\overline{c}_j := H\Big(A^s_{\mathsf{HID}^0_j}\Big)$,
   $c_j \leftarrow \mathsf{Enc}(\mathsf{MPK}', \mathsf{HID}^0_j, \mathsf{VK} \| A^s_{\mathsf{HID}^0_j} \| m_0^*)$.
4. Set $\widetilde{m} \leftarrow\$ \{0,1\}^{|\mathsf{VK}\|\overline{c}_0\|m_0^*|}$.
5. Compute $\overline{c}_{l_0-h+1} := H\Big(A^s_{\mathsf{HID}^0_{l_0-h+1}}\Big)$, $c_{l_0-h+1} \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$.
6. For $j := l_0 - h + 2$ to L, set $s_j \leftarrow\$ \mathbb{Z}_q$, and compute $\overline{c}_j := H(g^{s_j})$,
   $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$.

7. Set $\widehat{c} \coloneqq \Big(\overline{c}_0, \Big(\overline{c}_{\pi(1)}, c_{\pi(1)}\Big), \ldots, \Big(\overline{c}_{\pi(L)}, c_{\pi(L)}\Big)\Big)$.
8. Generate $\sigma \leftarrow \mathsf{Sign}(\mathsf{SK}, \mathsf{VK} \| \widehat{c}\,)$, and set $c^* \coloneqq (\sigma, \mathsf{VK}, \widehat{c}\,)$.

At the end, $\mathcal{A}$ outputs a bit b and wins if $b = 0$.

$\mathbf{Game}^{\mathbf{0}}_{\boldsymbol{h}}\mathbf{(1 \leq \boldsymbol{h} \leq \boldsymbol{l_0})}$**:** This game is similar to $\overline{\text{Game}}^0_h$, but $\mathcal{C}$ computes the challenge ciphertext c^* as follows.

1. Generate $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$.
2. Draw $s \leftarrow\$\ \mathbb{Z}_q$, and compute $\overline{c}_0 \coloneqq g^s$.
3. For $j \coloneqq 1$ to $l_0 - h$, compute $\overline{c}_j \coloneqq H\Big(A^s_{\mathsf{HID}^0_j}\Big)$,
   $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}^0_j, \mathsf{VK} \| A^s_{\mathsf{HID}^0_j} \| m^*_0)$.
4. Set $\widetilde{m} \leftarrow\$\ \{0,1\}^{|\mathsf{VK}\|\overline{c}_0\|m^*_0|}$.
5. For $j \coloneqq l_0 - h + 1$ to L, set $s_j \leftarrow\$\ \mathbb{Z}_q$, and compute $\overline{c}_j \coloneqq H(g^{s_j})$,
   $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$.
6. Set $\widehat{c} \coloneqq \Big(\overline{c}_0, \Big(\overline{c}_{\pi(1)}, c_{\pi(1)}\Big), \ldots, \Big(\overline{c}_{\pi(L)}, c_{\pi(L)}\Big)\Big)$.
7. Generate $\sigma \leftarrow \mathsf{Sign}(\mathsf{SK}, \mathsf{VK} \| \widehat{c}\,)$, and set $c^* \coloneqq (\sigma, \mathsf{VK}, \widehat{c}\,)$.

Finally, $\mathcal{A}$ outputs a bit b and wins if $b = 0$.

$\mathbf{Game}^{\mathbf{1}}_{\boldsymbol{l_1}}$**:** This game is identical to $\text{Game}^0_{l_0}$

$\overline{\mathbf{Game}}^{\mathbf{1}}_{\boldsymbol{k}}\mathbf{(1 \leq \boldsymbol{k} \leq \boldsymbol{l_1})}$**:** This game is similar to Game^1_k, with the challenge ciphertext c^* computed by $\mathcal{C}$ as follows.

1. Generate $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$.
2. Draw $s \leftarrow\$\ \mathbb{Z}_q$, and compute $\overline{c}_0 \coloneqq g^s$.
3. For $j \coloneqq 1$ to $l_1 - k$, compute $\overline{c}_j \coloneqq H\Big(A^s_{\mathsf{HID}^1_j}\Big)$,
   $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}^1_j, \mathsf{VK} \| A^s_{\mathsf{HID}^1_j} \| m^*_1)$.
4. Set $\widetilde{m} \leftarrow\$\ \{0,1\}^{|\mathsf{VK}\|\overline{c}_0\|m^*_1|}$.

5. Compute $\overline{c}_{l_1-k+1} \coloneqq H\Big(A^s_{\mathsf{HID}^1_{l_1-k+1}}\Big)$, $c_{l_1-k+1} \leftarrow \mathsf{Enc}(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$.

6. For $j \coloneqq l_1 - k + 2$ to L, set $s_j \leftarrow\$ \mathbb{Z}_q$, and compute $\overline{c}_j \coloneqq H(g^{s_j})$, $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$.

7. Set $\widehat{c} \coloneqq \Big(\overline{c}_0, \Big(\overline{c}_{\pi(1)}, c_{\pi(1)}\Big), \ldots, \Big(\overline{c}_{\pi(L)}, c_{\pi(L)}\Big)\Big)$.

8. Generate $\sigma \leftarrow \mathsf{Sign}(\mathsf{SK}, \mathsf{VK}\|\widehat{c})$, and set $c^* \coloneqq (\sigma, \mathsf{VK}, \widehat{c})$.

At last, $\mathcal{A}$ outputs a bit b and wins if $b = 0$.

Game$^{\mathbf{1}}_{\boldsymbol{k}}$($\boldsymbol{0 \le k < l_1}$): This game is similar to $\overline{\text{Game}}^1_{k+1}$, but $\mathcal{C}$ computes the challenge ciphertext c^* as follows.

1. Generate $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$.

2. Draw $s \leftarrow\$ \mathbb{Z}_q$, and compute $\overline{c}_0 \coloneqq g^s$.

3. For $j \coloneqq 1$ to $l_1 - k$, compute $\overline{c}_j \coloneqq H\Big(A^s_{\mathsf{HID}^1_j}\Big)$, $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}^1_j, \mathsf{VK}\|A^s_{\mathsf{HID}^1_j}\|m^*_1)$.

4. Set $\widetilde{m} \leftarrow\$ \{0,1\}^{|\mathsf{VK}\|\overline{c}_0\|m^*_1|}$.

5. For $j \coloneqq l_1 - k + 1$ to L, set $s_j \leftarrow\$ \mathbb{Z}_q$, and compute $\overline{c}_j \coloneqq H(g^{s_j})$, $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$.

6. Set $\widehat{c} \coloneqq \Big(\overline{c}_0, \Big(\overline{c}_{\pi(1)}, c_{\pi(1)}\Big), \ldots, \Big(\overline{c}_{\pi(L)}, c_{\pi(L)}\Big)\Big)$.

7. Generate $\sigma \leftarrow \mathsf{Sign}(\mathsf{SK}, \mathsf{VK}\|\widehat{c})$, and set $c^* \coloneqq (\sigma, \mathsf{VK}, \widehat{c})$.

Finally, $\mathcal{A}$ outputs a bit b and wins if $b = 0$.

For $0 \le i_1 \le l_0$, $1 \le i_2 \le l_0$, $0 \le j_1 \le l_1$ and $1 \le j_2 \le l_1$, let $\mathsf{Adv}^{0,i_1}_{\mathcal{A},\Pi}$, $\overline{\mathsf{Adv}}^{0,i_2}_{\mathcal{A},\Pi}$, $\mathsf{Adv}^{1,j_1}_{\mathcal{A},\Pi}$ and $\overline{\mathsf{Adv}}^{1,j_2}_{\mathcal{A},\Pi}$ denote $\mathcal{A}$'s advantage of winning $\text{Game}^0_{i_1}$, $\overline{\text{Game}}^0_{i_2}$, $\text{Game}^1_{j_1}$ and $\overline{\text{Game}}^0_{j_2}$, respectively. In Lemma 3.4.9, we show that if the underlying one-time signature scheme and AIBE scheme are, respectively, (t, Q_D, ϵ_1)-SIG-SEU-secure and $(t, Q_U, Q_D, \epsilon_2)$-AIBE-IND-CCA-secure, then $\mathcal{A}$'s advantage of distinguishing Game^0_{h-1} from $\overline{\text{Game}}^0_h$ is at most $\epsilon_1 + \epsilon_2$. And, in Lemma 3.4.10, we show that if CDH problem is (t, ϵ_3)-hard in $\mathbb{G}$ and DDH problem is efficiently computable in $\mathbb{G}$, then $\mathcal{A}$ has at most ϵ_3 advantage in distinguishing $\overline{\text{Game}}_h^0$ from Game_h^0. Similarly, Lemma 3.4.11 and Lemma 3.4.12 states that under analogous conditions, $\mathcal{A}$'s advantages of distinguishing $\overline{\text{Game}}_{k+1}^1$ from Game_k^1, and Game_k^1 from $\overline{\text{Game}}_k^1$ is at most $\epsilon_1 + \epsilon_2$ and ϵ_3, respectively. Therefore,

$$\begin{aligned}\left|\mathsf{Adv}_{\mathcal{A},\Pi}^{0,0} - \mathsf{Adv}_{\mathcal{A},\Pi}^{1,0}\right| &\leq (\epsilon_1 + \epsilon_2 + \epsilon_3)(l_0 + l_1) \\ &\leq 2(\epsilon_1 + \epsilon_2 + \epsilon_3) L \\ &\leq 2(\epsilon_1 + \epsilon_2 + \epsilon_3) r \log\left(\frac{N}{r}\right). \qquad \blacksquare\end{aligned}$$

Lemma 3.4.9: *For $1 \leq h \leq l_0$, if the underlying one-time signature scheme Σ is (t, Q_D, ϵ_1)-SIG-SEU-secure and the anonymous identity-based encryption scheme Π' is $(t, Q_U, Q_D, \epsilon_2)$-AIBE-IND-CCA-secure, then $\mathcal{A}$'s advantage of distinguishing Game_{h-1}^0 from $\overline{\text{Game}}_h^0$ is at most $\epsilon_1 + \epsilon_2$. In other words,*

$$\left|\mathsf{Adv}_{\mathcal{A},\Pi}^{0,h-1} - \overline{\mathsf{Adv}}_{\mathcal{A},\Pi}^{0,h}\right| \leq (\epsilon_1 + \epsilon_2). \qquad \square$$

Proof. The proof is very similar to that of Lemma 3.4.5 and is therefore omitted. The only difference which we should be careful about is the new tag system of the ciphertext components. The challenger can trivially compute these tags as specified in the construction of Section 3.4.3 and attach them to the corresponding ciphertext components during the simulation. ■

Lemma 3.4.10: *For $1 \leq h \leq l_0$, if the computational Diffie-Hellman problem is (t, ϵ_3)-hard in $\mathbb{G}$ and the decisional Diffie-Hellman problem is efficiently computable in $\mathbb{G}$, then $\mathcal{A}$'s advantage of distinguishing $\overline{\text{Game}}_h^0$ from Game_h^0 is at most ϵ_3, i.e.,*

$$\left|\overline{\mathsf{Adv}}_{\mathcal{A},\Pi}^{0,h} - \mathsf{Adv}_{\mathcal{A},\Pi}^{0,h}\right| \leq \epsilon_3. \qquad \square$$

Proof. Let F be the event that $\mathcal{A}$ queries the random oracle H at the point $A^s_{\mathsf{HID}^0_{l_0-h+1}}$. By construction, it is clear that

$$\left|\overline{\mathsf{Adv}}^{0,h}_{\mathcal{A},\Pi} - \mathsf{Adv}^{0,h}_{\mathcal{A},\Pi}\right| \leq \Pr[F].$$

We want to show $\Pr[F] \leq \Pr[\text{CDH}] \leq \epsilon_3$. Assuming $\mathcal{A}$ can distinguish $\overline{\text{Game}}^0_h$ from Game^0_h, we build a PPT CDH adversary $\mathcal{B}$ which uses $\mathcal{A}$ as a subroutine. First, $\mathcal{B}$ gets a CDH problem instance $(\mathbb{G}, q, g, X = g^x, Y = g^y)$ as input from the CDH challenger. Then, $\mathcal{B}$ simulates the challenger's behavior in Game^0_h to $\mathcal{A}$ as follows.

Setup: $\mathcal{B}$ runs the *Setup* step as in Definition 3.3.3 except that it reuses the group $\mathbb{G}$ that it received from the CDH challenger and sets $A_{\mathsf{HID}^0_{l_0-h+1}} = Y$.

Phase 1: $\mathcal{B}$ replies to $\mathcal{A}$'s queries as follows.

Secret-key query i: These queries are handle as specified in Definition 3.3.3.

Decryption query (i, c): $\mathcal{B}$ distinguishes two cases. If the node (which is denoted by u for simplicity) with hierarchical identifier $\mathsf{HID}^0_{l_0-h+1}$ is not among the ancestors of the leaf node corresponding to the user i in the tree $\mathcal{T}$, then $\mathcal{B}$ just runs the Decrypt algorithm in Section 3.4.3. Otherwise, $\mathcal{B}$ still runs the Decrypt algorithm as in Section 3.4.3, but with one modification. That is, during Step 1, he skips the computation of the tag corresponding to node u. If this modified computation of the Decrypt algorithm yielded a valid message m, $\mathcal{B}$ simply returns that m. If not, $\mathcal{B}$ proceeds as follows.

1. Denote by sk_u the AIBE secret key of the node u.
2. Parse c as $(\sigma, \mathsf{VK}, \widehat{c} = (\overline{c}_0, (\overline{c}_1, c_1), \ldots, (\overline{c}_L, c_L)))$.
3. For $j := 1$ to L,

   a. Compute $m' := \mathsf{Dec}(\mathsf{MPK}', sk_u, c_j)$.

b. If $m' = \mathsf{VK}\|Z\|m$, $\mathsf{Vrfy}(\mathsf{VK}, \sigma, \mathsf{VK}\|\widehat{c}) = 1$, the DDH algorithm accepts $(g, \overline{c}_0, Y, Z)$, and $\overline{c}_j = H(Z)$, return m.

4. If Step 3 did not result in a valid m, return $\perp$ (as the original $\mathsf{Decrypt}$ algorithm would have returned).

Challenge: Given two equal length messages $m_0^*, m_1^* \in \mathcal{MSP}$ and two equal length sets of user identities $S_0^*, S_1^* \subseteq U$ with the restriction that $R_U \cap (S_0^* \cup S_1^*) = \emptyset$, $\mathcal{B}$ computes the challenge ciphertext c^* as follows.

1. Generate $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$.
2. Set $\overline{c}_0 := X$.
3. For $j := 1$ to $l_0 - h$, compute $\overline{c}_j := H\Big(X^{a_{\mathsf{HID}_j^0}}\Big)$,
   $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}_j^0, \mathsf{VK}\|X^{a_{\mathsf{HID}_j^0}}\|m_0^*)$.
4. Set $\widetilde{m} \leftarrow\$ \{0,1\}^{|\mathsf{VK}\|\overline{c}_0\|m_0^*|}$.
5. Compute $\overline{c}_{l_0-h+1} \leftarrow\$ \{0,1\}^\lambda$, $c_{l_0-h+1} \leftarrow \mathsf{Enc}(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$.
6. For $j := l_0 - h + 2$ to L, set $s_j \leftarrow\$ \mathbb{Z}_q$, and compute $\overline{c}_j := H(g^{s_j})$,
   $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$.
7. Set $\widehat{c} := \Big(\overline{c}_0, \Big(\overline{c}_{\pi(1)}, c_{\pi(1)}\Big), \ldots, \Big(\overline{c}_{\pi(L)}, c_{\pi(L)}\Big)\Big)$.
8. Generate $\sigma \leftarrow \mathsf{Sign}(\mathsf{SK}, \mathsf{VK}\|\widehat{c})$, and set $c^* := (\sigma, \mathsf{VK}, \widehat{c})$.

Phase 2: $\mathcal{B}$ handles *Phase 2* as in *Phase 1* with the same restrictions given in Definition 3.3.3. A slight complication is that, post-challenge, the adversary could try to reuse the $\overline{c}_0, \overline{c}_{\pi(1)}, \ldots, \overline{c}_{\pi(L)}$ components from c^*, but combine them with fresh $\tilde{c}_1, \ldots, \tilde{c}_L$ components for some message $\widehat{m}$ of her choice. If a ciphertext so crafted were submitted to the decryption oracle for user u, then $\mathcal{B}$ would invoke the special decryption process described in *Phase 1*, and would be unable to test whether $\overline{c}_j = H(Z)$. However, in order for all the other checks in Step 3b to go through, m' should be equal to $\hat{\mathsf{VK}}\|Z\|\widehat{m}$, for a Z such that (g, X, Y, Z)

is a DDH tuple. Clearly, at that point $\mathcal{B}$ could simply halt its computation, and output Z as its answer to its CDH challenge.

Guess: $\mathcal{A}$ outputs a guess b and $\mathcal{B}$ saves it.

When simulating the random oracle H to $\mathcal{A}$, $\mathcal{B}$ picks $R \leftarrow\$ \{0,1\}^\lambda$ as the result and responds consistently. If $\mathcal{A}$ ever makes a random oracle query Z such that the DDH algorithm accepts (g, X, Y, Z), $\mathcal{B}$ halts the computation and outputs Z as its solution to the CDH problem.

By construction, $\mathcal{A}$ can distinguish $\overline{\text{Game}}_h^0$ from Game_h^0 only if it queries the random oracle on g^{xy} or sends a decryption oracle query with a ciphertext component containing g^{xy}. In both cases, $\mathcal{B}$ suspends the computation and wins the CDH game. Therefore, $\mathcal{A}$'s advantage in distinguishing $\overline{\text{Game}}_h^0$ from Game_h^0 is at most ϵ_3. ■

Lemma 3.4.11: *For $0 \le k < l_1$, if the underlying one-time signature scheme Σ is (t, Q_D, ϵ_1)-SIG-SEU-secure and the anonymous identity-based encryption scheme Π' is $(t, Q_U, Q_D, \epsilon_2)$-AIBE-IND-CCA-secure, then $\mathcal{A}$'s advantage of distinguishing $\overline{Game}_{k+1}^1$ from $Game_k^1$ is at most $\epsilon_1 + \epsilon_2$. More precisely,*

$$\left|\overline{\mathsf{Adv}}_{\mathcal{A},\Pi}^{1,k+1} - \mathsf{Adv}_{\mathcal{A},\Pi}^{1,k}\right| \le (\epsilon_1 + \epsilon_2). \qquad \square$$

Proof. The argument is analogous to the proof of Lemma 3.4.9. ■

Lemma 3.4.12: *For $1 \le k \le l_1$, if the computational Diffie-Hellman problem is (t, ϵ_3)-hard in $\mathbb{G}$ and the decisional Diffie-Hellman problem is efficiently computable in $\mathbb{G}$, then $\mathcal{A}$'s advantage of distinguishing $Game_k^1$ from $\overline{Game}_k^1$ is at most ϵ_3, i.e.,*

$$\left|\mathsf{Adv}_{\mathcal{A},\Pi}^{1,k} - \overline{\mathsf{Adv}}_{\mathcal{A},\Pi}^{1,k}\right| \le \epsilon_3. \qquad \square$$

Proof. The argument is analogous to the proof of Lemma 3.4.10. ■

3.4.4 An Enhanced oABE-CCA-Secure Public-Key Construction in the Standard Model

In this section, we augment the construction in Section 3.4.3 so that its security can be proven in the standard model under the decisional Diffie-Hellman assumption using techniques from [83]. The key ingredient of this modification is the "trapdoor test" of the strong twin computational Diffie-Hellman problem [29]. The definitions of the DDH and s2CDH problems are given in Section 2.2.2 and Section 2.2.3, respectively.

Let $\Pi' = (\mathsf{Setup}', \mathsf{Extract}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ be a weakly robust AIBE-IND-CCA-secure anonymous identity-based encryption scheme and $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ a strong existentially unforgeable one-time signature scheme. We construct an oABE-IND-CCA-secure outsider-anonymous broadcast encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with enhanced decryption in the standard model as follows. In this construction, $\mathbb{G} = \langle g \rangle$ denotes a group with prime order $q > 2^\lambda$ in which the decisional Diffie-Hellman problem is hard and g denotes a group generator.

$\mathsf{Setup}(1^\lambda, N)$: Obtain $(\mathsf{MPK}', \mathsf{MSK}') \leftarrow \mathsf{Setup}'(1^\lambda)$. For each node (with the hierarchical identifier HID) in $\mathcal{T}$, draw $a_{\mathsf{HID}}, b_{\mathsf{HID}}, c_{\mathsf{HID}}, d_{\mathsf{HID}} \leftarrow\$ \mathbb{Z}_q$, and compute

$$A_{\mathsf{HID}} := g^{a_{\mathsf{HID}}}, B_{\mathsf{HID}} := g^{b_{\mathsf{HID}}}, C_{\mathsf{HID}} := g^{c_{\mathsf{HID}}}, D_{\mathsf{HID}} := g^{d_{\mathsf{HID}}}.$$

Output MPK and MSK as

$$\mathsf{MPK} := \Big(\mathsf{MPK}', N, \mathbb{G}, g, \{A_{\mathsf{HID}}, B_{\mathsf{HID}}, C_{\mathsf{HID}}, D_{\mathsf{HID}}\}_{\mathsf{HID} \in \mathcal{T}}\Big)$$
$$\mathsf{MSK} := \Big(\mathsf{MSK}', \{a_{\mathsf{HID}}, b_{\mathsf{HID}}, c_{\mathsf{HID}}, d_{\mathsf{HID}}\}_{\mathsf{HID} \in \mathcal{T}}\Big).$$

$\mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)$: Let $\mathsf{HID}_i := (\mathsf{Root}, \mathsf{ID}_1, \ldots, \mathsf{ID}_n)$ be the hierarchical identifier associated with user i in the binary tree $\mathcal{T}$. For $k := 1$ to $n+1$, set $\overline{sk}_{i,k} := (a_{\mathsf{HID}_{i|k}}, b_{\mathsf{HID}_{i|k}}, c_{\mathsf{HID}_{i|k}}, d_{\mathsf{HID}_{i|k}})$, and compute $sk_{i,k} \leftarrow \mathsf{Extract}'(\mathsf{MPK}', \mathsf{MSK}', \mathsf{HID}_{i|k})$.

Output the secret key sk_i of user i as

$$sk_i := \Big(\big(\overline{sk}_{i,1}, sk_{i,1}\big), \ldots, \big(\overline{sk}_{i,n+1}, sk_{i,n+1}\big)\Big).$$

Encrypt(MPK, S, m): Generate $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda)$. Let Cover be the family of subtrees covering the set of receivers S according to the CS method. For each subtree T_j in Cover, let HID_j be the hierarchical identifier associated with the root of T_j. Let $l := |\mathsf{Cover}|$, $r := N - |S|$ and $L := \left\lfloor r \log\left(\frac{N}{r}\right)\right\rfloor$. Draw $s \leftarrow\!\!\$\ \mathbb{Z}_q$, and compute $\overline{c}_0 := g^s$. For $1 \le j \le l$, compute $\overline{c}_j := ((A_{\mathsf{HID}_j}^{\mathsf{VK}} B_{\mathsf{HID}_j})^s, (C_{\mathsf{HID}_j}^{\mathsf{VK}} D_{\mathsf{HID}_j})^s)$, $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}_j, \mathsf{VK}\|m)$. Set $\widetilde{m} \leftarrow\!\!\$\ \{0,1\}^{|\mathsf{VK}\|m|}$. For $l+1 \le j \le L$, set $s_{j,1}, s_{j,2} \leftarrow\!\!\$\ \mathbb{Z}_q$, and compute $\overline{c}_j := (g^{s_{j,1}}, g^{s_{j,2}})$, $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathtt{dummy}, \widetilde{m})$, where $\mathtt{dummy}$ is a special identifier used to obtain padding ciphertext components. Compute $\widehat{c}$ as

$$\widehat{c} := \Big(\overline{c}_0, \big(\overline{c}_{\pi(1)}, c_{\pi(1)}\big), \ldots, \big(\overline{c}_{\pi(L)}, c_{\pi(L)}\big)\Big),$$

where $\pi : \{1, \ldots, L\} \to \{1, \ldots, L\}$ is a random permutation.

Generate $\sigma \leftarrow \mathsf{Sign}(\mathsf{SK}, \mathsf{VK}\|\widehat{c})$, and output $c := (\sigma, \mathsf{VK}, \widehat{c})$.

Decrypt(MPK, sk_i, c): Parse the secret key sk_i as the tuple $\Big(\big(\overline{sk}_{i,1}, sk_{i,1}\big), \ldots, \big(\overline{sk}_{i,n+1}, sk_{i,n+1}\big)\Big)$ and the ciphertext c as $(\sigma, \mathsf{VK}, \widehat{c} = (\overline{c}_0, (\overline{c}_1, c_1), \ldots, (\overline{c}_L, c_L)))$.

1. For $k := 1$ to $n+1$,
   a. Parse $\overline{sk}_{i,k}$ as (a_k, b_k, c_k, d_k)
   b. Compute $tag_k := (\overline{c}_0^{\,a_k \mathsf{VK}}\ \overline{c}_0^{\,b_k}, \overline{c}_0^{\,c_k \mathsf{VK}}\ \overline{c}_0^{\,d_k})$
2. Check whether $\exists k \in [1, n+1]\ \exists j \in [1, L]$ such that $tag_k = \overline{c}_j$
   a. If suitable k, j exist, compute $m' := \mathsf{Decrypt}'(\mathsf{MPK}', sk_{i,k}, c_j)$.
   b. If m' can be parsed as $\mathsf{VK}\|m$ and $\mathsf{Vrfy}(\mathsf{VK}, \sigma, \widehat{c}) = 1$, return m.
   c. Otherwise, return $\perp$.

Remark 3.4.13. Notice that using a technique similar to the one given in Remark 3.4.7, we can reduce the tag-searching time in Step 2 of the $\mathsf{Decrypt}$ algorithm from $O(nL)$ to $O(n+L) = O(L)$.

The correctness of the above enhanced public-key construction in the CCA setting follows from the algebraic properties of the group $\mathbb{G}$ and the correctness of the underlying signature and AIBE schemes. The security of this construction is established in Theorem 3.4.14 below.

Theorem 3.4.14: *If the one-time signature scheme* $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is* (t, Q_D, ϵ_1)*-SIG-SEU-secure, the AIBE scheme* $\Pi' = (\mathsf{Setup}', \mathsf{Extract}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ *is* $(t, Q_U, Q_D, \epsilon_2)$*-AIBE-IND-CCA-secure, and DDH problem is* (t, ϵ_3)*-hard in* $\mathbb{G}$*, then the above construction is* $\left(t, Q_U, Q_D, 2\left(\epsilon_1 + \epsilon_2 + 2\left(\epsilon_3 + \frac{Q_D}{2^\lambda}\right)\right) r \log\left(\frac{N}{r}\right)\right)$*-oABE-IND-CCA-secure.* $\square$

Proof. The proof of this theorem follows almost the same structure as that of Theorem 3.4.8, with the exception that the tags are now created as described in Section 3.4.4. More specifically, as in the proof of Theorem 3.4.8, we again consider the following sequence of games between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$.

$$\text{Game}_0^0, \overline{\text{Game}}_1^0, \text{Game}_1^0, \ldots, \overline{\text{Game}}_{l_0}^0, \text{Game}_{l_0}^0 \equiv$$
$$\text{Game}_{l_1}^1, \overline{\text{Game}}_{l_1}^1, \ldots, \text{Game}_1^1, \overline{\text{Game}}_1^1, \text{Game}_0^1$$

During the *Challenge* step of the first game (Game_0^0), $\mathcal{A}$ receives an encryption of m_0^* for S_0^* and in the last game (Game_0^1), $\mathcal{A}$ receives an encryption of m_1^* for S_1^*.

Game_0^0 corresponds to the original game as described in Definition 3.3.3, when the challenge bit b^* is fixed to 0.

$\overline{\text{Game}}_h^0 (1 \le h \le l_0)$ is similar to Game_{h-1}^0, except that at position $j = l_0 - h + 1$, $\mathcal{C}$ pairs the correct tag $\overline{c}_j = ((A_{\mathsf{HID}_j}^{\mathsf{VK}} B_{\mathsf{HID}_j})^s, (C_{\mathsf{HID}_j}^{\mathsf{VK}} D_{\mathsf{HID}_j})^s)$ with an encryption c_j of a random string $\widetilde{m}$ of the same length of $\mathsf{VK} \| m_0^*$.

$\mathrm{Game}^0_h (1 \le h \le l_0)$ is similar to $\overline{\mathrm{Game}}^0_h$, but $\mathcal{C}$ computes the challenge ciphertext components for position $j = l_0 - h + 1$ as follows: to create tag $\overline{c}_{l_0-h+1}$, $\mathcal{C}$ uses a random value $s_j \leftarrow\$ \mathbb{Z}_q$.

The description of $\overline{\mathrm{Game}}^1_k (1 \le k \le l_1)$, and $\mathrm{Game}^1_k (0 \le k < l_1)$ is as above, where we replace m^*_0 with m^*_1.

For $0 \le i_1 \le l_0$, $1 \le i_2 \le l_0$, $0 \le j_1 \le l_1$ and $1 \le j_2 \le l_1$, let $\mathsf{Adv}^{0,i_1}_{\mathcal{A},\Pi}$, $\overline{\mathsf{Adv}}^{0,i_2}_{\mathcal{A},\Pi}$, $\mathsf{Adv}^{1,j_1}_{\mathcal{A},\Pi}$ and $\overline{\mathsf{Adv}}^{1,j_2}_{\mathcal{A},\Pi}$ denote $\mathcal{A}$'s advantage of winning $\mathrm{Game}^0_{i_1}$, $\overline{\mathrm{Game}}^0_{i_2}$, $\mathrm{Game}^1_{j_1}$ and $\overline{\mathrm{Game}}^0_{j_2}$, respectively.

The proof that $\mathcal{A}$'s advantage of distinguishing Game^0_{h-1} from $\overline{\mathrm{Game}}^0_h$ is at most $\epsilon_1 + \epsilon_2$ is essentially identical to that of Lemma 3.4.9.

The proof that $\mathcal{A}$'s advantage of distinguishing $\overline{\mathrm{Game}}^0_h$ from Game^0_h is at most $2\Big(\epsilon_3 + \frac{Q_D}{2^\lambda}\Big)$ (where ϵ_3 is the advantage of breaking the DDH assumption in $\mathbb{G}$) is essentially identical to that of Lemma 1 of [83].

Similarly, $\mathcal{A}$'s advantages of distinguishing $\overline{\mathrm{Game}}^1_{k+1}$ from Game^1_k, and Game^1_k from $\overline{\mathrm{Game}}^1_k$ are at most $\epsilon_1 + \epsilon_2$ and $2\Big(\epsilon_3 + \frac{Q_D}{2^\lambda}\Big)$, respectively. Therefore,

$$\begin{aligned}\Big|\mathsf{Adv}^{0,0}_{\mathcal{A},\Pi} - \mathsf{Adv}^{1,0}_{\mathcal{A},\Pi}\Big| &\le 2\Big(\epsilon_1 + \epsilon_2 + 2\Big(\epsilon_3 + \frac{Q_D}{2^\lambda}\Big)\Big)(l_0 + l_1)\\ &\le 2\Big(\epsilon_1 + \epsilon_2 + 2\Big(\epsilon_3 + \frac{Q_D}{2^\lambda}\Big)\Big)L\\ &\le 2\Big(\epsilon_1 + \epsilon_2 + 2\Big(\epsilon_3 + \frac{Q_D}{2^\lambda}\Big)\Big)r\log\Big(\frac{N}{r}\Big). \qquad \blacksquare\end{aligned}$$

3.4.5 An Enhanced oABE-IND-CCA-Secure Public-Key Construction with Shorter Ciphertexts

Below we sketch a variation of our techniques from Section 3.4.4 that results in an outsider-anonymous broadcast encryption scheme with ciphertext length $O(r)$. Unfortunately, this very compact ciphertext length comes at a price on the other parameters.

The idea is to combine the Dodis-Fazio [40] public-key extension of the subset difference method of Naor *et al.* [89] with a fully secure weakly robust anonymous hierarchical identity-based encryption scheme with constant ciphertext length such as [33, 34]. Following this approach, we would get the following system parameters.

Ciphertext Length: $O(r)$ AHIBE ciphertexts.

Public Key Length: $O(N \log N)$ public tags.

Secret Key Length: $O(\log^2 N)$ AHIBE secret keys and $O(N)$ secret tags.

Decryption Time: $O(N)$ tag computation/searching time and one AHIBE decryption attempt.

3.4.6 An Enhanced oABE-IND-CCA-Secure Private-Key Construction

The enhanced oABE-IND-CCA-secure public key constructions achieve a major performance gain in the Decrypt algorithm compared to the generic oABE-IND-CCA-secure construction, but it also changes the length of the master public key from $O(1)$ to $O(N)$. This increase in master public key length may not be a concern for many practical constructions, since the master public key can be stored as a static data file on a server on the Internet and also in users' computers. Still, for the private-key setting it is possible to accommodate storage-sensitive systems and attain constant key storage at the Center, while maintaining efficient decryption and logarithmic storage at the receivers.

In particular, recall from Section 2.5 that in the private-key setting, only the Center can broadcast messages to the receivers. Thus, the $O(N)$ information from which the tags for efficient decryption are created does not need to be published.

Therefore, this information can be compressed into $O(1)$ key storage using a standard trick based on any length-tripling pseudo-random number generator G (*cf. e.g.*, the SD method of Naor *et al.* [89]). In other words, the random exponents associated with the subtrees of $\mathcal{T}$ (*cf.* Section 3.4.3) are now pseudorandomly generated from a single seed, by repeated invocations of G on the left or right third of the result of the previous iteration, based on the path to the root of the subtree at hand. Finally, upon reaching the subtree root, the middle third of the pseudorandom output is used to generate the required exponent.

Chapter 4

Broadcast Steganography

4.1 Introduction

Simmons [104] introduced the cryptographic community to the problem of hidden communication with his famous *prisoners' dilemma*: Alice and Bob are in jail and can only talk in the presence of the jail warden Ward. Ward will not allow any encrypted communication, so Alice and Bob must hide their messages about an escape plan (the *hiddentext*) into innocent-looking communication (the *stegotext*) that Ward cannot distinguish from casual chatter (the *covertext*).

Modern cryptographic treatment of steganography began with Cachin's formalization in the information-security setting [26] and Hopper *et al.*'s in the complexity-theoretic one [67]. Since then, steganography has received regular attention by the cryptographic community. To a first approximation, existing solutions differ mostly in the degree of adversarial control that they can tolerate, and in the specific trade-off that they achieve among the main efficiency measures of transmission overhead, public/secret key storage, and encryption/decryption complexity.

Kiayias *et al.* [74] improved the efficiency of the steganographic protocol of [67] by replacing the use of a pseudorandom function family with the combination of a pseudorandom generator and a t-wise independent hash function. This approach was further refined in [75] to obtain a key-efficient steganographic system, where the gain stems from employing a novel *rejection sampling* method based on extractors.

Public-Key Steganography. The notion of steganography was extended to the public-key setting by von Ahn and Hopper [110], but they mostly focused on security against passive adversaries. A stronger security model (steganographic secrecy against adaptive chosen-covertext attacks or SS-IND-CCA) was defined by Backes and Cachin [9], but their constructions attained only an intermediate security notion, termed steganographic secrecy against publicly-detectable, replayable adaptive chosen-covertext attacks (SS-IND-PDR-CCA). Building upon the work of [9], Hopper [66] attained full SS-IND-CCA security under the Decisional Diffie-Hellman assumption in the standard model. In addition, Le and Kurosawa [80] suggested a weaker generalization of the model of [9], but with better efficiency than [66].

All steganographic constructions mentioned above assume that the communication channel can be modeled by an efficient covertext sampler that can be queried adaptively, in a black-box manner. Dedic *et al.* [38, 95] looked into communication bounds for stegosystems of this kind, while Lysyanskaya and Meyerovich [86] dealt with the case of imperfect channel oracle samplers.

From an operational standpoint, public-key steganography resembles the setting of public-key encryption: a participant with a public/secret key pair is able to receive covert messages (the hiddentexts) from another party, who only knows the public key. Unlike the case of public-key cryptography, however, it is assumed that the communication medium, called the *channel*, has a pre-determined distribution of possible "neutral content" (the covertexts). Furthermore, the encoded hiddentexts (the stegotexts), are required to be indistinguishable from the covertexts of the communication channel.

A common approach to realize public-key stegosystems is the *encrypt-then-embed*

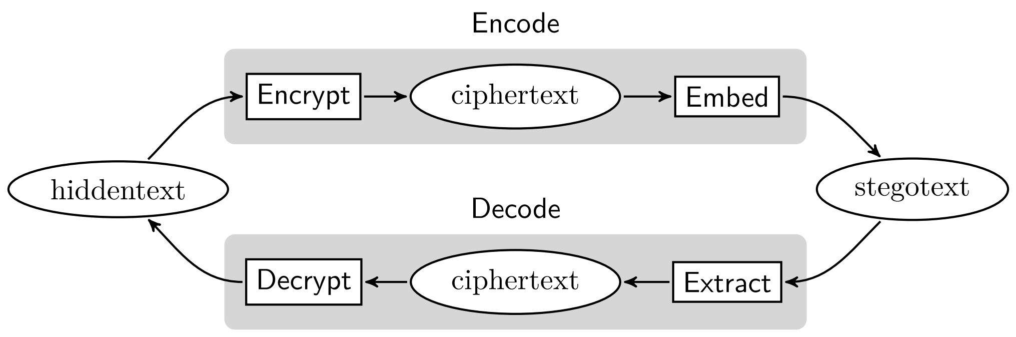

Figure 4.1: The encrypt-then-embed paradigm underlying steganography.

paradigm [9, 66, 67, 110], depicted in Figure 4.1. At a high level, encoding is accomplished by first encrypting the hiddentext using a public-key cryptosystem, and then implanting the resulting ciphertext in the stegotext using an embedding function. The decoding process develops similarly, but in the reverse direction. Based on the security properties of the underlying cryptosystem and embedding function, one obtains stegosystems with a variety of security guarantees.

4.2 Contributions

The state of the art steganographic protocols only allow covert communication between two parties. In certain applications, however, one-to-many covert communication is desired. As a simple example, assume that there is a country where freedom of speech is curtailed. Also assume that in this country, there is an activist who does not agree with the events occurring in his neighborhood. Consequently, he decides to use social media channels to broadcast messages about the events to his followers. If he broadcast the messages in the clear, he would run into trouble with the authorities. If he used broadcast encryption, he would raise suspicion due to its gibberish-looking distribution and again run into trouble with the authorities. What he needs is a mechanism to covertly send messages to his followers. Although steganography does allow covert communication, using a point-to-point steganographic protocol with its key shared among the followers does not help the activist much because it does not allow him to revoke compromised followers. What he really needs is a mechanism to *covertly broadcast* messages that also allows him to dynamically select the set of recipients for each broadcast message.

Broadcast Steganography. We formalize the above notion of steganography in this chapter under the title *broadcast steganography* (BS).[1] Intuitively, broadcast steganography enables a sender to communicate covertly with a dynamically designated set of receivers, so that authorized recipients correctly recover the original content, while unauthorized users and outsiders remain *unaware* of the covert communication. To construct broadcast steganography, we employ the encrypt-then-embed paradigm that underpins most steganographic constructions [9, 66, 67, 110]. Realizing this approach, however, requires solving several technical problems.

The first issue is that, in broadcast encryption, the receiver set is included explicitly in the ciphertext as part of its header (*e.g.*, [12, 18, 22, 40–42, 49, 50, 54, 64, 89, 118]). This is a non-starter for steganography, which intrinsically requires that the existence of any data in the channel be concealed. To address this issue, we turn to anonymous broadcast encryption, a notion introduced by Barth *et al.* [10] with the goal of keeping the identities of the authorized receivers anonymous.

The second hurdle is that the encrypt-then-embed paradigm requires the underlying encryption functionality to have *pseudorandom* ciphertexts. This property so far had not been considered in the broadcast encryption literature, and none of the existing constructions support it natively. Interestingly, attaining pseudorandom ciphertexts requires implicitly that the identities of the recipients be unintelligible *in the view of outsiders* (pseudorandomness of the ciphertext clearly cannot hold in the view of the recipients). This condition ties back directly to the previous issue, but in a weaker form, as recipient anonymity is only required to hold against outsiders. As it turns

[1]This result has also been published at the Cryptographer's Track at the RSA Conference—CT-RSA 2014 [47]

$$\begin{tikzcd}
 & \boxed{\text{AnoBE\$}} \arrow[dl] \arrow[dr] & & \\
\boxed{\text{AnoBE}} \arrow[dr] & \boxed{\text{BE}} & \boxed{\text{oABE\$}} \arrow[r, squiggly] \arrow[dl] & \boxed{\text{BS}} \arrow[dll, bend left] \\
 & \boxed{\text{oABE}} \arrow[u] & &
\end{tikzcd}$$

Figure 4.2: Relations between broadcast encryption (BE), (outsider) anonymous broadcast encryption (AnoBE/oABE), and broadcast steganography (BS). A straight arrow means that one notion implies the other, while the curly arrow denotes our black-box constructions from oABE$ to BS (*cf.* Section 4.5). (To avoid cluttering the figure, relations implied by transitivity are omitted.)

out, the notion of outsider-anonymous broadcast encryption that we presented in Chapter 3 provides a relaxation of full anonymity of exactly this sort. As mentioned in Section 3.2, oABE trades some degree of anonymity for better efficiency: whereas all known fully-anonymous broadcast encryption schemes [10, 83] have ciphertexts *linear* in the number of receivers, our constructions given in Section 3.4 obtain *sublinear* ciphertext length, though they do not necessarily guarantee that authorized users will learn no information about other members of the receiver set. Unfortunately, none of our oABE contructions attain pseudorandom ciphertexts required for the encrypt-then-embed paradigm given in Figure 4.1.

In light of the above observations, we put forth and realize a new broadcast encryption variant that we term *outsider-anonymous broadcast encryption with pseudorandom ciphertexts* (oABE$). oABE$ enables a black-box construction of broadcast steganography via the encrypt-then-embed paradigm. Realizing an efficient oABE$ scheme requires non-trivial enhancements to the oABE construction of Section 3.4.4, for it entails resolving the apparent tension between the ciphertext pseudorandom property and the ciphertext redundancy introduced by common approaches to CCA security [20, 43]. Our solution harmonizes these requirements using a novel Pedersen-like encapsulation mechanism. The definition of an encapsulation mechanism is presented in Section 2.4.1. Figure 4.2 shows how oABE$ relates to other broadcast communication protocols.

Table 4.1 shows the parameters of our oABE$-based BS schemes. In this table, Type-1 channels are the most general, and are modeled as stateful probabilistic oracles whose output distribution *may* depend on past samples. Type-2 channels are slightly more restrictive as they assume history independence, and can then be modeled as efficiently sampleable document distributions, in other words, efficiently computable randomized functions.

Applications of Broadcast Steganography. The combination of stealth and revocation capabilities offered by broadcast steganography enables defenses against insider threats in anti-censorship systems, intelligence scenarios, and other domains that rely on covert communication [87, 109].

For a military example, consider a camp where each soldier has an army smartphone, on which they receive weather forecast, unclassified news and other information in the clear. Suppose that headquarters suspect that a group of officials are conspiring to commit treachery, and decides to carry out an undercover investigation to confirm the identities of the traitors. Conventional broadcast encryption does not suffice to protect the transmission channel to the soldiers involved in the investigation of the traitors, because the selective exclusion of the conspirators from the communication would already put them on notice. Broadcast steganography, instead, would allow delivery of instructions to the investigating parties without risking alerting the traitors to the investigation.

For a civil rights scenario, an activist/blogger may want to hide her commentary into innocent-looking image postings to social media services (*e.g.*, Instagram or Weibo). Because censorship authorities may infiltrate among the activist's followers, the ability of broadcast steganography to authorize/deauthorize recipients at a fine grain would enable the blogger to revoke the infiltrator and prevent him from recovering the hiddentext, *without him noticing that he has been singled out.*

Table 4.1: The parameters of our black-box broadcast steganography schemes. N is the total number of users and r is the number of revoked users.

Scheme	**MPK Length**	***sk* Length**	***s* Length**	**Security Model**	**Channel Type**
Cons. 1 of Sect. 4.5.1	$O(N)$	$O(\log N)$	$O(r\log(\frac{N}{r}))$	CHA, Adaptive $\mathcal{A}$	1
Cons. 2 of Sect. 4.5.1	$O(N)$	$O(\log N)$	$O(r\log(\frac{N}{r}))$	PDR-CCA, Adaptive $\mathcal{A}$	1
Cons. of Sect. 4.5.2	$O(N)$	$O(\log N)$	$O(r\log(\frac{N}{r}))$	CCA, Adaptive $\mathcal{A}$	2

Remark 4.2.1. Work of von Ahn *et al.* [111] and Chandran *et al.* [31] introduced stealthiness to the setting of secure function evaluation, originating the notion of *covert two-party/multi-party computation.* Covert protocols allow parties to carry out distributed computations in a way that hides their very *intent* of taking part in the protocol: that is, unless *all* parties actively participate, nobody can detect that protocol messaging had been initiated (and aborted). This capability supports stealthy coordination between mutually mistrustful parties and enables fascinating applications like covert authentication [111] and co-spy detection [31]. However, it does not imply efficient covert dissemination of information to a chosen subset of (mostly passive) receivers, which is the main focus of this chapter.

Organization. We formally introduce the setting and the security models of broadcast steganography in Section 4.3. Next, we introduce the formal security models of oABE$ in Section 4.4.1 and a secure construction of oABE$ in Section 4.4.2. Finally, in Section 4.5, we devise efficient oABE$-based BS schemes at varying security levels with sublinear stegotexts secure in the standard model against adaptive adversaries.

4.3 Formal Model

4.3.1 Setting of BS

We now formally define the setting of broadcast steganography. Please refer to Section 2.4.5 for the formal definitions of documents, stegotexts, and channels.

Definition 4.3.1 (BS Setting): A broadcast steganography scheme, associated with a universe of users $U = [1, N]$, a message space $\mathcal{MSP}$, and a channel $\mathfrak{C}_h$ on a set of documents Σ, is a tuple of algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encode}, \mathsf{Decode})$ defined as follows.

$(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, N)$: Setup takes the security parameter 1^λ and the number of users in the system N as inputs and outputs the master public key MPK and the master secret key MSK.

$sk_i \leftarrow \mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)$: Given the master public key MPK, the master secret key MSK, and a user $i \in U$, KeyGen generates a secret key sk_i for user i.

$s \leftarrow \mathsf{Encode}(\mathsf{MPK}, S, h, m)$: Encode takes the master public key MPK, a set of receivers $S \subseteq U$, a channel history $h \in \Sigma^*$, and a message $m \in \mathcal{MSP}$ as inputs and outputs a stegotext $s \in \Sigma^*$ from the support of $\mathfrak{C}_h^l$ for some $l = poly(|m|)$.

$m/\bot := \mathsf{Decode}(\mathsf{MPK}, sk_i, s)$: Given the master public key MPK, a secret key sk_i, and a stegotext $s \in \Sigma^*$, Decode either outputs a message $m \in \mathcal{MSP}$ or the failure symbol $\bot$. We assume that Decode is deterministic.

Correctness. For every $S \subseteq U$, $i \in S$, legal channel history $h \in \Sigma^*$, and $m \in \mathcal{MSP}$, if $(\mathsf{MPK}, \mathsf{MSK})$ is output by $\mathsf{Setup}(1^\lambda, N)$ and sk_i is generated by $\mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)$, then it must be the case that

$$\mathsf{Decode}(\mathsf{MPK}, sk_i, \mathsf{Encode}(\mathsf{MPK}, S, h, m)) = m,$$

except with negligible probability in the security parameter λ. $\Diamond$

Remark 4.3.2. In contrast to the setting of regular steganography [66], which we presented in Definition 2.4.17, the setting of broadcast steganography requires that the Decode algorithm works without receiving the channel history h corresponding to the stegotext s as an input. This is crucial for an efficient broadcast steganography scheme, because requiring that authorized users feed the Decode algorithm with the same h that was used by the sender entails a level of coordination that is unrealistic in a broadcast setting. Our definition also applies to channels whose samples do not depend on h at all, as Encode may simply ignore h.

4.3.2 Security of BS

In broadcast encryption, the adversary's goal is to learn something about the message encrypted within a given ciphertext despite not having a valid decryption key. In broadcast steganography, the adversary's goal is to detect the *presence* of a message in a given covertext without a valid decoding key. In either case, one may consider multiple levels of security, according to the amount of power afforded to the attacker. We discuss below three models of security for broadcast steganography schemes, followed by formal definitions later in this section.

BS-IND-CHA Security. This model is called the *chosen-hidentext attack* security, and it is the weakest model of security for a broadcast steganography scheme. Analogous to the chosen-plaintext attack in broadcast encryption, the adversary in this context is only allowed to corrupt users by gaining their secret keys.

BS-IND-PDR-CCA Security. This is called the *publicly-detectable replayable chosen-covertext attack* security. In this model, the adversary is additionally given access to a decoding oracle through which he can obtain the hiddentext (if any) in any covertext s of his choice, as recovered by any honest user i of their choice, subject to the following restriction: after receiving the challenge covertext s^* for the set of recipients S^*, the adversary is not allowed to query the decoding oracle with a user index i and a covertext s such that $i \in S^*$ and $s \equiv_{\mathsf{MPK}} s^*$, where $\equiv_{\mathsf{MPK}}$ is an arbitrary *BS compatible relation* whose definition is given in Definition 4.3.3 below.

Definition 4.3.3 (BS Compatible Relation): Denote by $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encode}, \mathsf{Decode})$ a BS scheme. A binary relation on stegotexts of Π induced by a master public key MPK is called a *BS compatible relation* (denoted by $\equiv_{\mathsf{MPK}}$) if for any two stegotexts s_1, s_2 encoded under sets of receivers S_1, S_2, respectively, the following requirements are satisfied.

1. If $s_1 \equiv_{\mathsf{MPK}} s_2$ then for any $i_1 \in S_1$ and $i_2 \in S_2$, it must be the case that

$\mathsf{Decode}(\mathsf{MPK}, sk_{i_1}, s_1) = \mathsf{Decode}(\mathsf{MPK}, sk_{i_2}, s_2)$ except with negligible probability in the security parameter λ.

2. There exists a PPT algorithm that only takes MPK, s_1, and s_2 and determines whether $s_1 \equiv_{\mathsf{MPK}} s_2$.

BS-IND-CCA Security. This is the *chosen-covertext attack* model of security. A BS-IND-CCA adversary has the same capabilities from the BS-IND-PDR-CCA model of security, but the restriction for the decoding queries is now lifted. Specifically, the only covertext that the adversary is not allowed to submit to the decoding oracle with a user index $i \in S^*$ is the challenge covertext s^* itself.

We now formally define the BS-IND-CCA model of security as a game played between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Next, we show how this BS-IND-CCA game can be tweaked to obtain the games corresponding to the BS-IND-PDR-CCA and BS-IND-CHA security models.

Definition 4.3.4 (BS-IND-CCA Game): For a given BS scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encode}, \mathsf{Decode})$, the BS-IND-CCA game, which is played between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, is defined as follows.

Setup: $\mathcal{C}$ runs $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, N)$ and gives $\mathcal{A}$ the resulting master public key MPK, keeping the master secret key MSK to itself. $\mathcal{C}$ also initializes the set of revoked users R_U to be empty.

Phase 1: $\mathcal{A}$ adaptively issues queries of the following types.

Secret-key query $\boldsymbol{i}$: $\mathcal{A}$ requests the secret key of a user $i \in U$. $\mathcal{C}$ runs $sk_i \leftarrow \mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)$, adds i to R_U, and sends sk_i to $\mathcal{A}$.

Decoding query $(\boldsymbol{i}, \boldsymbol{s})$: $\mathcal{A}$ issues a decoding query on a user index $i \in U$ and a covertext $s \in \Sigma^*$. $\mathcal{C}$ computes $\mathsf{Decode}(\mathsf{MPK}, \mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i), s)$ and gives the result to $\mathcal{A}$.

Challenge: $\mathcal{A}$ gives $\mathcal{C}$ a message $m^* \in \mathcal{MSP}$, a legal history $h \in \Sigma^*$, and a set of user identities $S^* \subseteq U$ with the restriction that $S^* \cap R_U = \emptyset$. $\mathcal{C}$ picks a random bit $b^* \in \{0,1\}$ and generates the challenge s^* depending on it as follows. If $b^* = 0$, then $\mathcal{C}$ encodes m^* into a stegotext s^* for the receiver set S^*, more precisely $s^* \leftarrow \mathsf{Encode}(\mathsf{MPK}, S^*, h, m^*)$. Otherwise, $\mathcal{C}$ sample s^* as a covertext of equal length, *i.e.*, $s^* \leftarrow_\$ \mathfrak{C}_h^{l^*}$ for $l^* = |\mathsf{Encode}(\mathsf{MPK}, S^*, h, m^*)|/\sigma$. At the end, $\mathcal{C}$ gives s^* to $\mathcal{A}$.

Phase 2: The interaction between $\mathcal{A}$ and $\mathcal{C}$ in this phase is similar to *Phase 1* with two restrictions as given below.

Secret-key query *i*: $i \notin S^*$.

Decoding query (i, s): If $i \in S^*$, then $s \neq s^*$.

Guess: $\mathcal{A}$ outputs a guess $b \in \{0,1\}$ and wins if $b = b^*$.

The adversary $\mathcal{A}$ is called a BS-IND-CCA adversary and $\mathcal{A}$'s advantage is defined as

$$\mathsf{Adv}_{\mathcal{A},\Pi}^{\text{BS-IND-CCA}} := \left|\Pr[b = b^*] - \tfrac{1}{2}\right|,$$

where the probability is over the random coins used by $\mathcal{A}$ and $\mathcal{C}$. $\lozenge$

Definition 4.3.5 (BS-IND-CCA Security): A BS scheme Π is (t, Q_U, Q_D, ϵ)-BS-IND-CCA-secure if for any t-time BS-IND-CCA adversary making at most Q_U adaptive secret-key queries and at most Q_D adaptive decoding queries, it must be the case that $\mathsf{Adv}_{\mathcal{A},\Pi}^{\text{BS-IND-CCA}} \leq \epsilon$. $\lozenge$

By restricting the kind of decoding queries allowed in *Phase 2* of the BS-IND-CCA game above, we can obtain the BS-IND-PDR-CCA game. Specifically, the adversary now cannot issue any decoding query (i, s) such that $i \in S^*$ and $s \equiv_{\mathsf{MPK}} s^*$

for some BS compatible relation $\equiv_{\mathsf{MPK}}$. The adversary $\mathcal{A}$ in this game is called a BS-IND-PDR-CCA adversary and $\mathcal{A}$'s advantage is defined as

$$\mathsf{Adv}_{\mathcal{A},\Pi}^{\text{BS-IND-PDR-CCA}} := \left|\Pr[b = b^*] - \tfrac{1}{2}\right|,$$

where the probability is over the random coins used by $\mathcal{A}$ and $\mathcal{C}$.

Definition 4.3.6 (BS-IND-PDR-CCA Security): A BS scheme Π is (t, Q_U, Q_D, ϵ)-BS-IND-PDR-CCA-secure with respect to some BS compatible relation $\equiv_{\mathsf{MPK}}$ if for any t-time BS-IND-PDR-CCA adversary making at most Q_U adaptive secret-key queries and at most Q_D adaptive decoding queries, we have $\mathsf{Adv}_{\mathcal{A},\Pi}^{\text{BS-IND-PDR-CCA}} \leq \epsilon$.$\Diamond$

The BS-IND-CHA game is defined similar to the BS-IND-CCA game, with the restriction that the adversary is not allowed to issue any decoding queries during *Phase 1* and *Phase 2*. The adversary is still allowed to issue secret-key queries.

Definition 4.3.7 (BS-IND-CHA Security): A BS scheme Π is (t, Q_U, ϵ)-BS-IND-CHA-secure if Π is $(t, Q_U, 0, \epsilon)$-BS-IND-CCA-secure. $\Diamond$

4.4 Anonymity and Pseudorandomness in Broadcast Encryption

In Chapter 3, we presented the notion of outsider-anonymous broadcast encryption, a security model for BE whose goal is to hide the identities of the intended receivers of a broadcast ciphertext from unauthorized users. As outlined in Section 4.1, a crucial technical step to realize broadcast steganography is combining receiver anonymity with pseudorandomness of broadcast ciphertexts (*cf.* Section 4.5).

This section develops the notion of *outsider-anonymous broadcast encryption with pseudorandom ciphertexts*, and presents an efficient construction secure in the standard model under a stronger security model, *outsider anonymity and ciphertext pseudorandomness against chosen-ciphertext attacks* (oABE\$-IND-CCA). Being an extension of oABE, the setting of oABE\$ is identical to the one given in Definition 3.3.1. However, the security model of oABE\$ is quite different as it has to take into account the pseudorandomness of the ciphertexts. In Section 4.4.1 below, we formally present the security model of oABE\$.

4.4.1 Security of oABE\$

We now present three models of security for outsider-anonymous broadcast encryption with pseudorandom ciphertexts: oABE\$-IND-CPA, oABE\$-IND-PDR-CCA, and oABE\$-IND-CCA. These three models of security loosely corresponds to BS-IND-CHA, BS-IND-PDR-CCA, and BS-IND-CCA models of security for broadcast steganography. At a high level, these security models require that for any message m^* and set of recipients S^*, no PPT adversary $\mathcal{A}$ can distinguish between an actual encryption of m^* intended for the set S^*, and a truly random string of the same length as an encryption of m^* for S^*, so long as $\mathcal{A}$ does not possess the secret key of any user in S^*. In Section 4.4.2, we present an oABE\$-IND-CCA-secure construction.

First, we define the oABE\$-IND-CCA model of security as a game played between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Later, we show how BS-IND-CCA game can be modified to obtain the games corresponding to the oABE\$-IND-PDR-CCA and oABE\$-IND-CPA security models.

Definition 4.4.1 (oABE\$-IND-CCA Game): For a given oABE\$ scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$, the oABE\$-IND-CCA game, played between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, is defined as follows.

Setup: $\mathcal{C}$ runs $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, N)$ and gives $\mathcal{A}$ the resulting master public key MPK, keeping the master secret key MSK to itself. $\mathcal{C}$ also initializes the set of revoked users R_U to be empty.

Phase 1: $\mathcal{A}$ adaptively issues queries of the following types.

> **Secret-key query i:** $\mathcal{A}$ requests the secret key of a user $i \in U$. $\mathcal{C}$ runs $sk_i \leftarrow \mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)$, adds i to R_U, and sends sk_i to $\mathcal{A}$.
>
> **Decryption query (i, c):** $\mathcal{A}$ sends a decryption query on a user $i \in U$ and a ciphertext $c \in \mathcal{CSP}$. $\mathcal{C}$ computes $\mathsf{Decrypt}(\mathsf{MPK}, \mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i), c)$ and gives the result to $\mathcal{A}$.

Challenge: $\mathcal{A}$ gives $\mathcal{C}$ a message $m^* \in \mathcal{MSP}$ and a set of user identities $S^* \subseteq U$ with the restriction that $S^* \cap R_U = \emptyset$. $\mathcal{C}$ picks a random bit $b^* \in \{0,1\}$ and generates the challenge ciphertext c^* depending on it: if $b^* = 0$, then $c^* \leftarrow \mathsf{Encrypt}(\mathsf{MPK}, S^*, m^*)$, else $c^* \leftarrow_\$ \{0,1\}^{l^*}$ for $l^* = |\mathsf{Encrypt}(\mathsf{MPK}, S^*, m^*)|$. The challenge ciphertext c^* is then given to $\mathcal{A}$.

Phase 2: The interaction between $\mathcal{A}$ and $\mathcal{C}$ in this phase is similar to *Phase 1* with two restrictions as given below.

> **Secret-key query i:** $i \notin S^*$.
>
> **Decoding query (i, s):** If $i \in S^*$, then $c \neq c^*$.

Guess: $\mathcal{A}$ outputs a guess $b \in \{0,1\}$ and wins if $b = b^*$.

The adversary $\mathcal{A}$ is called an oABE\$-IND-CCA adversary and $\mathcal{A}$'s advantage is

$$\mathsf{Adv}_{\mathcal{A},\Pi}^{\text{oABE\$-IND-CCA}} := \left|\Pr[b = b^*] - \tfrac{1}{2}\right|,$$

where the probability is over the random coins used by $\mathcal{A}$ and $\mathcal{C}$. $\Diamond$

Observe that the key difference of the above definition from the one given in Definition 3.3.2 is in the *Challenge* step, where the challenger either returns the encryption of m^* or a random bit-string with appropriate length.

Definition 4.4.2 (oABE\$-IND-CCA Security): An oABE\$ scheme Π is (t, Q_U, Q_D, ϵ)-oABE\$-IND-CCA-secure if for any t-time oABE\$-IND-CCA adversary making at most Q_U adaptive secret-key queries and at most Q_D adaptive decryption queries, we have $\mathsf{Adv}_{\mathcal{A},\Pi}^{\text{oABE\$-IND-CCA}} \leq \epsilon$. $\Diamond$

The oABE\$-IND-PDR-CCA game is obtained by restricting the adversary during *Phase 2* of the oABE\$-IND-CCA game from submitting any decoding query (i, c) such that $i \in S^*$ and $c \equiv_{\mathsf{MPK}} c^*$, where $\equiv_{\mathsf{MPK}}$ is an arbitrary oABE\$ compatible relation of the oABE\$ scheme. The definition of an oABE\$ compatible relation is given in Definition 4.4.3 below. The adversary $\mathcal{A}$ in this game is called an oABE\$-IND-PDR-CCA adversary and $\mathcal{A}$'s advantage is defined as

$$\mathsf{Adv}_{\mathcal{A},\Pi}^{\text{oABE\$-IND-PDR-CCA}} := \left|\Pr[b = b^*] - \tfrac{1}{2}\right|.$$

Definition 4.4.3 (oABE\$ Compatible Relation): Let an oABE\$ scheme be denoted by $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$. A binary relation on ciphertexts of Π induced by a master public key MPK is called an *oABE\$ compatible relation* (denoted by $\equiv_{\mathsf{MPK}}$) if for any two ciphertexts c_1, c_2 encrypted under sets of receivers S_1, S_2, respectively, the following requirements are met.

1. If $c_1 \equiv_{\mathsf{MPK}} c_2$ then for any $i_1 \in S_1$ and $i_2 \in S_2$, it must be the case that $\mathsf{Decrypt}(\mathsf{MPK}, sk_{i_1}, c_1) = \mathsf{Decrypt}(\mathsf{MPK}, sk_{i_2}, c_2)$ except with negligible probability in the security parameter λ.

2. There exists a PPT algorithm that only takes MPK, c_1, and c_2 and determines whether $c_1 \equiv_{\mathsf{MPK}} c_2$.

Definition 4.4.4 (oABE\$-IND-PDR-CCA Security): An oABE\$ scheme Π is (t, Q_U, Q_D, ϵ)-oABE\$-IND-PDR-CCA-secure with respect to an oABE\$ compatible relation $\equiv_{\mathsf{MPK}}$ if for any t-time oABE\$-IND-PDR-CCA adversary making at most Q_U

adaptive secret-key queries and at most Q_D adaptive decryption queries, it must be the case that $\mathsf{Adv}_{\mathcal{A},\Pi}^{\text{oABE\$-IND-PDR-CCA}} \leq \epsilon$. ◊

By restricting the adversary in the oABE\$-IND-CCA game from submitting any decoding queries during *Phase 1* and *Phase 2*, we obtain the oABE\$-IND-CPA game. The adversary is still allowed to issue secret-key queries.

Definition 4.4.5 (oABE\$-IND-CPA Security): An oABE\$ scheme Π is (t, Q_U, ϵ)-oABE\$-IND-CPA-secure if Π is $(t, Q_U, 0, \epsilon)$-oABE\$-IND-CCA-secure. ◊

4.4.2 An oABE\$-IND-CCA-Secure Construction

Our oABE\$-IND-CCA-secure construction builds on the enhanced oABE-IND-CCA-secure construction of Section 3.4.4. At a high level, the approach of the oABE-IND-CCA-secure construction is to

1. "bundle" multiple ciphertexts of an anonymous identity-based encryption scheme (*e.g.*, [2, 24, 51]) into a single oABE ciphertext,
2. "tag" each AIBE ciphertext to enable the decryptor to efficiently locate the component compatible with her decryption key,
3. and "seal" everything together with a one-time signature to thwart CCA attacks.

To attain pseudorandom oABE ciphertexts in our oABE\$-IND-CCA-secure construction, we will start with an anonymous identity-based encryption scheme with *pseudorandom ciphertexts* (AIBE\$) like the one of [4]. Additionally, we will use an *entropy-smoothing* hash function [69] to hide the structure in the ciphertext tags. The definition of an entropy-smoothing hash function is given in Section 2.3.1.

These adjustments do not suffice because the presence of the one-time signature introduces additional structure in the oABE ciphertext of the construction from Section 3.4.4. To get around this, we substitute one-time signatures with MACs (implemented via pseudorandom functions) and employ a variant of an encapsulation mechanism [20, 43] with an additional pseudorandom property as described below. The definition of an encapsulation mechanism is given in Section 2.4.1.

Let p, q be primes such that $2^\lambda < q < 2^{\lambda+1}$ and $p = 2q + 1$, and g be a square modulo p. Denote by $\mathbb{G} = \langle g \rangle$ the group of quadratic residues modulo p. To "pack" quadratic residues into λ bits, we will use rejection sampling along with the following well-known $\mathbb{G}$–$\mathbb{Z}_q$ bijection (*cf. e.g.*, [66]).

$$\mathsf{mp}(a) = \begin{cases} a & \text{if } a \leq q \\ p - a & \text{otherwise} \end{cases} \qquad \mathsf{mp}^{-1}(b) = \begin{cases} b & \text{if } b^{\frac{p-1}{2}} \equiv 1 \bmod p \\ p - b & \text{otherwise} \end{cases}$$

Figures 4.3 to 4.5 show the $\mathsf{SetupCom}$, Commit, and Open functionalities, respectively, of our Pedersen-like [91] encapsulation mechanism over $\mathbb{G}$. The hiding requirement follows from the hiding properties of standard Pedersen commitments, coupled with the observation that $\mathsf{mp}(\cdot)$ is a bijection. Relaxed binding follows from the discrete logarithm assumption in $\mathbb{G}$, again similarly to standard Pedersen commitments. A novel feature of our encapsulation mechanism is that the distribution of commitments com induced by the $\mathsf{Commit}(\mathsf{PK})$ algorithm is *uniform over* $\{0,1\}^\lambda$, and hence the relaxed commitment scheme of Figures 4.3 to 4.5 has *pseudorandom commitments.*

Now we present our oABE\$ construction. Let $\Pi' = (\mathsf{Setup}', \mathsf{Extract}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ be an AIBE\$-IND-CCA-secure anonymous identity-based encryption scheme having pseudorandom ciphertexts with expansion ℓ (*i.e.*, $|\mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{ID}, m)| = \ell(|m|)$). Let $F : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\lambda$ be a pseudorandom function and let $\mathcal{H}_{es} = \{\mathbb{G}^2 \to \{0,1\}^\lambda\}$ be an entropy-smoothing hash function family. Using these primitives, we construct an oABE\$-IND-CCA-secure outsider-anonymous broadcast encryption scheme with pseudorandom ciphertexts $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as shown in Figures 4.6 to 4.9.

```
Algorithm: $\mathsf{SetupCom}(1^\lambda)$
1  repeat
2      generate a random $(\lambda+1)$-bit prime $p$
3      $q := \frac{(p-1)}{2}$
4  until $q$ is prime
5  repeat
6      $x \leftarrow\$ \mathbb{Z}_p^*$
7  until $x \neq \pm 1 \bmod p$
8  $g := x^2$
9  $y, z \leftarrow\$ \mathbb{Z}_q$
10 $g_{\mathsf{com}} := g^y$
11 $h_{\mathsf{com}} := g^z$
12 $\mathsf{PK} := (p, q, g, g_{\mathsf{com}}, h_{\mathsf{com}})$
13 return $\mathsf{PK}$
```

Figure 4.3: SetupCom algorithm of our Pedersen-like encapsulation mechanism.

```
Algorithm: $\mathsf{Commit}(\mathsf{PK})$
1  parse $\mathsf{PK}$ as $(p, q, g, g_{\mathsf{com}}, h_{\mathsf{com}})$
2  $\hat{k} \leftarrow\$ \{0,1\}^\lambda$
3  repeat
4      $\tilde{k} \leftarrow\$ \mathbb{Z}_q$
5      $\mathsf{com} := \mathsf{mp}(g_{\mathsf{com}}^{\hat{k}} h_{\mathsf{com}}^{\tilde{k}})$
6  until $\mathsf{com} < 2^\lambda$
7  $\mathsf{decom} := (\hat{k}, \tilde{k})$
8  return $(\hat{k}, \mathsf{com}, \mathsf{decom})$
```

Figure 4.4: Commit algorithm of our Pedersen-like encapsulation mechanism.

```
Algorithm: $\mathsf{Open}(\mathsf{PK}, \mathsf{com}, \mathsf{decom})$
1  parse $\mathsf{PK}$ as $(p, q, g, g_{\mathsf{com}}, h_{\mathsf{com}})$
2  parse $\mathsf{decom}$ as $(\hat{k}, \tilde{k})$
3  if $\mathsf{com} = \mathsf{mp}(g_{\mathsf{com}}^{\hat{k}} h_{\mathsf{com}}^{\tilde{k}})$ then
4      return $\hat{k}$
5  end
6  return $\perp$
```

Figure 4.5: Open algorithm of our Pedersen-like encapsulation mechanism.

```
Algorithm: $\mathsf{Setup}(1^\lambda, N)$
1  $(\mathsf{MPK}', \mathsf{MSK}') \leftarrow \mathsf{Setup}'(1^\lambda)$
2  $\mathsf{PK}'' \leftarrow \mathsf{SetupCom}(1^\lambda)$
3  parse $\mathsf{PK}''$ as $(p, q, g, g_{\mathsf{com}}, h_{\mathsf{com}})$
4  $H \leftarrow\$ \mathcal{H}_{es}$
5  // $\mathsf{Fam}$ denotes the set of all the subtrees in $\mathcal{T}$
6  for $j := 1$ to $|\mathsf{Fam}|$ do
7      // $T_j$ denotes the subtree in $\mathsf{Fam}$ indexed by $j$
8      // $\mathsf{HID}_j$ denotes the hierarchical identifier of the root of $T_j$
9      $a_{1,\mathsf{HID}_j}, a_{2,\mathsf{HID}_j}, b_{1,\mathsf{HID}_j}, b_{2,\mathsf{HID}_j} \leftarrow\$ \mathbb{Z}_q$
10     $A_{1,\mathsf{HID}_j} := g^{a_{1,\mathsf{HID}_j}}$
11     $A_{2,\mathsf{HID}_j} := g^{a_{2,\mathsf{HID}_j}}$
12     $B_{1,\mathsf{HID}_j} := g^{b_{1,\mathsf{HID}_j}}$
13     $B_{2,\mathsf{HID}_j} := g^{b_{2,\mathsf{HID}_j}}$
14 end
15 $\mathsf{MPK} := (\mathsf{MPK}', \mathsf{PK}'', H, N, \{A_{i,\mathsf{HID}_j}, B_{i,\mathsf{HID}_j}\}_{i\in\{1,2\}, j\in[1,|\mathsf{Fam}|]})$
16 $\mathsf{MSK} := (\mathsf{MSK}', \{a_{i,\mathsf{HID}_j}, b_{i,\mathsf{HID}_j}\}_{i\in\{1,2\}, j\in[1,|\mathsf{Fam}|]})$
17 return $(\mathsf{MPK}, \mathsf{MSK})$
```

Figure 4.6: Setup algorithm of our oABE$-IND-CCA-secure construction.

```
Algorithm: $\mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)$
1  parse $\mathsf{MSK}$ as $(\mathsf{MSK}', \{a_{i,\mathsf{HID}_j}, b_{i,\mathsf{HID}_j}\}_{i\in\{1,2\}, j\in[1,|\mathsf{Fam}|]})$
2  parse $\mathsf{MPK}$ as $(\mathsf{MPK}', \mathsf{PK}'', H, N, \{A_{i,\mathsf{HID}_j}, B_{i,\mathsf{HID}_j}\}_{i\in\{1,2\}, j\in[1,|\mathsf{Fam}|]})$
3  // $\mathsf{HID}_i$ denotes the hierarchical identifier of leaf $i$ in $\mathcal{T}$
4  for $z := 1$ to $n + 1$ do
5      $\overline{sk}_{i,z} := (a_{1,\mathsf{HID}_{i|z}}, a_{2,\mathsf{HID}_{i|z}}, b_{1,\mathsf{HID}_{i|z}}, b_{2,\mathsf{HID}_{i|z}})$
6      $sk_{i,z} \leftarrow \mathsf{Extract}'(\mathsf{MPK}', \mathsf{MSK}', \mathsf{HID}_{i|z})$
7  end
8  $sk_i := ((\overline{sk}_{i,1}, sk_{i,1}), \ldots, (\overline{sk}_{i,n+1}, sk_{i,n+1}))$
9  return $sk_i$
```

Figure 4.7: KeyGen algorithm of our oABE$-IND-CCA-secure construction.

```
**Algorithm**: $\mathsf{Encrypt}(\mathsf{MPK}, S, m)$
1  **parse** $\mathsf{MPK}$ **as** $(\mathsf{MPK}', \mathsf{PK}'', H, N, \{A_{i,\mathsf{HID}_j}, B_{i,\mathsf{HID}_j}\}_{i\in\{1,2\}, j\in[1,|\mathsf{Fam}|]})$
2  **parse** $\mathsf{PK}''$ **as** $(p, q, g, g_{\mathsf{com}}, h_{\mathsf{com}})$
3  $r := N - |S|$
4  $L := \left\lfloor r \log\left(\frac{N}{r}\right)\right\rfloor$
5  $(\hat{k}, \mathsf{com}, \mathsf{decom}) \leftarrow \mathsf{Commit}(\mathsf{PK}'')$
6  **repeat**
7      $s \leftarrow\$ \mathbb{Z}_q$
8      $\bar{c}_0 := \mathsf{mp}(g^s)$
9  **until** $\bar{c}_0 < 2^\lambda$
10 // $\mathsf{Cov}$ denotes the set of subtrees covering $S$ in $\mathcal{T}$
11 **for** $j := 1$ **to** $|\mathsf{Cov}|$ **do**
12     // $T_j$ denotes the subtree in $\mathsf{Cov}$ indexed by $j$
13     // $\mathsf{HID}_j$ denotes the hierarchical identifier of the root of $T_j$
14     $\bar{c}_j := H((A_{1,\mathsf{HID}_j}^{\mathsf{com}} A_{2,\mathsf{HID}_j})^s, (B_{1,\mathsf{HID}_j}^{\mathsf{com}} B_{2,\mathsf{HID}_j})^s)$
15     $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}_j, \mathsf{com}\|m\|\mathsf{decom})$
16 **end**
17 **for** $j := |\mathsf{Cov}| + 1$ **to** $L$ **do**
18     $\bar{c}_j \leftarrow\$ \{0,1\}^\lambda$
19     $c_j \leftarrow\$ \{0,1\}^{\ell(3\lambda+1+|m|)}$
20 **end**
21 $\hat{c} := \bar{c}_0\|\bar{c}_1\|c_1\|\ldots\|\bar{c}_L\|c_L$
22 $\sigma := F(\hat{k}, \hat{c})$
23 $c := \sigma\|\hat{c}\|\mathsf{com}$
24 **return** $c$
```

Figure 4.8: Encrypt algorithm of our oABE\$-IND-CCA-secure construction.

```
Algorithm: $\mathsf{Decrypt}(\mathsf{MPK}, sk_i, c)$
 1 parse $\mathsf{MPK}$ as $(\mathsf{MPK}', \mathsf{PK}'', H, N, \{A_{i,\mathsf{HID}_j}, B_{i,\mathsf{HID}_j}\}_{i\in\{1,2\},j\in[1,|\mathsf{Fam}|]})$
 2 parse $\mathsf{PK}''$ as $(p, q, g, g_{\mathsf{com}}, h_{\mathsf{com}})$
 3 parse $sk_i$ as $((\overline{sk}_{i,1}, sk_{i,1}), \ldots, (\overline{sk}_{i,n+1}, sk_{i,n+1}))$
 4 parse $c$ as $\sigma\|\hat{c}\|\mathsf{com}$
 5 parse $\hat{c}$ as $\bar{c}_0\|\bar{c}_1\|c_1\|\ldots\|\bar{c}_L\|c_L$
 6 $\tilde{c}_0 := \mathsf{mp}^{-1}(\bar{c}_0)$
 7 for $z := 1$ to $n+1$ do
 8     parse $\overline{sk}_{i,z}$ as $(\tilde{a}_{1,z}, \tilde{a}_{2,z}, \tilde{b}_{1,z}, \tilde{b}_{2,z})$
 9     $tag_z := H(\tilde{c}_0^{\tilde{a}_{1,z}\mathsf{com}+\tilde{a}_{2,z}}, \tilde{c}_0^{\tilde{b}_{1,z}\mathsf{com}+\tilde{b}_{2,z}})$
10 end
11 if $\exists z \in [1, n+1]\ \exists j \in [1, L] : tag_z = \bar{c}_j$ then
12     $m' := \mathsf{Decrypt}'(\mathsf{MPK}', sk_{i,z}, c_j)$
13     if $m' \neq \perp$ then
14         parse $m'$ as $\overline{\mathsf{com}}\|m\|\mathsf{decom}$
15         if $\overline{\mathsf{com}} = \mathsf{com}$ then
16             $\hat{k} := \mathsf{Open}(\mathsf{PK}'', \mathsf{com}, \mathsf{decom})$
17             if $\hat{k} \neq \perp \wedge \sigma = F(\hat{k}, \hat{c})$ then
18                 return $m$
19             end
20         end
21     end
22 end
23 return $\perp$
```

Figure 4.9: Decrypt algorithm of our oABE\$-IND-CCA-secure construction.

To attain sublinear ciphertexts, we follow the approach of of the construction given in Section 3.4.4, which is based on the public-key extension of the subset cover framework [40,89]. We arrange the $N = 2^n$ users in a perfect binary tree with N leaves, and assign to each user (using AIBE\$) $n+1$ decryption keys, corresponding to all the nodes in the path to its designated leaf (Line 6 of KeyGen). Each oABE\$ ciphertext consists of multiple AIBE\$ components. For efficient decryption, AIBE\$ components are tagged using a twin-DH-based [29] technique reminiscent of Section 3.4.4 (Line 14 of Encrypt) so that recipients can single out which AIBE\$ component to decrypt, and with which key (Lines 7–11 and 12 of Decrypt). Throughout Encrypt, we make sure that each piece in an oABE\$ ciphertext looks random, with the use of rejection sampling (Lines 6–9), entropy smoothing (Line 14), dummy components (Lines 17–20), and pseudorandom MACs (Line 23) in place of one-time signature. Forgoing signatures introduce a complication, as the input to the PRF appears to depend on the PRF key $\hat{k}$: the $\bar{c}_j$ values and the oABE\$ components c_j's computed in Lines 14 and 15 are derived from com and decom, which correlate with $\hat{k}$. We solve this circularity by mediating the occurrence of $\hat{k}$ in the ciphertext via our Pedersen-like encapsulation mechanism scheme given in Figures 4.3 to 4.5.

Theorem 4.4.6: *If the PRF F is (t_1, ϵ_1)-hard, the AIBE\$ scheme Π' is $(t_2, Q_U, Q_D, \epsilon_2)$-AIBE\$-IND-CCA-secure, the family of hash functions $\mathcal{H}_{es}$ is a (t_3, ϵ_3)-entropy-smoothing, and DDH is (t_4, ϵ_4)-hard in $\mathbb{G}$, then the construction given in Figures 4.6 to 4.9 is $\Big(t_1 + t_2 + t_3 + t_4, Q_U, Q_D, \Big(\epsilon_1 + \epsilon_2 + \epsilon_3 + 2\Big(\epsilon_4 + \frac{Q_D}{q}\Big)\Big) r \log\Big(\frac{N}{r}\Big)\Big)$-oABE\$-IND-CCA-secure, where N and r are the total number of users the number of revoked users, respectively.* □

Proof. We organize our proof as a sequence of games, $\text{Game}_0, \overline{\text{Game}}_1, \text{Game}_1, \ldots, \overline{\text{Game}}_l, \text{Game}_l$, between the oABE\$-IND-CCA adversary $\mathcal{A}$ and the challenger $\mathcal{C}$, where l denotes the cardinality of the coverset Cov induced by the set of authorized receivers S^* chosen by $\mathcal{A}$ during the *Challenge* step of the oABE\$-IND-CCA game.

During the *Challenge* step of the first game (Game$_0$), $\mathcal{A}$ receives an encryption of m^* for S^*, and in the last game (Game$_l$), $\mathcal{A}$ receives a uniformly random bit-string of the appropriate length as the challenge ciphertext.

Game$_0$: This game corresponds to the game given in Definition 4.4.2 when the challenge bit b^* is fixed to 0. The interaction between $\mathcal{A}$ and $\mathcal{C}$ during *Setup*, *Phase 1*, *Phase 2*, and *Guess* steps follows exactly as specified in Definition 4.4.2. During the *Challenge* step, $\mathcal{A}$ gives $\mathcal{C}$ a message $m^* \in \mathcal{MSP}$ and a set of user identities $S^* \subseteq U$ with the restriction that $S^* \cap R_U = \emptyset$, where R_U is the set of users that $\mathcal{A}$ corrupted during *Phase 1*. $\mathcal{C}$ computes the challenge ciphertext c^*, which is subsequently sent to $\mathcal{A}$, as follows.

```
1  parse MPK as $(\mathsf{MPK}', \mathsf{PK}'', H, N, \{A_{i,\mathsf{HID}_j}, B_{i,\mathsf{HID}_j}\}_{i\in\{1,2\}, j\in[1,|\mathsf{Fam}|]})$
2  parse PK'' as $(p, q, g, g_{\mathsf{com}}, h_{\mathsf{com}})$
3  $r := N - |S^*|$
4  $L := \left\lfloor r \log\left(\frac{N}{r}\right)\right\rfloor$
5  $(\hat{k}, \mathsf{com}, \mathsf{decom}) \leftarrow \mathsf{Commit}(\mathsf{PK}'')$
6  repeat
7      $s \leftarrow\$ \mathbb{Z}_q$
8      $\bar{c}_0 := \mathsf{mp}(g^s)$
9  until $\bar{c}_0 < 2^\lambda$
10 for $j := 1$ to $l$ do
11     $\bar{c}_j := H((A_{1,\mathsf{HID}_j}^{\mathsf{com}} A_{2,\mathsf{HID}_j})^s, (B_{1,\mathsf{HID}_j}^{\mathsf{com}} B_{2,\mathsf{HID}_j})^s)$
12     $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}_j, \mathsf{com}\|m^*\|\mathsf{decom})$
13 for $j := l+1$ to $L$ do
14     $\bar{c}_j \leftarrow\$ \{0,1\}^\lambda$
15     $c_j \leftarrow\$ \{0,1\}^{\ell(3\lambda+1+|m^*|)}$
16 $\hat{c} := \bar{c}_0\|\bar{c}_1\|c_1\|\ldots\|\bar{c}_L\|c_L$
17 $\sigma := F(\hat{k}, \hat{c})$
```

```
18 $c^* := \sigma \| \hat{c} \| \mathsf{com}$
```

$\overline{\text{Game}}_h(1 \leq h \leq l)$: This game is similar to Game_{h-1}, but, when creating c^*, $\mathcal{C}$ replaces Lines 10–15 with the following.

```
1′ for $j := 1$ to $l - h$ do
2′     $\overline{c}_j := H((A^{\mathsf{com}}_{1,\mathsf{HID}_j} A_{2,\mathsf{HID}_j})^s, (B^{\mathsf{com}}_{1,\mathsf{HID}_j} B_{2,\mathsf{HID}_j})^s)$
3′     $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}_j, \mathsf{com} \| m^* \| \mathsf{decom})$
4′ $\overline{c}_{l-h+1} := H((A^{\mathsf{com}}_{1,\mathsf{HID}_{l-h+1}} A_{2,\mathsf{HID}_{l-h+1}})^s, (B^{\mathsf{com}}_{1,\mathsf{HID}_{l-h+1}} B_{2,\mathsf{HID}_{l-h+1}})^s)$
5′ $c_{l-h+1} \leftarrow\$ \{0,1\}^{\ell(3\lambda+1+|m^*|)}$
6′ for $j := l - h + 2$ to $L$ do
7′     $\overline{c}_j \leftarrow\$ \{0,1\}^{\lambda}$
8′     $c_j \leftarrow\$ \{0,1\}^{\ell(3\lambda+1+|m^*|)}$
```

$\text{Game}_h(1 \leq h \leq l)$: This game is similar to $\overline{\text{Game}}_h$, but, when creating c^*, $\mathcal{C}$ replaces Lines 4′–8′ with the following lines.

```
1″ for $j := l - h + 1$ to $L$ do
2″     $\overline{c}_j \leftarrow\$ \{0,1\}^{\lambda}$
3″     $c_j \leftarrow\$ \{0,1\}^{\ell(3\lambda+1+|m^*|)}$
```

For $0 \leq i_1 \leq l$ and $1 \leq i_2 \leq l$ let $\mathsf{Adv}^{i_1}_{\mathcal{A},\Pi}$ and $\overline{\mathsf{Adv}}^{i_2}_{\mathcal{A},\Pi}$ denote $\mathcal{A}$'s advantage in winning Game_{i_1} and $\overline{\text{Game}}_{i_2}$, respectively. In Lemma 4.4.7, we show that if the underlying PRF F is (t_1, ϵ_1)-hard and the AIBE\$ scheme Π' is $(t_2, Q_U, Q_D, \epsilon_2)$-AIBE\$-IND-CCA-secure, then $\mathcal{A}$'s advantage of distinguishing Game_{h-1} from $\overline{\text{Game}}_h$ is at most $\epsilon_1 + \epsilon_2$. In Lemma 4.4.8, we show that if $\mathcal{H}_{es}$ is an (t_2, ϵ_2)-entropy-smoothing family of hash functions and DDH problem is (t_4, ϵ_4)-hard in $\mathbb{G}$, then $\mathcal{A}$ has at most $\epsilon_3 + 2\left(\epsilon_4 + \frac{Q_D}{q}\right)$ advantage in distinguishing $\overline{\text{Game}}_h$ from Game_h. Therefore,

$$\begin{aligned}\left|\mathsf{Adv}^{0}_{\mathcal{A},\Pi} - \mathsf{Adv}^{l}_{\mathcal{A},\Pi}\right| &\le \left(\epsilon_1 + \epsilon_2 + \epsilon_3 + 2\left(\epsilon_4 + \frac{Q_D}{q}\right)\right) l \\ &\le \left(\epsilon_1 + \epsilon_2 + \epsilon_3 + 2\left(\epsilon_4 + \frac{Q_D}{q}\right)\right) L \\ &\le \left(\epsilon_1 + \epsilon_2 + \epsilon_3 + 2\left(\epsilon_4 + \frac{Q_D}{q}\right)\right) r \log\left(\frac{N}{r}\right). \qquad \blacksquare\end{aligned}$$

Lemma 4.4.7: *For $1 \le h \le l$, if the underlying pseudorandom function F is (t_1, ϵ_1)-hard and the anonymous identity-based encryption scheme with pseudorandom ciphertexts Π' is $(t_2, Q_U, Q_D, \epsilon_2)$-AIBE\$-IND-CCA-secure, then $\mathcal{A}$'s advantage of distinguishing Game$_{h-1}$ from $\overline{\text{Game}}_h$ is at most $\epsilon_1 + \epsilon_2$, i.e.,*

$$\left|\mathsf{Adv}^{h-1}_{\mathcal{A},\Pi} - \overline{\mathsf{Adv}}^{\,h}_{\mathcal{A},\Pi}\right| \le \epsilon_1 + \epsilon_2. \qquad \square$$

Proof. We build a PPT adversary $\mathcal{B}$ that internally runs the oABE\$-IND-CCA game with the adversary $\mathcal{A}$ in order to gain advantage in the AIBE\$-IND-CCA game with the challenger $\mathcal{C}'$. We denote the secret-key oracle and the decryption oracle of $\mathcal{C}'$ by $\mathcal{O}'_U(\cdot)$ and $\mathcal{O}'_D(\cdot,\cdot)$, respectively. After receiving the master public key MPK' of the AIBE\$ scheme from $\mathcal{C}'$, $\mathcal{B}$ executes the oABE\$-IND-CCA game with $\mathcal{A}$ as follows.

Setup: $\mathcal{B}$ generates MPK, which he eventually sends to $\mathcal{A}$, by executing Lines 2–15 of the Setup algorithm given in Figure 4.6. $\mathcal{B}$ also keeps the exponents $\{a_{i,\mathsf{HID}_j}, b_{i,\mathsf{HID}_j}\}_{i\in\{1,2\},j\in[1,|\mathsf{Fam}|]}$ to himself and initializes the set of revoked users R_U to be empty.

Phase 1: $\mathcal{B}$ replies to $\mathcal{A}$'s queries as follows.

Secret-key query *i*: $\mathcal{B}$ computes the secret key sk_i by executing lines Lines 2–9 of the KeyGen algorithm of Figure 4.7 with one modification: during

Line 6, $\mathcal{B}$ sets $sk_{i,z} \leftarrow \mathcal{O}'_{sk}(\mathsf{HID}_{i|z})$. Next, after adding user i to R_U, $\mathcal{B}$ sends the secret key sk_i to $\mathcal{A}$.

Decryption query (i, c): $\mathcal{B}$ computes the hierarchical identifier HID_i of leaf i in $\mathcal{T}$ and proceeds as follows.

```
1  **parse** $\mathsf{MPK}$ **as** $(\mathsf{MPK}', \mathsf{PK}'', H, N, \{A_{i,\mathsf{HID}_j}, B_{i,\mathsf{HID}_j}\}_{i\in\{1,2\},j\in[1,|\mathsf{Fam}|]})$
2  **parse** $\mathsf{PK}''$ **as** $(p, q, g, g_{\mathsf{com}}, h_{\mathsf{com}})$
3  **parse** $c$ **as** $\sigma\|\hat{c}\|\mathsf{com}$
4  **parse** $\hat{c}$ **as** $\bar{c}_0\|\bar{c}_1\|c_1\|\ldots\|\bar{c}_L\|c_L$
5  $\tilde{c}_0 := \mathsf{mp}^{-1}(\bar{c}_0)$
6  **for** $z := 1$ **to** $n+1$ **do**
7      $\tilde{a}_{1,z} := a_{1,\mathsf{HID}_{i|z}}, \tilde{a}_{2,z} := a_{2,\mathsf{HID}_{i|z}}\ \tilde{b}_{1,z} := b_{1,\mathsf{HID}_{i|z}}, \tilde{b}_{2,z} := b_{2\mathsf{HID}_{i|z}}$
8      $tag_z := H(\tilde{c}_0^{\tilde{a}_{1,z}\mathsf{com}+\tilde{a}_{2,z}}, \tilde{c}_0^{\tilde{b}_{1,z}\mathsf{com}+\tilde{b}_{2,z}})$
9  **if** $\exists z \in [1, n+1]\ \exists j \in [1, L] : tag_z = \bar{c}_j$ **then**
10     $m' := \mathcal{O}'_d(\mathsf{HID}_{i|k}, c_j)$
11     **if** $m' \neq \perp$ **then**
12         **parse** $m'$ **as** $\overline{\mathsf{com}}\|m\|\mathsf{decom}$
13         **if** $\overline{\mathsf{com}} = \mathsf{com}$ **then**
14             $\hat{k} := \mathsf{Open}(\mathsf{PK}'', \mathsf{com}, \mathsf{decom})$
15             **if** $\hat{k} \neq \perp \wedge \sigma = F(\hat{k}, \hat{c})$ **then**
16                 **return** $m$
17 **return** $\perp$
```

Challenge: After receiving from $\mathcal{A}$ a message $m^* \in \mathcal{MSP}$ and a set of user identities $S^* \subseteq U$ with the restriction that $S^* \cap R_U = \emptyset$, $\mathcal{B}$ picks $(\hat{k}, \mathsf{com}, \mathsf{decom}) \leftarrow \mathsf{Commit}(\mathsf{PK}'')$ and sets

$$\mathsf{ID}' := \mathsf{HID}_{l-h+1}, \quad m' := \mathsf{com}\|m^*\|\mathsf{decom}.$$

Next, $\mathcal{B}$ sends the identity ID' and the messages m' as the challenge query to $\mathcal{C}'$. Then, $\mathcal{C}'$ picks a random bit $b' \in \{0,1\}$ and generates the challenge ciphertext c' depending on it: if $b' = 0$, then $c' \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{ID}', \mathsf{com}\|m^*\|\mathsf{decom})$, else $c' \leftarrow_\$ \{0,1\}^{\ell(|m'|)}$, and returns c' to $\mathcal{B}$. Finally, $\mathcal{B}$ computes the challenge ciphertext c^*, which is eventually sent to $\mathcal{A}$, as follows.

```
parse MPK as $(\mathsf{MPK}', \mathsf{PK}'', H, N, \{A_{i,\mathsf{HID}_j}, B_{i,\mathsf{HID}_j}\}_{i\in\{1,2\}, j\in[1,|\mathsf{Fam}|]})$
parse PK'' as $(p, q, g, g_{\mathsf{com}}, h_{\mathsf{com}})$
$r \coloneqq N - |S^*|$
$L \coloneqq \left\lfloor r \log\left(\frac{N}{r}\right)\right\rfloor$
$(\hat{k}, \mathsf{com}, \mathsf{decom}) \leftarrow \mathsf{Commit}(\mathsf{PK}'')$
repeat
    $s \leftarrow_\$ \mathbb{Z}_q$
    $\bar{c}_0 \coloneqq \mathsf{mp}(g^s)$
until $\bar{c}_0 < 2^\lambda$
for $j \coloneqq 1$ to $l - h$ do
    $\bar{c}_j \coloneqq H((A^{\mathsf{com}}_{1,\mathsf{HID}_j} A_{2,\mathsf{HID}_j})^s, (B^{\mathsf{com}}_{1,\mathsf{HID}_j} B_{2,\mathsf{HID}_j})^s)$
    $c_j \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', \mathsf{HID}_j, \mathsf{com}\|m^*\|\mathsf{decom})$
$\bar{c}_{l-h+1} \coloneqq H((A^{\mathsf{com}}_{1,\mathsf{HID}_{l-h+1}} A_{2,\mathsf{HID}_{l-h+1}})^s, (B^{\mathsf{com}}_{1,\mathsf{HID}_{l-h+1}} B_{2,\mathsf{HID}_{l-h+1}})^s)$
$c_{l-h+1} \coloneqq c'$
for $j \coloneqq l - h + 2$ to $L$ do
    $\bar{c}_j \leftarrow_\$ \{0,1\}^\lambda$
    $c_j \leftarrow_\$ \{0,1\}^{\ell(3\lambda+1+|m^*|)}$
$\hat{c} \coloneqq \bar{c}_0\|\bar{c}_1\|c_1\|\ldots\|\bar{c}_L\|c_L$
$\sigma \coloneqq F(\hat{k}, \hat{c})$
$c^* \coloneqq \sigma\|\hat{c}\|\mathsf{com}$
```

Phase 2: $\mathcal{B}$ replies to $\mathcal{A}$'s queries as follows.

Secret-key query i: These queries are handled similarly to *Phase 1*, with the usual restriction that $\mathcal{A}$ does *not* invoke a secret-key query i such that $i \in S^*$.

Decryption query (i, c): $\mathcal{B}$ replies to $(i, c = \sigma \| \hat{c} \| \mathsf{com})$, according to one of the following cases.

- If $c = c^*$ and $i \notin S^*$, then $\mathcal{B}$ proceeds as in *Phase 1.* (Note that in this case $\mathcal{B}$'s output will be $\perp$, as it should be.)
- If $c = c^*$, and $i \in S^*$, $\mathcal{B}$ just rejects the decryption query since $\mathcal{A}$ is submitting an invalid query.
- If $c \neq c^*$ and $i \notin S^*$, then $\mathcal{B}$ proceeds as in *Phase 1.*
- If $c \neq c^*$ and $i \in S^*$, then $\mathcal{B}$ computes HID_i and proceeds as follows.
  - ◇ If for all $z = 1$ to $n + 1$, it is the case that $\mathsf{HID}_{i|z} \neq \mathsf{HID}_{l-h+1}$, then $\mathcal{B}$ proceeds as in *Phase 1.* Observe that the condition $\forall z \in [1, n+1] : \mathsf{HID}_{i|z} \neq \mathsf{HID}_{l-h+1}$ ensures that the decryption query that $\mathcal{B}$ will make to its challenger $\mathcal{C}'$ in the process of responding to $\mathcal{A}$'s query is allowed.
  - ◇ If $\exists\ z \in [1, n+1]$ such that $\mathsf{HID}_{i|z} = \mathsf{HID}_{l-h+1}$, and c' does not appear among the ciphertext components of c, then again $\mathcal{B}$ proceeds as in *Phase 1.* Observe that the condition that c does not contain c' ensures that also in this case the decryption query that $\mathcal{B}$ will make to its challenger $\mathcal{C}'$ in the process of responding to $\mathcal{A}$'s query is also allowed.
  - ◇ If $\exists\ z \in [1, n+1]$ such that $\mathsf{HID}_{i|z} = \mathsf{HID}_{l-h+1}$, but c' appears among the ciphertext components of c, then $\mathcal{B}$ outputs $\perp$. Arguing that this (*i.e.*, $\perp$) is the real reply that $\mathcal{A}$ would get in either $\overline{\text{Game}_h}$ or Game_h requires some care, but can be done along the

lines of the proofs of [20] and [43]. In a nutshell, the issue is the circularity in the PRF usage: in generating the σ component of the ciphertext, $F(\hat{k}, \cdot)$ is computed over $\hat{c}$, which includes ciphertext components that contain com and decom, which in turn correlate with $\hat{k}$. The reason this circularity does not break the argument is that the appearance of $\hat{k}$ into the ciphertext is mediated by the relaxed commitment scheme. In particular, since com is included both in the clear and inside each ciphertext component (which are individually AIBE-IND-CCA-secure as part of c^*), and since the decryption algorithm checks that they be consistent, the adversary is forced to keep in the outer layer of her query ciphertext c the same value of com that was in the challenge c^*, or decryption would fail. Now for that value of com, by the relaxed binding property, the only valid PRF key that can be decommitted is $\hat{k}$. At this point the argument would seem to get stuck again, as it is not apparent how to guarantee that the adversary does not learn enough about $\hat{k}$ from the several ciphertext components in c^* so as to be able to compute F-values under that key. As it turns out, this point can also be tamed through a separate sequence-of-games analysis [20]. It then follows that the adversary will not be able to compute the proper σ for the ciphertext she was trying to craft, which finally fully justifies the $\perp$ reply by the simulator.

Guess: $\mathcal{A}$ outputs a guess b and $\mathcal{B}$ passes this bit as his guess for b' to $\mathcal{C}'$.

Observe that, by construction, it holds that if $\mathcal{C}'$ chooses $b' = 0$, then $\mathcal{B}$ is playing Game_{h-1}, whereas if $b' = 1$, then $\mathcal{B}$ is playing $\overline{\text{Game}_h}$. Therefore, the PRF and the AIBE\$-IND-CCA advantage of B is essentially $\mathcal{A}$'s advantage in distinguishing

Game$_{h-1}$ from $\overline{\text{Game}}_h$, *i.e.*,

$$\left|\mathsf{Adv}^{h-1}_{\mathcal{A},\Pi} - \overline{\mathsf{Adv}}^{h}_{\mathcal{A},\Pi}\right| \leq \epsilon_1 + \epsilon_2. \qquad \blacksquare$$

Lemma 4.4.8: *For $1 \leq h \leq l$, if $\mathcal{H}_{es}$ in an (t_3, ϵ_3)-entropy-smoothing hash function family and the decisional Diffie-Hellman problem is (t_4, ϵ_4)-hard in $\mathbb{G}$, then $\mathcal{A}$'s advantage of distinguishing $\overline{\mathit{Game}}_h$ from Game$_h$ is at most $\epsilon_3 + \left(\epsilon_4 + \frac{Q_D}{q}\right)$, i.e.,*

$$\left|\overline{\mathsf{Adv}}^{h}_{\mathcal{A},\Pi} - \mathsf{Adv}^{h}_{\mathcal{A},\Pi}\right| \leq \epsilon_3 + \left(\epsilon_4 + \frac{Q_D}{q}\right). \qquad \square$$

Proof. The proof of this lemma follow with the help of two intermediate games $\widetilde{\text{Game}}_{1,h}$ and $\widetilde{\text{Game}}_{2,h}$. During the transition from $\overline{\text{Game}}_h$ to $\widetilde{\text{Game}}_{1,h}$, we replace $(B^{\mathsf{com}}_{1,\mathsf{HID}_{l-h+1}} B_{2,\mathsf{HID}_{l-h+1}})^s$ with a random group element $r_2 \in \mathbb{G}$. Next, during the transition from $\widetilde{\text{Game}}_{1,h}$ to $\widetilde{\text{Game}}_{2,h}$, we replace $(A^{\mathsf{com}}_{1,\mathsf{HID}_{l-h+1}} A_{2,\mathsf{HID}_{l-h+1}})^s$ with another random group element $r_1 \in \mathbb{G}$. Finally, during the transition from $\widetilde{\text{Game}}_{2,h}$ to Game$_h$, we replace $H(r_1, r_2)$ with a truly random bit-string of length λ.

The indistinguishability of the first two transitions follows via a reduction argument from the DDH problem and a PPT adversary $\mathcal{B}$ that internally executes the oABE\$-IND-CCA game with the adversary $\mathcal{A}$ in order to gain advantage in breaking the DDH assumption. This reduction argument proceeds along the same lines as Lemma 1 of [83]. The indistinguishability of the second transition follows from the fact that $\mathcal{H}_{es}$ is an entropy-smoothing hash function. Therefore, we have

$$\left|\overline{\mathsf{Adv}}^{h}_{\mathcal{A},\Pi} - \mathsf{Adv}^{h}_{\mathcal{A},\Pi}\right| \leq \epsilon_3 + \left(\epsilon_4 + \frac{Q_D}{q}\right). \qquad \blacksquare$$

4.5 Constructions

We now present three constructions of broadcast steganography: one for each model of security defined in Section 4.3.2. Our constructions employ the encrypt-then-embed paradigm depicted in Figure 4.1, using outsider-anonymous broadcast encryption with pseudorandom ciphertexts (Section 4.4) for encryption and rejection-sampling [7,67,110] for embedding. In what follows, s_i^σ denotes the i^{th} leftmost non-overlapping substring with length σ of a given bit-string s.

4.5.1 A BS-IND-CHA-Secure Construction

The rejection-sampler function used in our first construction is given in Figure 4.10. Sample takes as input a security parameter λ, a channel history $h \in \Sigma^*$, a function $H : \Sigma \to \{0,1\}$, and a bit-string $c \in \{0,1\}^*$, and outputs a covertext $s \in \Sigma^*$. Internally, for every bit c_i, Sample attempts to find a covertext $s_i^\sigma \in \Sigma$ such that $H(s_i^\sigma) = c_i$ by repeatedly querying the channel oracle up to λ number of times. Sample may fail to find a valid s_i during the λ iterations, but only with negligible probability in the parameter λ. This mechanism allows a simple method to extract c from s: compute $c = H(s_1^\sigma)\| \ldots \|H(s_l^\sigma)$ where $l = |s|/\sigma$. As shown in [9,110], if the channel is always informative, H is a strong 2-universal hash function, and c is uniformly random, then the maximum statistical distance between $s_1 \leftarrow \mathsf{Sample}(\lambda, h, H, c)$ and $s_2 \leftarrow \mathfrak{C}_h^{|c|}$ for any valid $h \in \Sigma^*$ is negligible in the security parameter λ. For simplicity, we denote this statistical distance when $|c| = 1$ by ϵ_1 in the reminder of this chapter.

We obtain our BS-IND-CHA-secure broadcast steganography scheme by combining the rejection-sampler function from Figure 4.10 with our outsider-anonymous broadcast encryption scheme with pseudorandom ciphertexts given in Figures 4.6 to 4.9. Formally, given a strong 2-universal hash function family $\mathcal{H}_{s2u} = \{H : \Sigma \to \{0,1\}\}$ and an oABE\$-IND-CPA-secure outsider-anonymous broadcast encryption with pseudo-

```
Function: Sample($\lambda, h, H, c$)
Input: parameter $\lambda$, history $h$, function $H$, bit-string $c$
Output: stegotext $s$
 1 $l \coloneqq |c|$
 2 for $i \coloneqq 1$ to $l$ do
 3     $j \coloneqq 0$
 4     repeat
 5         $j \coloneqq j + 1,\ s_i \leftarrow \mathfrak{C}_h$
 6     until $H(s_i) = c_i \vee j = \lambda$
 7     $h \coloneqq h \| s_i$
 8 end
 9 $s \coloneqq s_1 \| \dots \| s_l$
10 return $s$
```

Figure 4.10: The regular rejection-sampler function.

```
Function: DSample($\lambda, H, c, r$)
Input: parameter $\lambda$, function $H$, bit-string $c$, randomness $r$
Output: stegotext $s$
 1 $l \coloneqq |c|$
 2 for $i \coloneqq 1$ to $l$ do
 3     $j \coloneqq 0$
 4     repeat
 5         $j \coloneqq j + 1,\ s_i \coloneqq \mathsf{Channel}(r^{\lambda}_{\lambda(i-1)+j})$
 6     until $H(s_i) = c_i \vee j = \lambda$
 7 end
 8 $s \coloneqq s_1 \| \dots \| s_l$
 9 return $s$
```

Figure 4.11: The deterministic rejection-sampler function.

```
Algorithm: Setup$(1^\lambda, N)$
1 $(\mathsf{MPK}', \mathsf{MSK}') \leftarrow \mathsf{Setup}'(1^\lambda, N)$
2 $H \leftarrow\!\$\ \mathcal{H}_{s2u}$
3 $\mathsf{MPK} \coloneqq (\mathsf{MPK}', H)$
4 $\mathsf{MSK} \coloneqq \mathsf{MSK}'$
5 return $(\mathsf{MPK}, \mathsf{MSK})$
```

Figure 4.12: Setup algorithm of our BS-IND-CHA-secure construction.

```
Algorithm: KeyGen$(\mathsf{MPK}, \mathsf{MSK}, i)$
1 $sk_i \leftarrow \mathsf{KeyGen}'(\mathsf{MPK}', \mathsf{MSK}', i)$
2 return $sk_i$
```

Figure 4.13: KeyGen algorithm of our BS-IND-CHA-secure construction.

```
Algorithm: Encode$(\mathsf{MPK}, S, h, m)$
1 $c \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', S, m)$
2 $s \leftarrow \mathsf{Sample}(\lambda, h, H, c)$
3 return $s$
```

Figure 4.14: Encode algorithm of our BS-IND-CHA-secure construction.

```
Algorithm: Decode$(\mathsf{MPK}, sk_i, s)$
1 $l \coloneqq |s|/\sigma$
2 for $j \coloneqq 1$ to $l$ do
3     $c_j \coloneqq H(s_j^\sigma)$
4 end
5 $c \coloneqq c_1 \| \ldots \| c_l$
6 $m \coloneqq \mathsf{Decrypt}'(\mathsf{MPK}', sk_i, c)$
7 return $m$
```

Figure 4.15: Decode algorithm of our BS-IND-CHA-secure construction.

random ciphertexts scheme $\Pi' = (\mathsf{Setup}', \mathsf{KeyGen}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ with expansion ℓ (*i.e.*, $|\mathsf{Encrypt}'(\mathsf{MPK}', S, m)| = \ell(|m|)$), we build a BS-IND-CHA-secure BS scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encode}, \mathsf{Decode})$ as given in Figures 4.12 to 4.15.

Remark 4.5.1. If the outsider-anonymous broadcast encryption scheme with pseudorandom ciphertexts employed in our BS-IND-CHA-secure broadcast steganography construction is oABE\$-IND-PDR-CCA-secure, then the resulting broadcast steganography scheme is BS-IND-PDR-CCA-secure.

Theorem 4.5.2: *If the channel is always informative, $\mathcal{H}_{s2u}$ is a strong 2-universal hash function family, and the oABE\$ scheme Π' is (t_2, Q_U, ϵ_2)-oABE\$-IND-CPA-secure, then the construction in Figures 4.12 to 4.15 is $(t_2, Q_U, \mu\epsilon_1 + \epsilon_2)$-BS-IND-CHA-secure, where μ is the polynomial bound on the total message length.* □

Proof. We organize the proof as a sequence of games (Game$_0$, Game$_1$, Game$_2$) between a BS-IND-CHA adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. During the *Challenge* step in Game$_0$, $\mathcal{A}$ is given a stegotext for m^* under S^*, whereas in Game$_2$, $\mathcal{A}$ is given a covertext consisting of some samples from the channel oracle.

Game$_0$: This game is the actual BS-IND-CHA game when the challenge bit b^* is fixed to 0. The interaction between $\mathcal{A}$ and $\mathcal{C}$ during *Setup*, *Phase 1*, *Phase 2*, and *Guess* steps follows as specified in Definition 4.3.7. During the *Challenge* step, $\mathcal{A}$ sends $\mathcal{C}$ a message $m^* \in \mathcal{MSP}$, a legal history $h \in \Sigma^*$, and a set of user identities $S^* \subseteq U$ with the restriction that $S^* \cap R_U = \emptyset$. Next, $\mathcal{C}$ generates the challenge stegotext s^*, which is later sent to $\mathcal{A}$, as follows.

```
1 $c \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', S^*, m^*)$
2 $s^* \leftarrow \mathsf{Sample}(\lambda, h, H, c)$
```

Game$_1$: This game is similar to Game$_0$, but $\mathcal{C}$ computes the challenge s^* as follows.

```
1 $c \leftarrow_\$ \{0,1\}^{\ell(|m^*|)}$
```

```
2 $s^* \leftarrow \mathsf{Sample}(\lambda, h, H, c)$
```

Game$_2$: This game is similar to Game$_1$, but $\mathcal{C}$ now computes the challenge s^* as a covertext consisting of samples from the channel oracle.

```
1 $s^* \leftarrow \mathfrak{C}_h^{\ell(|m^*|)}$
```

For $0 \leq i \leq 2$, let $\mathsf{Adv}^i_{\mathcal{A},\Pi}$ denote $\mathcal{A}$'s advantage of winning Game$_i$. Since Π' is $(t_2, Q_{sk}, \epsilon_2)$-oABE\$-IND-CPA-secure, it follows from a straightforward reduction argument that $\mathcal{A}$'s advantage in distinguishing Game$_0$ from Game$_1$ is at most ϵ_2 (*i.e.*, $\left|\mathsf{Adv}^0_{\mathcal{A},\Pi} - \mathsf{Adv}^1_{\mathcal{A},\Pi}\right| \leq \epsilon_2$). Once we bound the total message length by the polynomial μ, it follows from another simple reduction argument that $\mathcal{A}$'s advantage in distinguishing Game$_1$ from Game$_2$ is at most $\mu\epsilon_1$ (*i.e.*, $\left|\mathsf{Adv}^1_{\mathcal{A},\Pi} - \mathsf{Adv}^2_{\mathcal{A},\Pi}\right| \leq \mu\epsilon_1$). Therefore, we have

$$\left|\mathsf{Adv}^0_{\mathcal{A},\Pi} - \mathsf{Adv}^2_{\mathcal{A},\Pi}\right| \leq \mu\epsilon_1 + \epsilon_2.$$

The theorem then follows from the observation that Game$_2$ amounts to the actual BS-IND-CHA game when the challenge bit b^* is fixed to 1. ■

4.5.2 A BS-IND-CCA-Secure Construction

Unfortunately, our first construction fails to provide a BS-IND-CCA-secure broadcast steganography scheme even if the internally used outsider-anonymous broadcast encryption scheme with pseudorandom ciphertexts provides oABE\$-IND-CCA security. The problem is that the rejection-sampler function from Figure 4.10 allows multiple covertexts corresponding to a given bit-string. However, this limitation can be overcome in the case of channels that are efficiently computable and whose samples are independently distributed. In fact, for channels of this type, Hopper [65] devised a *deterministic* rejection-sampler function DSample that maps a given bit-string to exactly one covertext.

```
Algorithm: $\mathsf{Setup}(1^\lambda, N)$
$(\mathsf{MPK}', \mathsf{MSK}') \leftarrow \mathsf{Setup}'(1^\lambda, N)$
$H \leftarrow\$ \mathcal{H}_{s2u}$
$\mathsf{MPK} \coloneqq (\mathsf{MPK}', H, G)$
$\mathsf{MSK} \coloneqq \mathsf{MSK}'$
return $(\mathsf{MPK}, \mathsf{MSK})$
```

Figure 4.16: Setup algorithm of our BS-IND-CCA-secure construction.

```
Algorithm: $\mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)$
$sk_i \leftarrow \mathsf{KeyGen}'(\mathsf{MPK}', \mathsf{MSK}', i)$
return $sk_i$
```

Figure 4.17: KeyGen algorithm of our BS-IND-CCA-secure construction.

```
Algorithm: $\mathsf{Encode}(\mathsf{MPK}, S, m)$
$\hat{r} \leftarrow\$ \{0,1\}^\lambda$
$c \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', S, \hat{r} \| m)$
$r \coloneqq G(\hat{r}, |c| \cdot \lambda^2)$
$s \coloneqq \mathsf{DSample}(\lambda, H, c, r)$
return $s$
```

Figure 4.18: Encode algorithm of our BS-IND-CCA-secure construction.

```
Algorithm: $\mathsf{Decode}(\mathsf{MPK}, sk_i, s)$
$l \coloneqq |s|/\sigma$
for $j \coloneqq 1$ to $l$ do
    $c_j \coloneqq H(s_j^\sigma)$
end
$c \coloneqq c_1 \| \dots \| c_l$
$m' \coloneqq \mathsf{Decrypt}'(\mathsf{MPK}', sk_i, c)$
if $m' \neq \perp$ then
    parse $m'$ as $\hat{r} \| m$
    $r \coloneqq G(\hat{r}, l \cdot \lambda^2)$
    if $\mathsf{DSample}(\lambda, H, c, r) = s$ then
        return $m$
    end
end
return $\perp$
```

Figure 4.19: Decode algorithm of our BS-IND-CCA-secure construction.

As shown in Figure 4.11, $\mathsf{DSample}$ takes in input a security parameter λ, a predicate $H : \Sigma \to \{0,1\}$ along with a bit-string $c \in \{0,1\}^*$ to embed, and a random bit-string $r \in \{0,1\}^{|c|\cdot\lambda^2}$ that controls the embedding. To sample $s \in \Sigma^*$, for every bit c_i of c, $\mathsf{DSample}$ seeks $s_i^\sigma \in \Sigma$ such that $H(s_i^\sigma) = c_i$, by repeatedly drawing from the channel according to the random chunks specified in r. This approach requires that the channel be efficiently computable by a function $\mathsf{Channel}$ whose samples are independent of the history (hence we drop h from its input), but guarantees that an adversary who intercepts a stegotext is unable to tweak it meaningfully. Furthermore, as shown in [9, 66, 110], if H is a strong 2-universal hash function, and c and r are uniformly random, then the statistical distance between stegotexts produced by $\mathsf{DSample}$ and innocent covertexts sampled from $\mathsf{Channel}$ is a negligible function ϵ_1 of λ. The definition of a strong 2-universal hash function is given in Section 2.3.2.

Figures 4.16 to 4.19 reports the details of our BS-IND-CCA-secure broadcast steganography scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encode}, \mathsf{Decode})$, which is based on a strong 2-universal hash function family $\mathcal{H}_{s2u}$, a variable-length pseudorandom generator (vPRG) $G : \{0,1\}^\lambda \times \mathbb{Z} \to \{0,1\}^*$ (whose second input sets the output length), and an oABE\$-IND-CCA-secure outsider-anonymous broadcast encryption scheme with pseudorandom ciphertexts $\Pi' = (\mathsf{Setup}', \mathsf{KeyGen}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ with expansion ℓ.

Theorem 4.5.3: *If the channel is always informative, $\mathcal{H}_{s2u}$ is a strong 2-universal hash function family, G is a (t_2, ϵ_2)-hard vPRG, and the oABE\$ scheme Π' is $(t_3, Q_U, Q_D, \epsilon_3)$-oABE\$-IND-CCA-secure, then the construction presented in Figures 4.16 to 4.19 is $(t_2 + t_3, Q_U, Q_D, \mu\epsilon_1 + \epsilon_2 + \epsilon_3)$-BS-IND-CCA-secure, where μ is the polynomial bound on the total message length.* □

Proof. We organize this proof as a sequence of games (Game$_0$, Game$_1$, Game$_2$, Game$_3$) between a BS-IND-CCA adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. During the *Challenge* step of Game$_0$, $\mathcal{A}$ is given a stegotext for m^* under S^*. The stegotext given to $\mathcal{A}$ during the *Challenge* step of Game$_3$, on the other hand, consists just of documents sampled from the channel function under uniform randomness.

Game$_\mathbf{0}$: This game is the actual BS-IND-CCA game when the challenge bit b^* is fixed to 0. The interaction between $\mathcal{A}$ and $\mathcal{C}$ during *Setup*, *Phase 1*, *Phase 2*, and *Guess* steps follows as specified in Definition 4.3.5. After $\mathcal{A}$ submitted a message $m^* \in \mathcal{MSP}$ and a set of user identities $S^* \subseteq U$ (with the restriction that $S^* \cap R_U = \emptyset$) during the *Challenge* step, $\mathcal{C}$ generates the challenge stegotext s^*, which is later given to $\mathcal{A}$, as follows.

```
1 $\hat{r} \leftarrow_\$ \{0,1\}^\lambda$
2 $c \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', S^*, \hat{r} \| m^*)$
3 $r := G(\hat{r}, |c| \cdot \lambda^2)$
4 $s^* := \mathsf{DSample}(\lambda, H, c, r)$
```

Game$_\mathbf{1}$: This game is similar to Game$_0$, but $\mathcal{C}$ now computes the challenge stegotext s^* as given below.

```
1 $\hat{r} \leftarrow_\$ \{0,1\}^\lambda$
2 $c \leftarrow_\$ \{0,1\}^{\ell(\lambda+|m^*|)}$
3 $r := G(\hat{r}, |c| \cdot \lambda^2)$
4 $s^* := \mathsf{DSample}(\lambda, H, c, r)$
```

Game$_\mathbf{2}$: This game is similar to Game$_1$, but $\mathcal{C}$ now computes the challenge stegotext s^* as shown below.

```
1 $c \leftarrow_\$ \{0,1\}^{\ell(\lambda+|m^*|)}$
2 $r \leftarrow_\$ \{0,1\}^{|c| \cdot \lambda^2}$
3 $s^* := \mathsf{DSample}(\lambda, H, c, r)$
```

Game$_\mathbf{3}$: This game is similar to Game$_2$, but $\mathcal{C}$ generates the challenge stegotext s^* as follows.

```
1 $l := \ell(\lambda + |m^*|)$
2 for $j := 1$ to $l$ do
3     $r \leftarrow_\$ \{0,1\}^\lambda$
4     $s_j^* := \mathsf{Channel}(r)$
5 $s^* := s_1^* \| \ldots \| s_l^*$
```

For $0 \leq i \leq 3$, let $\mathsf{Adv}^i_{\mathcal{A},\Pi}$ denote $\mathcal{A}$'s advantage of winning Game$_i$. Because Π' is $(t_3, Q_{sk}, Q_d, \epsilon_3)$-oABE\$-IND-CCA-secure, it follows from a simple reduction argument that $\mathcal{A}$'s advantage in distinguishing Game$_0$ from Game$_1$ is at most ϵ_3 (*i.e.*, $\left|\mathsf{Adv}^0_{\mathcal{A},\Pi} - \mathsf{Adv}^1_{\mathcal{A},\Pi}\right| \leq \epsilon_3$). Since G is (t_2, ϵ_2)-hard, it follows from another straightforward reduction argument that A's advantage in distinguishing Game$_1$ from Game$_2$ is at most ϵ_2 (*i.e.*, $\left|\mathsf{Adv}^1_{\mathcal{A},\Pi} - \mathsf{Adv}^2_{\mathcal{A},\Pi}\right| \leq \epsilon_2$). Once we bound the total message length by the polynomial μ, it follows from yet another simple reduction argument that $\mathcal{A}$'s advantage in distinguishing Game$_2$ from Game$_3$ is at most $\mu\epsilon_1$ (*i.e.*, $\left|\mathsf{Adv}^2_{\mathcal{A},\Pi} - \mathsf{Adv}^3_{\mathcal{A},\Pi}\right| \leq \mu\epsilon_1$). Thus,

$$\left|\mathsf{Adv}^0_{\mathcal{A},\Pi} - \mathsf{Adv}^3_{\mathcal{A},\Pi}\right| \leq \mu\epsilon_1 + \epsilon_2 + \epsilon_3.$$

The theorem then follows from the observation that Game$_3$ amounts to the actual BS-IND-CCA game when the challenge bit b^* is fixed to 1. ∎

Chapter 5

Oblivious Group Storage

5.1 Introduction

Recent developments in cloud computing has given rise to the convenience of remote storage services, *a.k.a.* cloud storage. Taking advantage of the economies of scale, third party companies are now able to provide cloud storage with high availability and low latency for pay-per-use cost structures that amount to a couple of pennies per gigabyte stored [6, 62, 88]. Instead of enduring the cost and the hassle of managing in-house storage infrastructure, more and more individuals and companies alike are relying on cloud storage to manage large amounts of data they generate.

One of the biggest concerns with outsourcing data storage to a cloud is the confidentiality of the outsourced data. Most security-conscious clients encrypt their data before exporting to a cloud. However, while encryption preserves the secrecy of the data, it does not hide the access patterns between the clients and the storage providers. It has been shown that in certain situations, these access patterns can reveal a considerable amount of sensitive information regarding the outsourced data [32,70,92]. For example, as Islam *et al.* [70] demonstrated, statistical attacks on access patterns between the clients and an encrypted email repository can be leveraged to infer about 80% of the search queries. It is evident that a secure cloud storage system should preserve the confidentiality of the outsourced data and obfuscate the access patterns between the clients and their cloud storage providers. These requirements have been formalized in the cryptographic literature under the term *oblivious storage* (OS) [5, 21, 25, 35, 36, 58–61, 78, 84, 92, 102, 105–108, 115–117, 119].

A related notion to oblivious storage is *oblivious random access machine* (ORAM). It is a mechanism proposed by Goldreich and Ostrovsky [55, 56, 90] for protecting computer software from memory access pattern-based reverse engineering attacks. In this setting, the authors represented a computer software as a *random access machine* (RAM) program executed within a CPU that accessed external memory; and showed that an adversary who had control over only the memory had the potential to reverse engineer the RAM program executed inside the CPU even when the CPU was fully trusted. They proved that this attack could be prevented if one converted the RAM program into an ORAM program.

With the insight that the CPU and the external memory in the setting of software protection conveniently correlate to the client and the remote storage in the cloud storage setting, Williams and Sion [114] proposed that ORAM protocols can be used to construct oblivious storage protocols. Since then, there have been several subsequent results [5, 21, 25, 35, 36, 58–61, 78, 84, 92, 102, 105–108, 115–117, 119] utilizing ORAM protocols in the construction of oblivious storage protocols. While most of these results have focused on optimizing the parameters of the original model of ORAM [21, 58–61, 78, 92, 102, 106–108, 115, 116, 119], others have proposed constructions that outsourced storage to multiple non-colluding data centers [84, 105], that could support parallel executions from mutually trusting clients [25, 117], that were information theoretically secure [36], statistically secure [35], and even secure without the need for any cryptographic assumptions [5].

An attractive benefit of having the data outsourced to a cloud is the ability to easily share that data with other clients. In fact, most of the popular cloud storage providers nowadays provide some mechanism, such as temporary links [45], for their clients to share their data. However, giving group access to outsourced data while also preserving the oblivious storage guarantees is still a problem not well understood, let alone solved. There have been several proposals, however, to provide oblivious group data access to cloud storage [25, 61, 72, 117]. All these proposals only provide an all-or-nothing access control policy to the outsourced content since they all require (either explicitly [25, 61, 117] or implicitly [72]) the authorized clients to share the secret keys to the cloud storage. As real-world clients prefer to impose fine-grained access control policies over their shared data, an all-or-nothing level of access is simply out of the question. Part of the reason behind this disconnect with the real-world requirements is in the original definition of OS that is based on the assumption that only a single client accesses the outsourced content.

An important way of categorizing the OS protocols from the perspective of oblivious shared storage is *stateful vs. stateless* solutions. In a stateful OS protocol, the client is required to maintain some state in between data access requests [5,21,35,36,60,105–108, 116,119], whereas in a stateless OS protocol [25,55,56,58,59,61,78,84,90,92,102,115,117], the client carries no such state. This difference renders stateless OS protocols an ideal starting point for oblivious shared storage protocols since sharing some state among clients in a shared data access setting prohibitively increases the communication complexity. In fact, there have been a couple of results [25, 61, 72, 117] that attempted to construct oblivious shared storage protocols using stateless ORAM protocols, albeit with an all-or-nothing access policy on the shared data. Still, constructing oblivious shared storage protocols that match the read-world data-secrecy and access-pattern-obliviousness requirements is a non-trivial task.

Jinsheng *et al.* [72] proposed *multi-user oblivious RAM*, a promising extension of ORAM to the oblivious shared storage setting. The novel idea of M-ORAM is the inclusion of a sequence of proxies, called the anonymizers, in between the clients and the cloud storage provider. Any message sent between the clients and the storage provider would pass through the anonymizers. With this arrangement, Jinsheng *et al.* [72] were able to prove that as long as at least one anonymizer remains honest, the access pattern between any honest user and the storage provider remains hidden from any other user, the anonymizers, and the storage provider. However, all the clients in a M-ORAM protocol, including those who are compromised, have access to all the data in the cloud storage; thus, even with the help of anonymizers, M-ORAM can only provide all-or-nothing level of access control on the outsourced data. A review of the setting of M-ORAM is given in Section 2.4.6.

5.2 Contributions

To illustrate the need for fine-grained access control for shared data, consider a hospital that wishes to outsource the storage of medical records to a third-party cloud storage provider. Since encryption alone is not enough to protect the medical records from the prying eyes of the outsiders, including the third-party cloud storage provider [32,70,92], the hospital may decide to communicate with the cloud storage using an OS protocol. However, all the existing OS protocols allow the entire hospital staff to access the medical records of any patient of their choosing. This is a serious violation of patient privacy since the medical records of any individual patient should only be accessible to medical staff directly related to that patient's medical history [30]. A more alarming concern with this approach is that the compromise of the access credentials of a single member of the hospital staff has the potential to compromise the medical records of all the patients of the hospital. As one can see, what the hospital really needs is an OS-like system that also allows the medical staff to control access to the outsourced medical data at the record level.

Table 5.1: A comparison of different kinds of oblivious cloud storage protocols. The parameters are chosen to indicate to which extent these protocols allow multiple clients to share a cloud storage.

Protocol	Stateful ORAM	Stateless ORAM	M-ORAM	OGS
Allows multiple clients to share a cloud storage.	✗	✓	✓	✓
Allows clients to hide their data access patterns with the cloud storage from other clients.	✗	✗	✓	✓
Allows revoked clients to collude with the cloud storage provider.	✗	✗	✓	✓
Allows clients to hide their data stored at a cloud storage from revoked clients.	✗	✗	✗	✓

Oblivious Group Storage. In this chapter, we propose *oblivious group storage* (OGS), an extension of the OS model to the setting of shared data access. An OGS protocol allows one to share the data stored in a cloud with fine-grained access control policies while also enjoying the data-secrecy and access-pattern obliviousness guarantees provided by OS protocols. Table 5.1 shows how OGS compares to other types of oblivious cloud storage protocols. As one can see, only OGS allows clients to impose fine-grained access control policies over their data. We believe that this is the first proposal to provide such capabilities in the setting of oblivious cloud storage.

Organization. Our presentation is twofold. First, in Section 5.3, we formally define the setting and the security model of OGS so that one can argue about oblivious shared data access without ambiguity. A crucial component of the setting of OGS is the inclusion of a sequence of proxies, an idea originally proposed by Jinsheng *et al.* [72], that facilitate the communication between the clients and the cloud storage provider. Second, we provide a proof-of-concept construction in Section 5.4 along with a formal analysis of security with respect to our security model in order to show the feasibility of OGS. This construction can be seen as a framework that produces an OGS protocol by combining existing cryptographic primitives.

5.3 Formal Model

Oblivious group storage is an extension of oblivious storage. An OGS protocol allows a group of capacity-constrained clients to obliviously share the storage at a remote untrusted storage provider $\mathcal{S}$, where the amount of storage is abundant and the cost of storage is cheap. The clients also have some storage capacity, although it is much smaller than at the server $\mathcal{S}$.

An OGS protocol provides several security guarantees. Similar to a stateless oblivious storage protocol deployed in a multi-user setting [25, 61, 117], an OGS protocol allows the clients to hide the contents of the outsourced data items from $\mathcal{S}$ and prevents $\mathcal{S}$ from extracting any significant information about the outsourced data items based on the pattern in accessing the data between the clients and $\mathcal{S}$. Also, similar to a multi-user oblivious RAM protocol [72], an OGS protocol allows any individual client to prevent other clients from extracting any non-trivial information about the outsourced data by observing his access patterns with $\mathcal{S}$. In addition, an OGS protocol also allows the clients to share the data items stored at $\mathcal{S}$ among subgroups of clients so that the shared data items are hidden from the unauthorized clients even if they *collude* with $\mathcal{S}$. In order to provide security against collusion, an OGS protocol makes use of a sequence of proxies, *a.k.a.* anonymizers, between the clients and $\mathcal{S}$.

5.3.1 Setting of OGS

An OGS protocol consists of four types of parties: the system initialization authority, the clients, the anonymizers, and the storage provider $\mathcal{S}$; and four algorithms: Setup, KeyGen, Write, and Read. The two duties of the SIA are,

1. To initialize the OGS protocol by generating the system-wide secret keys, the anonymizer secret keys, and setting up the initial server storage at $\mathcal{S}$ via the algorithm Setup.

2. To enroll new clients to an OGS instance by generating client-specific secret keys via the algorithm KeyGen.

Once the OGS protocol is initialized and the clients are enrolled into the system, the clients interact with $\mathcal{S}$ through the anonymizers to store new data items in $\mathcal{S}$'s storage by executing the algorithm Write and to retrieve existing data items from $\mathcal{S}$'s storage by running the algorithm Read. The clients can also share the storage at $\mathcal{S}$ by authorizing other clients to execute Read and Write algorithms on the data items that they have stored at $\mathcal{S}$. The formal definition of an OGS protocol is given in Definition 5.3.1 below.

Following the standards of existing OS protocols, we model the storage in OGS protocols as a key-value interface. We refer to the key-value pairs supplied by the clients for storage as *position-message* pairs. Depending on the execution path of the OGS algorithm, these position-message pairs translate into one or more key-value pairs in the storage provided by $\mathcal{S}$. We refer to those key-value pairs that get stored at $\mathcal{S}$ as *address-block* pairs.

As we described in Definition 5.3.1, the algorithms Read and Write executed between the clients, the anonymizers, and $\mathcal{S}$ are interactive. We call the complete execution of a single interactive algorithm between these parties an *episode*. For ease of reference, we refer to the entire storage at $\mathcal{S}$ at the completion of an episode a server state st. It should be noted that a given server state includes all the address-block pairs stored at $\mathcal{S}$ as well as any other outsourced data (such as encrypted secret keys, encrypted hash function seeds, *etc.*) related to a given OGS instance.

Definition 5.3.1 (OGS Setting): An oblivious group storage scheme, associated with an SIA, a set of clients $C = [1, N]$, a set of anonymizers $A = [1, M]$, a server $\mathcal{S}$, a position space $\mathcal{PSP}$, and a message space $\mathcal{MSP}$, is a tuple of algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Write}, \mathsf{Read})$ defined as follows.

$\boldsymbol{(\mathsf{MPK}, \mathsf{MSK}, ak_1, \ldots, ak_M, st_0) \leftarrow \mathsf{Setup}(1^\lambda, M, N)}$**:** Setup is a non-interactive algorithm executed by the SIA. This algorithm takes the security parameter 1^λ and the number of anonymizers and clients in the system M, N as inputs and outputs the master public key MPK, the master secret key MSK, the anonymizers' secret keys $ak_1, \ldots, ak_M$, and the initial state of the server storage st_0. At the end of this algorithm, the SIA places MPK in a publicly accessible location, keeps MSK to herself, provides the anonymizers' secret keys to the corresponding anonymizers, and places st_0 in the storage provided by $\mathcal{S}$.

$\boldsymbol{ck_i} \leftarrow \textsf{\textbf{KeyGen}}(\textsf{\textbf{MPK}}, \textsf{\textbf{MSK}}, \boldsymbol{i})$**:** $\textsf{KeyGen}$ is also a non-interactive algorithm executed by the SIA. This algorithm takes the master public key $\textsf{MPK}$, the master secret key $\textsf{MSK}$, and a client $i \in C$ as inputs and outputs a secret key ck_i for client i. At the end of this algorithm, the SIA gives ck_i to client i.

$(\boldsymbol{ak'_1, \ldots, ak'_M, st_{t+1}}) \leftarrow \textsf{\textbf{Write}}(\textsf{\textbf{MPK}}, \boldsymbol{ak_1, \ldots, ak_M, ck_i, S, p, m, st_t})$**:** $\textsf{Write}$ is an interactive algorithm executed between a client $i \in C$, the M anonymizers, and $\mathcal{S}$. The anonymizers supply their secret keys $ak_1, \ldots, ak_M$. The client i supplies his secret key ck_i, a set of receivers $S \subseteq C$, a position $p \in \mathcal{PSP}$, and a message $m \in \mathcal{MSP}$. $\mathcal{S}$ provides the current server state st_t. At the end of this algorithm, the server state transforms from st_t to st_{t+1} and the old anonymizers' secret keys get replaced by the new secret keys $ak'_1, \ldots, ak'_M$.

$(\boldsymbol{m/\perp, ak'_1, \ldots, ak'_M, st_{t+1}}) \leftarrow \textsf{\textbf{Read}}(\textsf{\textbf{MPK}}, \boldsymbol{ak_1, \ldots, ak_M, ck_i, p, st_t})$**:** $\textsf{Read}$ is also an interactive algorithm executed between a client $i \in C$, the M anonymizers, and $\mathcal{S}$. The anonymizers supply their secret keys $ak_1, \ldots, ak_M$. The client i provides the secret key ck_i and a position p. $\mathcal{S}$ provides the current server state st_t. At the end of this algorithm, the client i learns a message m or the failure symbol $\perp$, the server state transforms from st_t to st_{t+1}, and the old anonymizers' secret keys get replaced by the new secret keys $ak'_1, \ldots, ak'_M$.

Correctness. Let $\textsf{MPK}$, $ak_1, \ldots, ak_M$, and st_t be a valid master public key, a sequence of M anonymizers' valid secret keys, and a valid server state, respectively. For every $h \in C$, $S \subseteq C$, $i \in S$, $p \in \mathcal{PSP}$, and $m \in \mathcal{MSP}$, if ck_h, ck_i are the secret keys of the clients h, i, respectively, $\textsf{Write}(\textsf{MPK}, ak_1, \ldots, ak_M, ck_h, S, p, m, st_t) = (ak'_1, \ldots, ak'_M, st_{t+1})$, and $ak'_1, \ldots, ak'_M$ and st_{t+1} transform into $\overline{ak}'_1, \ldots, \overline{ak}'_M$ and $\overline{st}_{t+1}$ after zero or more executions of $\textsf{Read}$ and $\textsf{Write}$ algorithms that do not modify the message at position p, then $\textsf{Read}(\overline{ak}'_1, \ldots, \overline{ak}'_M, ck_i, p, \overline{st}_{t+1})$ yields m, except with negligible probability in the security parameter λ. $\Diamond$

5.3.2 Security of OGS

The basic security requirement of a secure storage protocol is to prevent unauthorized parties from learning any non-trivial information about the data items. Usually, this requirement is satisfied by encrypting the data items using a semantically secure cryptosystem before exporting to the storage provider. As mentioned in Section 5.1, the pattern in accessing data items between a storage provider and its clients can reveal non-trivial information regarding the data items. The goal of an oblivious storage protocol is to also prevent that information leakage.

Informally, the security model of an OGS protocol guarantees four aspects of security for honest clients against unauthorized clients and an honest-but-curious server, who are also allowed to collude. These aspects of security are, namely, server-side/client-side access pattern obliviousness and server-side/client-side data secrecy. The two aspects of access pattern obliviousness require that any honest client's pattern of accessing data stored at $\mathcal{S}$ leaks no non-trivial information regarding the accessed data items to either the unauthorized clients or $\mathcal{S}$. Server-side data secrecy aspect assures that any honest client's exported data items are hidden from $\mathcal{S}$ even though the server states are maintained by $\mathcal{S}$ during the execution of the OGS protocol. As mentioned in Section 5.2, one attractive use case of an OGS protocol is that it allows the clients to share the storage among groups of clients. The client-side data secrecy property guarantees that the data items belonging to any group of such clients are not accessible to any unauthorized client event if that client is a legitimate participant of the OGS protocol.

Remark 5.3.2. The security model of a M-ORAM protocol, which is presented in Section 2.4.6, only provides three out of four aspects of security provided by an OGS protocol, leaving out the client-side data secrecy property. In other words, while an OGS protocol allows the clients to impose a fine-grained access control policy for each of their outsourced data item, a M-ORAM protocol allows any client to access the outsourced data items belonging to any other client in the system.

In spite of the missing security guarantee mentioned above, a M-ORAM protocol can preserve the access pattern obliviousness of any honest client even in the face of an the honest-but-curious storage server colluding with the compromised clients. Jinsheng *et al.* [72] allows this adversarial collusion by using a sequence of proxies (*a.k.a.* anonymizers), who are also allowed to collude, between the clients and $\mathcal{S}$. It is this property of security against collusion that makes M-ORAM protocols an indispensable underlying primitive in our OGS protocol construction in Section 5.4.

In order to provide the OGS security guarantees mentioned above while allowing the honest-but-curious server to collude with the unauthorized clients, an OGS protocol makes use of a chain of proxies called anonymizers that facilitate the communication between the clients and $\mathcal{S}$. To allow a stronger level of security, we also allow these anonymizers to be corrupt and to collude with the unauthorized clients and $\mathcal{S}$. However, we require at least one of these anonymizers remain honest.

The formal definition of security of an OGS protocol is also defined in an honest-but-curious adversarial model, where the adversary is allowed to gain as much information as he can with the requirement that he follow the OGS protocol as specified in Definition 5.3.1. We define the OGS security model as a game (OGS-IND-OBC game) which is played between a PPT adversary and a challenger. In this game, the challenger simulates the entire OGS protocol while giving the adversary read access to the private states of the storage server, the revoked clients, and the compromised anonymizers. It is required, however, at least one anonymizer remains honest throughout the game. A secure OGS protocol assures that the adversary's advantage in winning the OGS-IND-OBC game is negligible in the security parameter λ.

Definition 5.3.3 (OGS-IND-OBC Game): For a given oblivious group storage scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Write}, \mathsf{Read})$, the OGS-IND-OBC game, played between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, is defined as follows.

Setup: $\mathcal{C}$ runs $(\mathsf{MPK}, \mathsf{MSK}, ak_1, \ldots, ak_M, st_0) \leftarrow \mathsf{Setup}(1^\lambda, N)$, gives $\mathcal{A}$ the resulting master public key MPK and the initial server state st_0, and keeps the master secret key MSK and the anonymizer keys $ak_1, \ldots, ak_M$ to itself. $\mathcal{C}$ also initializes the sets of revoked clients R_C and compromised anonymizers R_A as empty sets.

Phase 1: $\mathcal{A}$ adaptively issues queries of the following types.

Client secret-key query $\boldsymbol{i}$: $\mathcal{A}$ requests the secret key of a client $i \in C$. $\mathcal{C}$ runs $ck_i \leftarrow \mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)$, adds i to R_C, and gives ck_i to $\mathcal{A}$.

Anonymizer secret-key query $\boldsymbol{j}$: $\mathcal{A}$ requests the secret key of an anonymizer $j \in A$. $\mathcal{C}$ adds j to R_A and gives ak_j, the most recent secret key of the anonymizer j, to $\mathcal{A}$. We do, however, require that at least one anonymizer remain uncompromised for the duration of the OGS-IND-OBC game.

Revoked client write query $\boldsymbol{(i, S, p, m)}$: $\mathcal{A}$ inquires $\mathcal{C}$ to execute the Write algorithm on behalf of a revoked client $i \in R_C$ with a set of target clients $S \subseteq C$, a position $p \in \mathcal{PSP}$, and a message $m \in \mathcal{MSP}$. Then, using the secret key of client i, the most recent anonymizer keys, and the server state, $\mathcal{C}$ simulates the interactive algorithm $\mathsf{Write}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_i, S, p, m, st_t)$ and gives the adversarial view of this execution of Write algorithm to $\mathcal{A}$.

Revoked client read query $\boldsymbol{(i, p)}$: $\mathcal{A}$ asks $\mathcal{C}$ to run the Read algorithm on a position $p \in \mathcal{PSP}$ on behalf of a revoked client $i \in R_C$. $\mathcal{C}$ simulates $\mathsf{Read}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_i, p, st_t)$ using the client i's secret key, the most recent anonymier keys, and the server state; and gives the adversarial view of this execution of Read to $\mathcal{A}$.

Honest client write query $\boldsymbol{(i, S, m)}$: $\mathcal{A}$ asks $\mathcal{C}$ to run the Write algorithm for an honest client $i \in C \setminus R_C$ with a set of target clients $S \subseteq C$, and a message $m \in \mathcal{MSP}$. $\mathcal{C}$, however, does not allow $\mathcal{A}$ to provide a position for this query. Instead, $\mathcal{C}$ picks $p \leftarrow\$ \mathcal{PSP}$ uniformly at random and assigns to it a sequential identifier id. After running $\mathsf{Write}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_i, S, p, m, st_t)$ using the client i's secret key, the most recent anonymizer keys, and the server state, $\mathcal{C}$ gives the adversarial view of this execution of Write and the position identifier id to $\mathcal{A}$.

Honest client read query (i, id): $\mathcal{A}$ inquires $\mathcal{C}$ to run the Read algorithm for an honest client $i \in C \setminus R_C$ on the position associated with the identifier id. $\mathcal{C}$ looks up the position p that she associated with id during a previous honest client write query, simulates $\mathsf{Read}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_i, p, st_t)$ using the client i's secret key, the most recent anonymier keys, and the server state; and provides the adversarial view of this execution of Read to $\mathcal{A}$.

Pre-Challenge: $\mathcal{A}$ gives $\mathcal{C}$ two messages $m_0^*, m_1^* \in \mathcal{MSP}$, two sets of target clients S_0^*, S_1^* such that $R_C \cap (S_0^* \cup S_1^*) = \emptyset$, and a client $i^* \in S_0^* \cap S_1^*$. $\mathcal{C}$ picks two positions $p_0^*, p_1^* \leftarrow\$ \mathcal{PSP}$ and assigns sequential identifiers id_0^*, id_1^* to them, respectively. Next, it runs $\mathsf{Write}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_{i^*}, S_0^*, p_0^*, m_0^*, st_t)$ and $\mathsf{Write}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_{i^*}, S_1^*, p_1^*, m_1^*, st_{t+1})$ using the client i^*'s secret key, the most recent anonymizer keys, and the server state; and gives the adversarial views of these executions of Write and their corresponding position identifiers id_0^*, id_1^* to $\mathcal{A}$.

Phase 2: This phase is similar to *Phase 1* with the exception that $\mathcal{A}$ is not allowed to submit any revocation query for any client $i \in S_0^* \cup S_1^*$. Also, note that $\mathcal{A}$ may submit honest client read queries on the position identifiers id_0^*, id_1^*.

Challenge: $\mathcal{C}$ picks a random bit $b^* \in \{0, 1\}$ and simulates the execution of $\mathsf{Read}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_{i^*}, p_{b^*}^*, st_t)$ using the client i^*'s secret key, the most recent anonymier keys, and the server state. $\mathcal{C}$ also assigns two new sequential position identifiers id_2^*, id_3^* to p_b^*, p_{1-b}^*, respectively. Finally, $\mathcal{C}$ gives the adversarial view of this execution of Read and the two position identifiers id_2^*, id_3^* to the adversary.

Phase 3: This phase is similar to *Phase 2* with the exception that $\mathcal{A}$ is not allowed to submit any honest client read queries on the position identifiers id_0^*, id_1^*. However, $\mathcal{A}$ is allowed to submit such queries on the position identifiers id_2^*, id_3^*

Guess: $\mathcal{A}$ outputs a guess $b \in \{0, 1\}$ and wins if $b = b^*$.

$\mathcal{A}$ is called an OGS-IND-OBC adversary and $\mathcal{A}$'s advantage in winning the above game is defined as

$$\mathsf{Adv}_{\mathcal{A},\Pi}^{\text{OGS-IND-OBC}} := \left|\Pr[b = b^*] - \tfrac{1}{2}\right|,$$

where the probability is over the random coins used by $\mathcal{A}$ and $\mathcal{C}$. $\Diamond$

Definition 5.3.4 (OGS-IND-OBC Security): An OGS scheme Π is $(t, Q_C, Q_A, Q_D, \epsilon)$-OGS-IND-OBC-secure if for any t-time OGS-IND-OBC adversary making at most Q_C, Q_A, and Q_D adaptive client secret-key, anonymizer secret-key, and data access queries, respectively, it is the case that $\mathsf{Adv}_{\mathcal{A},\Pi}^{\text{OGS-IND-OBC}} \leq \epsilon$. $\Diamond$

5.4 Construction

We now present our construction of an oblivious group storage protocol. One can view our approach as a framework that produces a secure OGS protocol by using a secure multi-user oblivious RAM protocol and an outsider-anonymous broadcast encryption scheme as underlying primitives. Consequently, a major advantage of our OGS construction is that it allows one to easily improve its parameters if more efficient underlying primitives are realized in the future. The security of our OGS construction is proven with respect to the OGS-IND-OBC game given in Definition 5.3.3. This proof consists of reduction arguments from the security models of the underlying M-ORAM protocol and the oABE protocol.

Given an outsider-anonymous broadcast encryption protocol $\Pi' = (\mathsf{Setup}', \mathsf{KeyGen}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ and a multi-user oblivious RAM protocol $\Pi'' = (\mathsf{Setup}'', \mathsf{KeyGen}'',$

Write'', Read''), we construct an oblivious group storage protocol $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Write}, \mathsf{Read})$ as follows.

$\mathsf{Setup}(1^\lambda, M, N)$: Setup algorithm is a combination of Setup' and Setup''. First, it obtains $(\mathsf{MPK}', \mathsf{MSK}') \leftarrow \mathsf{Setup}'(1^\lambda, N)$ and $(\mathsf{MPK}'', \mathsf{MSK}'', ak_1, \ldots, ak_M, st_0) \leftarrow \mathsf{Setup}''(1^\lambda, M)$. Next, it sets $\mathsf{MPK} := (\mathsf{MPK}', \mathsf{MPK}'')$ and $\mathsf{MSK} := (\mathsf{MSK}', \mathsf{MSK}'')$. Finally, it outputs $(\mathsf{MPK}, \mathsf{MSK}, ak_1, \ldots, ak_M, st_0)$.

$\mathsf{KeyGen}(\mathsf{MPK}, \mathsf{MSK}, i)$: KeyGen algorithm is a combination of KeyGen' and KeyGen''. First, it computes $sk_i \leftarrow \mathsf{KeyGen}'(\mathsf{MPK}', \mathsf{MSK}', i)$ and $ck''_i \leftarrow \mathsf{KeyGen}''(\mathsf{MPK}'', \mathsf{MSK}'', i)$. Then, it sets $ck_i := (sk_i, ck''_i)$ and outputs ck_i.

$\mathsf{Write}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_i, S, p, m, st_t)$: Write algorithm is a composition of $\mathsf{Encrypt}'$ and Write''. First, it computes the oABE ciphertext $c \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', S, m)$. Then, it interactively computes $(ak'_1, \ldots, ak'_M, st_{t+1}) \leftarrow \mathsf{Write}''(\mathsf{MPK}'', ak_1, \ldots, ak_M, ck''_i, p, c, st_t)$ with client i, the anonymizers, and $\mathcal{S}$. Note that at the end of Write'', the server state at $\mathcal{S}$ gets updated to st_{t+1} and the anonymizer keys get updated to $ak'_1, \ldots, ak'_M$.

$\mathsf{Read}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_i, p, st_t)$: Read algorithm is a composition of Read'' and $\mathsf{Decrypt}'$. First it interactively computes $(c, ak'_1, \ldots, ak'_M, st_{t+1}) \leftarrow \mathsf{Read}''(\mathsf{MPK}'', ak_1, \ldots, ak_M, ck''_i, p, st_t)$ with client i, the anonymizers, and $\mathcal{S}$. During the execution of Read'', the server state at $\mathcal{S}$ gets updated to st_{t+1} and the anonymizer keys get updated to $ak'_1, \ldots, ak'_M$. Finally, it computes $m := \mathsf{Decrypt}'(\mathsf{MPK}', sk_i, c)$ and returns m to client i.

The security of the above construction tightly depends on the underlying oABE and M-ORAM schemes. According to Section 3.3.2, there exists two types of oABE schemes with respect to the provided level of security, namely oABE-IND-CPA security and oABE-IND-CCA security. In our OGS construction, we require Π' to be oABE-

IND-CPA-secure. We also require the M-ORAM scheme Π'' to be secure with respect to the M-ORAM-IND-OBC model of security presented in Definition 2.4.25.

The following theorem summarizes the security of our OGS construction. The proof of this theorem consists of a sequence of three hybrid games, where each is based on the one given in Definition 5.3.3. We prove that each consecutive pair of these games is indistinguishable by reduction arguments to the security of the underlying oABE or M-ORAM schemes.

Theorem 5.4.1: *If the oABE scheme* $\Pi' = (\mathsf{Setup}', \mathsf{KeyGen}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ *is* $(t, Q_C, Q_D, \epsilon_1)$*-oABE-IND-CPA-secure and the M-ORAM protocol* $\Pi'' = (\mathsf{Setup}'', \mathsf{KeyGen}'', \mathsf{Write}'', \mathsf{Read}'')$ *is* $(t, Q_C, Q_A, Q_D, \epsilon_2)$*-M-ORAM-IND-OBC-secure, then the construction given above is* $(t, Q_C, Q_A, Q_D, 2\epsilon_1 + \epsilon_2)$*-OGS-IND-OBC-secure.* □

Proof. We organize our proof as a sequence of games, Game$_0$, Game$_1$, Game$_2$, and Game$_3$, played between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Each of these games is a slightly modified version of the OGS-IND-OBC game given in Definition 5.3.3. During the *Challenge* step of Game$_0$, $\mathcal{C}$ always considers the position p_0^*, whereas in Game$_3$, she always considers p_1^*. Game$_1$ and Game$_2$ are hybrid games that lie in between Game$_0$ and Game$_3$. We prove that $\mathcal{A}$ is unable to distinguish between the pair Game$_0$ and Game$_1$ as well as the pair Game$_2$ and Game$_3$ using reduction arguments to the security of the underlying oABE scheme. Similarly, we prove that Game$_1$ and Game$_2$ are indistinguishable to $\mathcal{A}$, but this time reducing to the security of the M-ORAM scheme. Given below are the details of our proof.

Game$_0$: This game corresponds to the OGS-IND-OBC game given in Definition 5.3.3 when the challenge bit b^* is fixed to 0. The interaction between $\mathcal{A}$ and $\mathcal{C}$ during *Setup*, *Phase 1*, *Pre-Challenge*, *Phase 2*, and *Phase 3* steps follow exactly as stated in Definition 5.3.3.

During the *Challenge* step, $\mathcal{C}$ simulates the execution of $\mathsf{Read}(\mathsf{MPK}, ak_1, \ldots,$

$ak_M, ck_{i^*}, p_0^*, st_t)$ using the secret key of client i^* that the adversary submitted to her during *Pre-Challenge* step, the most recent anonymier keys, and the server state. Then, $\mathcal{C}$ assigns two new sequential position identifiers id_2^*, id_3^* to p_0^*, p_1^*, respectively. At the end, $\mathcal{C}$ gives the adversarial view of this execution of Read and the two position identifiers id_2^*, id_3^* to $\mathcal{A}$.

Finally, $\mathcal{A}$ submits his guess b to $\mathcal{C}$ and wins the game if $b = 0$.

Game$_1$: This game is a slightly modified version of Game$_0$. Except for the *Pre-Challenge* step, the interaction between $\mathcal{C}$ and $\mathcal{A}$ during this game is the same as in Game$_0$.

During the *Pre-Challenge* step, $\mathcal{A}$ gives to $\mathcal{C}$ two messages $m_0^*, m_1^* \in \mathcal{MSP}$, two sets of target clients $S_0^*, S_1^* \subseteq C \setminus R_C$, and a client $i^* \in S_0^* \cap S_1^*$. Next, after picking two positions $p_0^*, p_1^* \leftarrow\$ \mathcal{PSP}$ and assigning sequential identifiers id_0^*, id_1^* to them, respectively, $\mathcal{C}$ runs $\mathsf{Write}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_{i^*}, S_1^*, p_0^*, m_1^*, st_t)$ and $\mathsf{Write}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_{i^*}, S_1^*, p_1^*, m_1^*, st_{t+1})$ using the client i^*'s secret key, the most recent anonymizer keys, and the server state. At last, $\mathcal{C}$ gives the adversarial views of these executions of Write and their corresponding position identifiers id_0^*, id_1^* to $\mathcal{A}$.

At the end, $\mathcal{A}$ outputs his guess b and wins if $b = 0$.

Game$_2$: The interaction between $\mathcal{C}$ and $\mathcal{A}$ during Game$_2$ is similar to Game$_1$, except for the *Challenge* step. During the *Challenge* step, $\mathcal{C}$ simulates the execution of $\mathsf{Read}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_{i^*}, p_1^*, st_t)$ using the secret key of client i^* that the adversary submitted during *Pre-Challenge* step, the most recent anonymier keys, and the server state. After assigning two new sequential position identifiers id_2^*, id_3^* to p_0^*, p_1^*, respectively, $\mathcal{C}$ gives the adversarial view of this execution of Read and the two position identifiers id_2^*, id_3^* to $\mathcal{A}$.

Finally, $\mathcal{A}$ outputs his guess b and wins the game if $b = 0$.

Game$_3$: This game is similar to Game$_2$, except for the *Pre-Challenge* step. During the *Pre-Challenge* step, $\mathcal{A}$ sends $\mathcal{C}$ two messages $m_0^*, m_1^* \in \mathcal{MSP}$, two sets of target clients $S_0^*, S_1^* \subseteq C \setminus R_C$, and a client $i^* \in S_0^* \cap S_1^*$. Then, $\mathcal{C}$ picks two positions $p_0^*, p_1^* \leftarrow\$ \mathcal{PSP}$ and assigns sequential identifiers id_0^*, id_1^* to them, respectively. Next, $\mathcal{C}$ runs $\mathsf{Write}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_{i^*}, S_0^*, p_0^*, m_0^*, st_t)$ and $\mathsf{Write}(\mathsf{MPK}, ak_1, \ldots, ak_M, ck_{i^*}, S_1^*, p_1^*, m_1^*, st_{t+1})$ using the client i^*'s secret key, the most recent anonymizer keys, and the server state. Finally, $\mathcal{C}$ gives the adversarial views of these executions of Write and their corresponding position identifiers id_0^*, id_1^* to $\mathcal{A}$.

At the end, $\mathcal{A}$ sends his guess b to $\mathcal{C}$ and wins the game if $b = 0$.

Let $\mathsf{Adv}^i_{\mathcal{A},\Pi}$ denote $\mathcal{A}$'s advantage of winning Game$_i$. In Lemma 5.4.2 (respectively Lemma 5.4.4), we prove that if the underlying oABE scheme is $(t, Q_C, Q_D, \epsilon_1)$-oABE-IND-CPA-secure then the adversary's advantage in distinguishing Game$_0$ from Game$_1$ (respectively Game$_2$ from Game$_3$) is at most ϵ_1. In Lemma 5.4.3, we show that if the underlying M-ORAM scheme is $(t, Q_C, Q_A, Q_D, \epsilon_2)$-M-ORAM-IND-OBC-secure then $\mathcal{A}$'s advantage in distinguishing Game$_1$ from Game$_2$ is at most ϵ_2. Therefore,

$$\left|\mathsf{Adv}^0_{\mathcal{A},\Pi} - \mathsf{Adv}^3_{\mathcal{A},\Pi}\right| \leq 2\epsilon_1 + \epsilon_2. \qquad \blacksquare$$

Lemma 5.4.2: *If the underlying outsider-anonymous broadcast encryption scheme* Π' *is* $(t, Q_C, Q_D, \epsilon_1)$*-oABE-IND-CPA-secure, then* $\mathcal{A}$*'s advantage of distinguishing Game*$_0$ *from Game*$_1$ *is at most* ϵ_1. *In other words,*

$$\left|\mathsf{Adv}^0_{\mathcal{A},\Pi} - \mathsf{Adv}^1_{\mathcal{A},\Pi}\right| \leq \epsilon_1. \qquad \square$$

Proof. We build a PPT adversary $\mathcal{B}$ that runs the oABE-IND-CPA game with her challenger $\mathcal{C}'$ as follows. After receiving the master public key MPK' of the oABE scheme from $\mathcal{C}'$, $\mathcal{B}$ executes the OGS-IND-OBC game with $\mathcal{A}$ in order to gain advantage in the OGS-IND-CPA game. The details of the interaction among the adversaries $\mathcal{C}'$, $\mathcal{B}$, and $\mathcal{A}$ are explained below.

Setup: First, $\mathcal{B}$ executes $(\mathsf{MPK}'', \mathsf{MSK}'', ak_1, \ldots, ak_M, st_0) \leftarrow \mathsf{Setup}''(1^\lambda, M)$ and sets $\mathsf{MPK} := (\mathsf{MPK}', \mathsf{MPK}'')$. Next, she gives MPK and st_0 to $\mathcal{A}$ and keeps $ak_1, \ldots, ak_M$ to herself. $\mathcal{B}$ also initializes the sets of revoked clients R_C and compromised anonymizers R_A as empty sets.

Phase 1: $\mathcal{B}$ replies to $\mathcal{A}$'s queries as follows.

Client secret-key query $\boldsymbol{i}$: First, $\mathcal{B}$ sends a secret key query to $\mathcal{C}'$ and obtains the oABE secret key sk_i of client i. Then, $\mathcal{B}$ computes $ck''_i \leftarrow \mathsf{KeyGen}''(\mathsf{MPK}'', \mathsf{MSK}'', i)$, sets $ck_i := (sk_i, ck''_i)$, adds i to R_C, and finally returns ck_i to $\mathcal{A}$.

Anonymizer secret-key query $\boldsymbol{j}$: $\mathcal{B}$ adds j to R_A and gives ak_j to $\mathcal{A}$. Note that at least one anonymizer should remain uncompromised.

Revoked client write query $\boldsymbol{(i, S, p, m)}$: First, $\mathcal{B}$ computes the oABE ciphertext $c \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', S, m)$. Then, she simulates $\mathsf{Write}''(\mathsf{MPK}'', ak_1, \ldots, ak_M, ck''_i, p, c, st_t)$ and gives its adversarial view to $\mathcal{A}$. Since $\mathcal{A}$ has compromised client i, $\mathcal{A}$ is allowed to see any private information visible to i, and as a result, c is part of the adversarial view of Write as well. Therefore, $\mathcal{B}$ also provides the ciphertext c to $\mathcal{A}$.

Revoked client read query $\boldsymbol{(i, p)}$: First, $\mathcal{B}$ simulates $(c, ak'_1, \ldots, ak'_M, st_{t+1}) \leftarrow \mathsf{Read}''(\mathsf{MPK}'', ak_1, \ldots, ak_M, ck''_i, p, st_t)$. After that, she computes $m := \mathsf{Decrypt}'(\mathsf{MPK}', sk_i, c)$. Finally, she gives the adversarial view of this execution of Read'', c, and m to $\mathcal{A}$. The reason for returning c and m is the same as for giving c in the revoked client write query above.

Honest client write query (i, S, m): $\mathcal{B}$ picks $p \leftarrow\$ \mathcal{PSP}$ and assigns to it a sequential identifier id. Then, she computes $ck''_i \leftarrow \mathsf{KeyGen}''(\mathsf{MPK}'', \mathsf{MSK}'', i)$ if she has not computed ck''_i before. Next, she computes $c \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', S, m)$ and simulates $\mathsf{Write}''(\mathsf{MPK}, ak_1, \ldots, ak_M, ck''_i, p, c, st_t)$. Finally, she gives the adversarial view of this execution of Write'' and the position identifier id to $\mathcal{A}$. $\mathcal{B}$ does not provide c to $\mathcal{A}$ since $\mathcal{A}$ has not corrupted i.

Honest client read query (i, id): $\mathcal{B}$ first maps id to the corresponding position p. Then, she computes $ck''_i \leftarrow \mathsf{KeyGen}''(\mathsf{MPK}'', \mathsf{MSK}'', i)$ if she has not previously computed ck''_i. Next, she simulates $(c, ak'_1, \ldots, ak'_M, st_{t+1}) \leftarrow \mathsf{Read}''(\mathsf{MPK}'', ak_1, \ldots, ak_M, ck''_i, p, st_t)$ and returns the adversarial view of this execution to $\mathcal{A}$. Since neither c nor the message m encrypted in c is part of the adversarial view, $\mathcal{B}$ does not decrypt c or return c to $\mathcal{A}$.

Pre-Challenge: $\mathcal{B}$ receives two messages $m^*_0, m^*_1 \in \mathcal{MSP}$, two sets of target clients $S^*_0, S^*_1 \subseteq C \setminus R_C$, and a client $i^* \in S^*_0 \cap S^*_1$ from $\mathcal{A}$. Then, she submits m^*_0, m^*_1 and S^*_0, S^*_1 as her challenge query to $\mathcal{C}'$. $\mathcal{C}'$ picks a random bit $b' \in \{0, 1\}$ and sends $c' \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', S^*_{b'}, m^*_{b'})$ to $\mathcal{B}$. Next, $\mathcal{B}$ computes $c \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', S^*_1, m^*_1)$, $ck''_{i^*} \leftarrow \mathsf{KeyGen}''(\mathsf{MPK}'', \mathsf{MSK}'', i^*)$, and picks two positions $p^*_0, p^*_1 \leftarrow\$ \mathcal{PSP}$ and assigns sequential identifiers id^*_0, id^*_1 to them, respectively. Next, it runs $\mathsf{Write}''(\mathsf{MPK}'', ak_1, \ldots, ak_M, ck''_{i^*}, p^*_0, c', st_t)$ and $\mathsf{Write}''(\mathsf{MPK}'', ak_1, \ldots, ak_M, ck''_{i^*}, p^*_1, c, st_{t+1})$ and gives the adversarial views of these executions of Write'' and their corresponding position identifiers id^*_0, id^*_1 to $\mathcal{A}$.

Phase 2: $\mathcal{B}$ responds to $\mathcal{A}$'s queries similar to *Phase 1* with the exception that $\mathcal{A}$ is not allowed to submit any revocation query for any client $i \in S^*_0 \cup S^*_1$.

Challenge: $\mathcal{B}$ simulates the execution of $(c, ak'_1, \ldots, ak'_M, st_{t+1}) \leftarrow \mathsf{Read}''(\mathsf{MPK}'', ak_1, \ldots, ak_M, ck''_{i^*}, p^*_0, st_t)$. Next, she assigns two new sequential position identifiers id^*_2, id^*_3 to p^*_0, p^*_1, respectively. Finally, she returns the adversarial view of

Read'', id_2^*, and id_3^* to $\mathcal{A}$. Note that $\mathcal{B}$ does not provide c to $\mathcal{A}$ or even attempt to decrypt c because its not part of the adversarial view.

Phase 3: $\mathcal{B}$ responds to $\mathcal{A}$'s queries as in *Phase 2* with the usual exception that $\mathcal{A}$ is not authorized to submit any honest client read queries on the position identifiers id_0^*, id_1^*.

Guess: $\mathcal{A}$ outputs a guess b and $\mathcal{B}$ passes it to $\mathcal{C}'$.

Observe that if $\mathcal{C}'$ chooses $b' = 0$, then $\mathcal{B}$ is playing Game$_0$, whereas if $b' = 1$, then $\mathcal{B}$ is playing Game$_1$. Therefore, $\mathcal{B}$'s oABE-IND-CPA advantage is equivalent to $\mathcal{A}$'s advantage in distinguishing Game$_0$ from Game$_1$. More formally,

$$\left|\mathsf{Adv}^0_{\mathcal{A},\Pi} - \mathsf{Adv}^1_{\mathcal{A},\Pi}\right| \leq \epsilon_1. \qquad \blacksquare$$

Lemma 5.4.3: *If the underlying multi-user oblivious RAM protocol* Π'' *is* $(t, Q_C, Q_A, Q_D, \epsilon_2)$*-M-ORAM-IND-OBC-secure, then* $\mathcal{A}$*'s advantage of distinguishing Game*$_1$ *from Game*$_2$ *is at most* ϵ_2*. In other words,*

$$\left|\mathsf{Adv}^1_{\mathcal{A},\Pi} - \mathsf{Adv}^2_{\mathcal{A},\Pi}\right| \leq \epsilon_2. \qquad \square$$

Proof. We build a PPT adversary $\mathcal{B}$ that runs the M-ORAM-IND-OBC game with its challenger $\mathcal{C}''$ as follows. After receiving, the master public key MPK'' and the initial server state st_0 from $\mathcal{C}''$, $\mathcal{B}$ executes the OGS-IND-OBC game with $\mathcal{A}$ in order to gain advantage in the M-ORAM-IND-OBC game. Given below are the details.

Setup: First, $\mathcal{B}$ computes $(\mathsf{MPK}', \mathsf{MSK}') \leftarrow \mathsf{Setup}'(1^\lambda, N)$ and initializes $\mathsf{MPK} := (\mathsf{MPK}', \mathsf{MPK}'')$. Next, she gives MPK and st_0 to $\mathcal{A}$. $\mathcal{B}$ also initializes the sets of revoked clients R_C and compromised anonymizers R_A as empty sets.

Phase 1: $\mathcal{B}$ replies to $\mathcal{A}$'s queries as follows.

Client secret-key query $\boldsymbol{i}$: First, $\mathcal{B}$ sends a client secret-key query to $\mathcal{C}''$ and obtains the M-ORAM secret key ck''_i of client i. Then, she computes $sk'_i \leftarrow \mathsf{KeyGen}'(\mathsf{MPK}', \mathsf{MSK}', i)$, sets $ck_i := (sk_i, ck''_i)$, adds i to R_C, and returns ck_i to $\mathcal{A}$.

Anonymizer secret-key query $\boldsymbol{j}$: $\mathcal{B}$ first adds j to R_A. Next, she submits an anonymizer secret-key query to $\mathcal{C}''$, obtains obtains ak_j of anonymizer j, and eventually passes it to $\mathcal{A}$. Note that at least one anonymizer should remain uncompromised.

Revoked client write query $\boldsymbol{(i, S, p, m)}$: First, $\mathcal{B}$ computes the oABE ciphertext $c \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', S, m)$. Then, she submits a revoked client write query (i, p, c) to $\mathcal{C}''$, obtains the adversarial view from $\mathcal{C}''$, and passes it to $\mathcal{A}$. She also returns the ciphertext c to $\mathcal{A}$ because he has compromised client i and, as a result, he is allowed to see any private information visible to client i.

Revoked client read query $\boldsymbol{(i, p)}$: $\mathcal{B}$ first submits a revoked client read query (i, p) to $\mathcal{C}''$ and obtains the adversarial view and the oABE ciphertext c. Next, she computes $m := \mathsf{Decrypt}'(\mathsf{MPK}', sk_i, c)$ and passes the adversarial view, c, and m to $\mathcal{A}$. The reason for returning c and m to $\mathcal{A}$ is the same as for return c in the previous type of query.

Honest client write query $\boldsymbol{(i, S, m)}$: First, $\mathcal{B}$ computes the oABE ciphertext $c \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', S, m)$. Then, she sends an honest client write query (i, c) to $\mathcal{C}''$, obtains the adversarial view and a position identifier id, and passes them to $\mathcal{A}$. $\mathcal{B}$ does not provide c to $\mathcal{A}$ since $\mathcal{A}$ has not compromised the client i.

Honest client read query $\boldsymbol{(i, id)}$: $\mathcal{B}$ sends an honest client query (i, id) to $\mathcal{C}''$, obtains the adversarial view, and passes it to $\mathcal{A}$.

Pre-Challenge: $\mathcal{A}$ sends to $\mathcal{B}$ two messages $m_0^*, m_1^* \in \mathcal{MSP}$, two sets of target clients $S_0^*, S_1^* \subseteq C \setminus R_C$, and a client $i^* \in S_0^* \cap S_1^*$. $\mathcal{B}$ generates $c_0^* \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', S_0^*, m_0^*)$, $c_1^* \leftarrow \mathsf{Encrypt}'(\mathsf{MPK}', S_1^*, m_1^*)$ and sends (c_0^*, c_1^*, i^*) as her *Pre-Challenge* query to $\mathcal{C}''$. $\mathcal{C}''$ picks two positions $p_0^*, p_1^* \leftarrow\$ \mathcal{PSP}$, assigns sequential identifiers id_0^*, id_1^* to them, respectively, it runs $\mathsf{Write}''(\mathsf{MPK}'', ak_1, \ldots, ak_M, ck''_{i^*}, p_0^*, c_0^*, st_t)$ and $\mathsf{Write}''(\mathsf{MPK}'', ak_1, \ldots, ak_M, ck''_{i^*}, p_1^*, c_1^*, st_{t+1})$, and returns the adversarial views of these executions of Write'', id_0^*, and id_1^* to $\mathcal{B}$. $\mathcal{B}$ forwards all these values returned by $\mathcal{C}''$ to $\mathcal{A}$.

Phase 2: $\mathcal{B}$ responds to $\mathcal{A}$'s queries similar to *Phase 1* with the exception that $\mathcal{A}$ is not allowed to submit any revocation query for any client $i \in S_0^* \cup S_1^*$.

Challenge: $\mathcal{C}''$ picks a random bit $b'' \in \{0,1\}$, simulates the execution of $\mathsf{Read}''(\mathsf{MPK}'', ak_1, \ldots, ak_M, ck''_{i^*}, p_{b''}^*, st_t)$, assigns two new sequential position identifiers id_2^*, id_3^* to $p_{b''}^*, p_{1-b''}^*$, respectively, and sends the adversarial view of of the execution of Read'', id_2^*, and id_3^* to $\mathcal{B}$. $\mathcal{B}$ forwards this adversarial view, id_2^*, and id_3^* to $\mathcal{A}$.

Phase 3: $\mathcal{B}$ responds to $\mathcal{A}$'s queries as in *Phase 2* with the exception that $\mathcal{A}$ is not allowed to submit any honest client read queries on the position identifiers id_0^*, id_1^*.

Guess: $\mathcal{A}$ outputs a guess b and $\mathcal{B}$ sends it to $\mathcal{C}''$.

Note that if $\mathcal{C}''$ chooses $b'' = 0$, then $\mathcal{B}$ is playing Game$_1$, whereas if $b'' = 1$, then $\mathcal{B}$ is playing Game$_2$. Therefore, $\mathcal{B}$'s M-ORAM-IND-OBC advantage is equivalent to $\mathcal{A}$'s advantage in distinguishing Game$_1$ from Game$_2$. More formally,

$$\left|\mathsf{Adv}^1_{\mathcal{A},\Pi} - \mathsf{Adv}^2_{\mathcal{A},\Pi}\right| \leq \epsilon_2. \qquad \blacksquare$$

Lemma 5.4.4: *If the underlying outsider-anonymous broadcast encryption scheme Π' is $(t, Q_C, Q_D, \epsilon_1)$-oABE-IND-CPA-secure, then $\mathcal{A}$'s advantage of distinguishing Game$_2$*

from Game$_3$ is at most ϵ_1. More precisely,

$$\left|\mathsf{Adv}^{2}_{\mathcal{A},\Pi} - \mathsf{Adv}^{3}_{\mathcal{A},\Pi}\right| \leq \epsilon_1. \qquad \square$$

Proof. The argument is analogous to the proof of Lemma 5.4.2. ■

Bibliography

[1] AACS. Advanced access content system. `http://www.aacsla.com/`.

[2] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to Anonymous IBE, and extensions. In *Advances in Cryptology—CRYPTO*, pages 205–222, 2005.

[3] M. Abdalla, M. Bellare, and G. Neven. Robust encryption. In *Theory of Cryptography—TCC*, pages 480–497, 2010.

[4] S. Agrawal and X. Boyen. Identity-based encryption from lattices in the standard model. Manuscript, 2009. `http://www.cs.stanford.edu/~xb/ab09/`.

[5] M. Ajtai. Oblivious rams without cryptogrpahic assumptions. In *ACM Symposium on Theory of Computing—STOC*, pages 181–190, 2010.

[6] Amazon.com Inc. Amazon s3. `http://aws.amazon.com/s3/`.

[7] R. Anderson and F. Petitcolas. On the limits of steganography. *IEEE Journal on Selected Areas in Communications*, 16(4):474–481, May 1998.

[8] G. Ateniese and P. Gasti. Universally anonymous ibe based on the quadratic residuosity assumption. In *Topics in Cryptology—CT-RSA*, pages 32–47, 2009.

[9] M. Backes and C. Cachin. Public-key steganography with active attacks. In *Theory of Cryptography—TCC*, pages 210–226, 2005.

[10] A. Barth, D. Boneh, and B. Waters. Privacy in encrypted content distribution using private broadcast encryption. In *Financial Cryptography and Data Security—FC*, pages 52–64, 2006.

[11] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In *Advances in Cryptology—ASIACRYPT*, pages 566–582, 2001.

[12] S. Berkovits. How to broadcast a secret. In *Advances in Cryptology—EUROCRYPT*, pages 535–541, 1991.

[13] D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In *Advances in Cryptology—CRYPTO*, pages 443–459, 2004.

[14] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology—EUROCRYPT*, pages 440–456, 2005.

[15] D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Advances in Cryptology—EUROCRYPT*, pages 506–522, 2004.

[16] D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology—CRYPTO*, pages 213–229, 2001.

[17] D. Boneh, C. Gentry, and M. Hamburg. Space-efficient identity based encryption without pairings. In *IEEE Symposium on Foundations of Computer Science—FOCS*, pages 647–657, 2007.

[18] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology—CRYPTO*, pages 258–275, 2005.

[19] D. Boneh, E.-J. Goh, N. Modadugu, and H. Shacham. Sirius: Securing remote untrusted storage. In *ISOC Network and Distributed System Security Symposium—NDSS*, pages 131–145, 2003.

[20] D. Boneh and J. Katz. Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. In *Topics in Cryptology—CT-RSA*, pages 87–103, 2005.

[21] D. Boneh, D. Mazieres, and R. A. Popa. Remote oblivious storage: Making oblivious RAM practical. Manuscript, 2011. `http://dspace.mit.edu/handle/1721.1/62006`.

[22] D. Boneh and B. Waters. A fully collusion resistant broadcast, trace, and revoke system. In *ACM Conference on Computer and Communications Security—CCS*, pages 211–220, 2006.

[23] D. Boneh, B. Waters, and M. Zhandry. Low overhead broadcast encryption from multilinear maps. In *Advances in Cryptology—CRYPTO*, pages 206–223, 2014.

[24] X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Advances in Cryptology—CRYPTO*, pages 290–307, 2006.

[25] E. Boyle, K. Chung, and R. Pass. Oblivious parallel RAM. Cryptology ePrint Archive, Report 2014/594, 2014.

[26] C. Cachin. An information-theoretic model for steganography. *Information and Computation*, 192(1):41–56, 2004.

[27] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. *Journal of Cryptology*, 20(3):265–294, 2007.

[28] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. *Journal of Cryptology*, 25(4):601–639, 2012.

[29] D. Cash, E. Kiltz, and V. Shoup. The twin Diffie-Hellman problem and applications. In *Advances in Cryptology—EUROCRYPT*, pages 127–145, 2008.

[30] Centers for Medicare & Medicaid Services. The Health Insurance Portability and Accountability Act of 1996 (HIPAA), 1996. `http://www.cms.hhs.gov/hipaa/`.

[31] N. Chandran, V. Goyal, R. Ostrovsky, and A. Sahai. Covert multi-party computation. In *IEEE Symposium on Foundations of Computer Science—FOCS*, pages 238–248, 2007.

[32] S. Chen, R. Wang, X. Wang, and K. Zhang. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In *IEEE Symposium on Security and Privacy—IEEE S&P*, pages 191–206, 2010.

[33] Y. Chen, Z. Chen, J. Hu, and S. Luo. New fully secure hierarchical identity-based encryption with constant size ciphertexts. In *Information Security Practice and Experience—ISPEC*, pages 55–70, 2011.

[34] J. H. Cheon and J. H. Seo. Fully secure anonymous hierarchical identity-based encryption with constant size ciphertexts. Cryptology ePrint Archive, Report 2011/021, 2011.

[35] K.-M. Chung, Z. Liu, and R. Pass. Statistically-secure oram with $\tilde{O}(\log^2 n)$ overhead. In *Advances in Cryptology—ASIACRYPT*, pages 62–81, 2014.

[36] I. Damgård, S. Meldgaard, and J. Nielsen. Perfectly secure oblivious ram without random oracles. In *Theory of Cryptography—TCC*, pages 144–163, 2011.

[37] A. De Caro, V. Iovino, and G. Persiano. Fully secure anonymous hibe and secret-key anonymous ibe with short ciphertexts. In *Pairing-Based Cryptography—Pairing*, pages 347–366, 2010.

[38] N. Dedic, G. Itkis, L. Reyzin, and S. Russell. Upper and Lower Bounds on Black-Box Steganography. *Journal of Cryptology*, 22(3):365–394, 2009.

[39] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, Sept. 1976.

[40] Y. Dodis and N. Fazio. Public-key broadcast encryption for stateless receivers. In *Digital Rights Management—DRM*, pages 61–80, 2002.

[41] Y. Dodis and N. Fazio. Public-key trace and revoke scheme secure against adaptive chosen ciphertext attack. In *Public Key Cryptography—PKC*, pages 100–115, 2003.

[42] Y. Dodis, N. Fazio, A. Kiayias, and M. Yung. Scalable public-key tracing and revoking. In *ACM Symposium on Principles of Distributed Computing—PODC*, pages 190–199, 2003. Invited to the Special Issue of Journal of Distributed Computing PODC 2003.

[43] Y. Dodis and J. Katz. Chosen-ciphertext security of multiple encryption. In *Theory of Cryptography—TCC*, pages 188–209, 2005.

[44] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, Apr. 2000.

[45] Dropbox Inc. Sharing files and folders. `https://www.dropbox.com/help/category/Sharing`.

[46] L. Ducas. Anonymity from asymmetry: New constructions for anonymous hibe. In *Topics in Cryptology—CT-RSA*, pages 148–164, 2010.

[47] N. Fazio, A. R. Nicolosi, and I. M. Perera. Broadcast steganography. In *Topics in Cryptology—CT-RSA*, pages 64–84, 2014.

[48] N. Fazio and I. M. Perera. Outsider-anonymous broadcast encryption with sublinear ciphertexts. In *Public Key Cryptography—PKC*, pages 225–242, 2012.

[49] A. Fiat and M. Naor. Broadcast encryption. In *Advances in Cryptology—CRYPTO*, pages 480–491, 1993.

[50] J. A. Garay, J. Staddon, and A. Wool. Long-lived broadcast encryption. In *Advances in Cryptology—CRYPTO*, pages 333–352, 2000.

[51] C. Gentry. Practical identity-based encryption without random oracles. In *Advances in Cryptology—EUROCRYPT*, pages 445–464, 2006.

[52] C. Gentry and S. Halevi. Hierarchical identity based encryption with polynomially many levels. In *Theory of Cryptography—TCC*, pages 437–456, 2009.

[53] C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In *Advances in Cryptology—ASIACRYPT*, pages 548–566, 2002.

[54] C. Gentry and B. Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In *Advances in Cryptology—EUROCRYPT*, pages 171–188, 2009.

[55] O. Goldreich. Towards a theory of software protection and simulation by oblivious rams. In *ACM Symposium on Theory of Computing—STOC*, pages 182–194, 1987.

[56] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious rams. *Journal of the ACM—JACM*, 43(3):431–473, May 1996.

[57] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[58] M. T. Goodrich and M. Mitzenmacher. Privacy-preserving access of outsourced data via oblivious ram simulation. In *Automata, Languages and Programming—ICALP*, pages 576–587, 2011.

[59] M. T. Goodrich, M. Mitzenmacher, O. Ohrimenko, and R. Tamassia. Oblivious ram simulation with efficient worst-case access overhead. In *ACM Workshop on Cloud Computing Security Workshop—CCSW*, pages 95–100, 2011.

[60] M. T. Goodrich, M. Mitzenmacher, O. Ohrimenko, and R. Tamassia. Practical oblivious storage. In *ACM Conference on Data and Application Security and Privacy—CODASPY*, pages 13–24, 2012.

[61] M. T. Goodrich, M. Mitzenmacher, O. Ohrimenko, and R. Tamassia. Privacy-preserving group data access via stateless oblivious ram simulation. In *ACM-SIAM Symposium on Discrete Algorithms—SODA*, pages 157–167, 2012.

[62] Google Inc. Google cloud platform. `https://cloud.google.com/`.

[63] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security—CCS*, pages 89–98, 2006.

[64] D. Halevy and A. Shamir. The LSD broadcast encryption scheme. In *Advances in Cryptology—CRYPTO*, pages 47–60, 2002.

[65] N. J. Hopper. *Toward a Theory of Steganography.* PhD thesis, Carnegie Mellon University, 2004.

[66] N. J. Hopper. On steganographic chosen covertext security. In *Automata, Languages and Programming—ICALP*, pages 311–323, 2005.

[67] N. J. Hopper, J. Langford, and L. von Ahn. Provably Secure Steganography. In *Advances in Cryptology—CRYPTO*, pages 77–92, 2002.

[68] J. Horwitz and B. Lynn. Toward hierarchical identity-based encryption. In *Advances in Cryptology—EUROCRYPT*, pages 466–481, 2002.

[69] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *IEEE Symposium on Foundations of Computer Science—FOCS*, pages 248–253, 1989.

[70] M. S. Islam, M. Kuzu, and M. Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *Network and Distributed System Security Symposium—NDSS*, 2012.

[71] S. Jarecki and X. Liu. Unlinkable secret handshakes and key-private group key management schemes. In *Applied Cryptography and Network Security—ACNS*, pages 270–287, 2007.

[72] Z. Jinsheng, Z. Wensheng, and D. Qiao. A multi-user oblivious ram for outsourced data. Manuscript, 2014.

[73] J. Katz and Y. Lindell. *Introduction to Modern Cryptography.* Chapman & Hall/CRC, 2007.

[74] A. Kiayias, Y. Raekow, and A. Russell. Efficient steganography with provable security guarantees. In *Information Hiding—IH*, pages 118–130, 2005.

[75] A. Kiayias, A. Russell, and N. Shashidhar. Key-efficient steganography with provable security guarantees. In *Information Hiding—IH*, pages 118–130, 2012.

[76] A. Kiayias and K. Samari. Lower bounds for private broadcast encryption. In *Information Hiding—IH*, pages 176–190, 2012.

[77] L. Krzywiecki, P. Kubiak, and M. Kutylowski. A revocation scheme preserving privacy. In *Information Security and Cryptology—Inscrypt*, pages 130–143, 2006.

[78] E. Kushilevitz, S. Lu, and R. Ostrovsky. On the (in)security of hash-based oblivious ram and a new balancing scheme. In *ACM-SIAM Symposium on Discrete Algorithms—SODA*, pages 143–156, 2012.

[79] L. Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, 1979.

[80] T. Le and K. Kurosawa. Efficient Public Key Steganography Secure Against Adaptive Chosen Stegotext Attacks. Cryptology ePrint Archive, Report 2003/244, 2003.

[81] D. H. Lee and K. Lee. New techniques for anonymous hibe with short ciphertexts in prime order groups. *KSII Transactions on Internet and Information Systems (TIIS)*, 4(5):968–988, 2010.

[82] A. B. Lewko and B. Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In *Theory of Cryptography—TCC*, pages 455–479, 2010.

[83] B. Libert, K. G. Paterson, and E. A. Quaglia. Anonymous broadcast encryption. In *Public Key Cryptography—PKC*, pages 206–224, 2012.

[84] S. Lu and R. Ostrovsky. Distributed oblivious ram for secure two-party computation. In *Theory of Cryptography—TCC*, pages 377–396, 2013.

[85] S. Luo, Y. Chen, J. Hu, and Z. Chen. New fully secure hierarchical identity-based encryption with constant size ciphertexts. In *Information Security Practice and Experience—ISPEC*, pages 55–70, 2011.

[86] A. Lysyanskaya and M. Meyerovich. Provably Secure Steganography with Imperfect Sampling. In *Public Key Cryptography—PKC*, pages 123–139, 2006.

[87] W. Mazurczyk, M. Karas, and K. Szczypiorski. Skyde: A skype-based steganographic method. Manuscript, 2013. `arxiv.org/abs/1301.3632`.

[88] Microsoft Corp. Microsoft azure. `http://azure.microsoft.com/en-us/`.

[89] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology—CRYPTO*, pages 41–62, 2001.

[90] R. Ostrovsky. Efficient computation on oblivious rams. In *ACM Symposium on Theory of Computing—STOC*, pages 514–523, 1990.

[91] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology—CRYPTO*, pages 129–140, 1991.

[92] B. Pinkas and T. Reinman. Oblivious ram revisited. In *Advances in Cryptology—CRYPTO*, pages 502–519, 2010.

[93] K. Ren, S. Yu, and W. Lou. Attribute-based on-demand multicast group setup with receiver anonymity. In *Security and Privacy in Communication Networks—SecureComm*, pages 18:1–18:6, 2008.

[94] Y. Ren, S. Wang, and X. Zhang. Anonymous hierarchical identity-based encryption in prime order groups. In *Data and Knowledge Engineering—ICDKE*, pages 230–242, 2012.

[95] L. Reyzin and S. Russell. Simple Stateless Steganography. Cryptology ePrint Archive, Report 2003/093, 2003.

[96] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, Feb. 1978.

[97] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Advances in Cryptology—EUROCRYPT*, pages 457–473, 2005.

[98] J. H. Seo and J. H. Cheon. Fully secure anonymous hierarchical identity-based encryption with constant size ciphertexts. Cryptology ePrint Archive, Report 2011/021, 2011.

[99] J. H. Seo, T. Kobayashi, M. Ohkubo, and K. Suzuki. Anonymous hierarchical identity-based encryption with constant size ciphertexts. In *Public Key Cryptography—PKC*, pages 215–234, 2009.

[100] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology—CRYPTO*, pages 47–53, 1984.

[101] C. E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

[102] E. Shi, T.-H. Chan, E. Stefanov, and M. Li. Oblivious ram with $o((logn)^3)$ worst-case cost. In *Advances in Cryptology—ASIACRYPT*, pages 197–214, 2011.

[103] E. Shi and B. Waters. Delegating capabilities in predicate encryption systems. In *Automata, Languages and Programming—ICALP*, pages 560–578, 2008.

[104] G. Simmons. The Prisoners' Problem and the Subliminal Channel. In *Advances in Cryptology—CRYPTO*, pages 51–67, 1983.

[105] E. Stefanov and E. Shi. Multi-cloud oblivious storage. In *ACM Conference on Computer and Communications Security—CCS*, pages 247–258, 2013.

[106] E. Stefanov and E. Shi. Oblivistore: High performance oblivious cloud storage. In *IEEE Symposium on Security and Privacy—IEEE S&P*, pages 253–267, 2013.

[107] E. Stefanov, E. Shi, and D. X. Song. Towards practical oblivious RAM. In *Network and Distributed System Security Symposium—NDSS*, 2012.

[108] E. Stefanov, M. van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, and S. Devadas. Path oram: An extremely simple oblivious ram protocol. In *ACM Conference on Computer and Communications Security—CCS*, pages 299–310, 2013.

[109] The Economist. Speaking with silence, February 2013.

[110] L. von Ahn and N. J. Hopper. Public-key steganography. In *Advances in Cryptology—EUROCRYPT*, pages 323–341, 2004.

[111] L. von Ahn, N. J. Hopper, and J. Langford. Covert two-party computation. In *ACM Symposium on Theory of Computing—STOC*, pages 513–522, 2005.

[112] B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *Advances in Cryptology—CRYPTO*, pages 619–636, 2009.

[113] M. N. Wegman and J. L. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22(3):265–279, 1981.

[114] P. Williams and R. Sion. Usable pir. In *Network and Distributed System Security Symposium—NDSS*, 2008.

[115] P. Williams and R. Sion. Single round access privacy on outsourced storage. In *ACM Conference on Computer and Communications Security—CCS*, pages 293–304, 2012.

[116] P. Williams, R. Sion, and B. Carbunar. Building castles out of mud: Practical access pattern privacy and correctness on untrusted storage. In *ACM Conference on Computer and Communications Security—CCS*, pages 139–148, 2008.

[117] P. Williams, R. Sion, and A. Tomescu. Privatefs: A parallel oblivious file system. In *ACM Conference on Computer and Communications Security—CCS*, pages 977–988, 2012.

[118] D. Yao, N. Fazio, Y. Dodis, and A. Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *ACM Conference on Computer and Communications Security—CCS*, pages 354–363, 2004.

[119] J. Zhang, W. Zhang, and D. Qiao. S-oram: A segmentation-based oblivious ram. In *ACM Symposium on Information, Computer and Communications Security—ASIACCS*, pages 147–158, 2014.

[120] D. Zuckerman. General weak random sources. In *IEEE Symposium on Foundations of Computer Science—FOCS*, pages 534–543, 1990.