Unified Frameworks for Practical Broadcast Encryption and Public Key Encryption with High Functionalities

by

Nuttapong Attrapadung

Thesis submitted to
Department of Information and Communication Engineering
Graduate School of Information Science and Technology
for the degree of

Doctor of Philosophy

at the

UNIVERSITY of TOKYO

February 2007

Certified by .....
Hideki Imai
Professor
Thesis Supervisor

Certified by .....
Kanta Matsuura
Associate Professor
Thesis Supervisor

Unified Frameworks for Practical Broadcast Encryption and Public Key Encryption with High Functionalities

by

Nuttapong Attrapadung

Thesis submitted to Department of Information and Communication Engineering,
Graduate School of Information Science and Technology for the degree of
Doctor of Philosophy

Abstract

In this thesis, we study encryption schemes with various "high functionalities" including one specific focus on broadcast encryption. As for the main contributions, we propose practical constructions for broadcast encryption schemes and a unified framework for public-key encryption with various functionalities.

The first focus of the thesis is on a special but important kind of encryption schemes, namely broadcast encryption. Such a scheme has many useful applications; the most important one to be mentioned is in the area of the digital right management, where broadcast encryption enables the protection of digital contents such as copyrighted DVD. Such a technology is "inevitable" nowadays as modern advancements in communication infrastructure and digital storage technologies have, on one hand, enabled pervasive digital media distribution, but on the other hand, also allowed the spread of "pirate" contents to be done easier than ever before.

There are some broadcast encryption schemes available in the literature; however, as the number of all users in the system tends to be increased, these existing solutions tend to be quite inefficient, and eventually cannot be used in the real-world application. Our focus is then to construct practical broadcast encryption schemes, which can be "scalable", in the sense that the efficiency of scheme will not be affected by the increasing number of users. As a result of the research, we achieve this goal by constructing the first schemes whose the main two parameters, namely the ciphertext size and the private key size, are independent of the number of all users, while the computational cost is semi-scalable (namely, the cost is increasing but slowly as logarithmically). Behind this scheme, we proposed a theoretical framework that can be used to construct efficient schemes in a systematical way.

The second topic shifts the research focus from the practical point of views to more theoretical ones and looked beyond to more general kinds of what we call *public key encryption with "high functionalities"*. The motivation came from the fact that in recent years, there have been many proposals of cryptographic primitives which extend the normal public-key encryption to achieve useful functionalities such as ID-based encryption, key-insulated encryption, forward-secure encryption, certificate-based encryption, and many more. Each functionality is proved to be useful in different scenarios and applications thereof. Although being seemingly related primitives, there was no unified framework for defining or constructing them.

In this work, we proposed a unified framework called *Directed Acyclic Graph Encryption* (DAGE) that unifies these highly-functional encryption primitives into a unified syntax, a unified security notion, and unified generic/specific constructions. More precisely, we reduce a specification of such a primitive to its necessary and sufficient information, which is turned out to be its "underlying graph": by specifying a graph, the definition and constructions will

be automatically induced by the framework. We also give a primitive implication theorem which gives a simple criterion whether a primitive implies another.

In the theoretical point of view, the merits of the proposed framework are direct. It helps understanding the theoretical essences of public-key encryption schemes with high-functionalities from our unified characterization. This result simplifies the previous complicated researches into one piece. The result on the primitive implication theorem gives an automated verification of relations among primitives. This reduces the proof of relations which has to be performed based on complexity-theoretic approaches in the previous individual researches, which is quite complicated and can be verified only by human, to the logical-based approach, which is much simpler and can be verified automatedly by computer.

The proposed generic construction implies the possibility result for arbitrary graphs. This has merits not only in the theoretical point of view but also in the practical point of view where the protocol designer can just specify a "tailor-made" graph for the on-purposed application and the implementation of the scheme will be prompted to use. Furthermore, any esoteric scheme featured with many combined functionalities can be directly implemented; for example, a forward-secure certificate-based public-key encryption with keyword-searchability. This is also something that previous works cannot achieve, particularly since there was no unified framework to cope with.

For the third main topic, we focus on the combination of the above-mentioned two previous results: public-key broadcast encryption schemes that are *simultaneously* practical and featuring high functionalities. To be able to attain such practical broadcast schemes, it is unavoidable to focus on more specific functionalities (not generic as in the second topic above). We focus on some most useful functionalities, namely forward-security and keyword-searchability. Forward-security enables the private-key updating and guarantees the security of the previously-encrypted ciphertexts even when the present-time private key is exposed. We presents the most practical and scalable forward-secure broadcast encryption so far in the literature. Keyword-searchability enables the search over encrypted data. It has a killer application of encrypted file sharing systems over public database. We presented the first such scheme in the literature.

Thesis Supervisor: Hideki Imai
Title: Professor

Thesis Supervisor: Kanta Matsuura
Title: Associate Professor

Acknowledgments

It has been a pleasure and an honor to work with Prof. Hideki Imai for all these years. To Associate Prof. Kanta Matsuura, I am really grateful when he took over the supervision for me after Imai sensei has retired from Todai. Both have provided insight and has suggested challenging and fruitful research directions to pursue. I will be forever in their debt.

I thank the members and ex-members of Imai lab and Matsuura lab for making my stay at Todai as enjoyable and fulfilling as it has been from my Master period. Also I thank to the members at security group of AIST, or "the new Imai lab" for making every time of my visit enjoyable at most.

I especially thank to Goichiro Hanaoka, the leader of our advance reading group at AIST, who always comes with new challenging ideas. His enthusiasm is positively infectious. Talking to him always restore back my research motivation when I lost it somehow sometimes. I thank to Yang Cui – Sai-kun, the only same-graduate-year friend for all these years ranged from Imai lab to Matsuura lab period. It is always fun to have discussions with him. I thank to Rui Zhang – Cho-san. Debating with him always stimulates new ideas for researches. I also thank him for being the guide when we were in China for Asiacrypt 2006. I thank to Kazukuni Kobara, Miodrag Mihaljevic, Ryo Nojima for fruitful discussions on broadcast encryption as always been from the master period. I am grateful to Hajime Watanabe for supporting me when I applied for the post-doctoral position. Without him, I would be much more worried about jobs and could never concentrate to write the thesis. I am grateful to Kentarou Imafuku, Takayuki Miyadera for many kind advices for research life. I thank to Peng Yang – Yo-kun for helping me in various ways when I was writing this thesis. I thank to Manabu Hagiwara for fun discussions about life and researches as always been. I thank to Takeshi Gomi, Masanori Yoshida, Abdelilah Tabet, Makoto Eguchi, Haruhiro Yoshimoto, and all the graduated master students for making Imai lab lively and be such a warm place to stay. I thank to Takahiro Matsuda, Wataru Kitada, Phan Thi Lan Anh, Vadim Zendejas for making our small basic reading group active and full with energy. I thank to Takashi Kitagawa, Seong Han Shin, Rie Shigetomi, Jin Tamura, Akira Otsuka, and all the people at AIST for making every time of my visit enjoyable at most.

I thank Tomoyuki Asano of Sony for many useful discussions on broadcast encryption and real-world situation on digital content protection industry, especially on Blu-Ray Disc. I thank to Toshihisa Nakano for allowing me to proceed the patents with Matsushita Electric co,ltd.

I also give many thanks to secretaries of the lab from past to present: Miki Watanabe, Yukiko Ito, Naomi Ogasawara, Makiko Nishimura, Kaori Ookubo, Akiko Dobson, Hitomi Hasegawa. Without them, Imai lab would be not really enjoyable as it should be. I especially thank to Yoko Nagahama for moving to Matsuura lab together with us. Without her, I might not do the research smoothly.

This thesis would not have been possible without the financial support from Monbu-kagaku-sho Scholarship of the Japanese Government. I am grateful for that.

I thank to all my friends who support me through all these years. I thank to Burin Anuchitkittikul who always be a good friend. I thank to Juta Pansang, Gabriel Pablo Nava, Weerapong Sattapon, Chiaki Mitsuhashi, Dolrudee Angkapichit, Chakrit Suwannachoke, Hanane Fathi for making my life in Japan enjoyable at most.

I deeply thank to Aiko for her generosity and kind supports to me in all situations.

Most of all, I would like to thank my parents and my brother back in Thailand for continual loves and supports in all stages of my life.

Contents

Chapter 1

Introduction

Encryption is the procedure of rendering a message into a concealed form so that it can be decrypt exclusively by one particular recipient or a group of recipients. From classical to modern cryptography, the aim of encryption has been to enable two parties to exchange messages confidentially, even in the presence of an eavesdropper capable of intercepting the communication. The use of encryption has been confined mostly to diplomatic and military uses in the past, but its scope in nowadays life has broadened enormously in recent years. Thanks to the rise of the Internet for its commercial use in 1990's, nearly every computer sold today is equipped with strong encryption capabilities. Encryption plays an important role in most important industrial communication systems, such as networks used for bank transactions, electronics commerce. It is also used as a tool for larger systems, such as digital right management, electronic cash, electronic voting, electronic auctions, and many more.

In this thesis, we consider encryption schemes with various "high functionalities". Each functionality is proved to be useful in different scenarios and applications thereof. Towards the study, our two general extreme goals are to formalize a unified framework and to construct practical schemes. In this chapter, we describe some background, research motivations and our contributions in brief.

1.1 Background

We first describe some background on cryptography in general and encryption in particular. We organize the outline of survey in the historical aspect as done in many textbooks in cryptography such as [Gol99], as it appears to be a natural way to give broad overview.

Classical Cryptography. Classical cryptography was confined to the art of designing and breaking encryption schemes or "codes". Shannon [Sha49] answered the fundamental question of classical cryptography that it is "impossible" to design a code that cannot be broken. Instead, the challenge was then to construct codes which is "infeasible" or hard to break. Until the late 1970's, following the challenge, some *symmetric-key encryption* schemes, such as the Data Encryption Standard (DES), has been designed. Such a symmetric-key scheme allows anyone who has enough encryption to encrypt also to be able to decrypt messages. Therefore, any two users who wants to communicate securely must have exchanged keys beforehand in a secure way.

The Beginning of Modern Cryptography (1970's). Motivated by the problem of secret key distribution problem for symmetric-key encryption, Diffie and Hellman, in their revolutionary paper [DH76], invented an entirely new type of cryptography called *public key*, together with new concepts such as *digital signatures* and *one-way functions*. Speaking informally, an injective function $f : A \to B$ is "one-way" if it is easy to compute $f(x)$ for any $x \in X$ but hard to compute $f^{-1}(y)$ for most randomly selected y in the range of x. *Public-key encryption* (PKE) can be informally viewed as a kind of one-way function: it is a function which maps plaintext message to ciphertext that can be computed by anyone possessing the *public key* but whose inverse function cannot be computed in a reasonable amount of time without some additional information called the *private key*. Therefore, anyone can send a message by using only the public key of the intended recipient, who disseminate her public key as she likes.

The first implementations of public key encryption are due to Merkle and Hellman [MH78], Rivest, Shamir, and Adleman [RSA78], and Rabin [Rab79]. The most popular scheme is that of [RSA78], known as RSA encryption, has since then been used in practical for many systems and still.

Rigorous Treatments (1980's). It is subtle issue to evaluate the security of encryption schemes. It is typically insufficient to ensure only that the adversary is unlikely to be able to compute the inverse of encryption function. Goldwasser and Micali, in their seminal paper [GM84], introduced the first rigorous treatments of stronger notions for secure encryption. The notion, called *semantic security*, captures the inability of adversary to determine any information whatsoever about the message from the ciphertexts. Their work did not only supply the robust notions for encryption, but also introduced paradigms, such as simulation paradigms and computational indistinguishability, which played a major part for subsequent researches, including zero-knowledge proof [GMR85], secure multi-party computation [GMW87], and many more.

Naor and Yung [NY90], Rackoff and Simon [RS91], further considered stronger notions for public key encryption, called semantic security against non-adaptive and adaptive chosen-ciphertext attack (CCA1 and CCA2). The intuition for such notions is that even if an adversary is able to obtain the decryption of any ciphertexts of his choice, he still gets no information whatsoever about other encrypted messages. Dolev, Dwork, and Naor [DDN91] considered the notion called non-malleability. It was later proved [BDPR98] that both semantic security and non-malleability are equivalent when adaptive chosen-ciphertext attack is considered. In [NY90, DDN91], CCA1-secure and CCA2-secure generic constructions based on non-interactive zero knowledge proofs were presented.

Practical Constructions (1990's). Generic constructions of [NY90, DDN91] are not so efficient; they only showed the *plausibility result* of constructing schemes in the strongest notion in generic ways. The first practical CCA-secure scheme was proposed by Bellare and Rogaway [BR94], called Optimal Asymmetric Encryption Padding (OAEP). (Note that the correct revised CCA2-security proof for the case when applying to RSA was done by [FOPS01]). The security proof was done in the *random oracle model* [BR93], of which hash functions are idealized in the proof, but are replaced by some real-world hash functions when the scheme is used in practice. Conversions from weak PKE to CCA-secure one were proposed in [FO99], also with the security proof in the random oracle model. However, [CGH98] showed a negative result regarding random oracle model which states

that there is an encryption scheme which is secure in the random oracle model but admits no secure real-world instantiations. The question was then left to construct CCA-secure in the standard model, *i.e.,* without random oracle. Cramer and Shoup [CS98] proposed the first such scheme, based on ElGamal encryption [ElG85]. Their results were also generalized in [CS02]. The framework of hybrid encryption called key-encapsulation/data-encapsulation mechanisms (KEM/DEM), which is useful particularly for constructing secure PKE with long messages, was proposed in [Sho01, CS03].

In this period, rigorous definitions for security of encryptions also have been further refined. The work by [BDPR98] proved some relationships among various notions for the case of public key encryption, while [BDJR97] concerned the case of symmetric key encryption.

Encryption Schemes with Advance Features (2000's). In the present decade, we still have seen some refinement on the definitional works for security notions of encryption (and beyond): one of the most sophisticated notion is the universal composability [Can01]. Other examples are relaxed chosen ciphertext security [CKN03], chosen ciphertext security for multiple encryption [DK05], adaptively-secure non-interactive encryption [CHK05]. On the other hand, new public key encryption schemes with efficiency improvements were also proposed, such as [KD04]. A framework called Tag-KEM/DEM [AGKS05] extends the previous hybrid encryption framework and can unify and/or improve many existing PKE constructions. Regarding symmetric key encryption, Rijndael was chosen as the new Advance Encryption Standard (AES) in 2001, and is expected to be used worldwide and analyzed extensively, as was the case with its predecessor, DES.

Although many important researches as described above have been conducted, one of the most distinguished streamline of research in this 2000's period turned out to be the area of encryption schemes that feature useful functionalities. Although some of them can be already noticed from earlier time, their crucial revolutions are in the present era.

We mention firstly to *broadcast encryption*, which is an encryption scheme that allows a broadcaster to deliver an encrypted data so that only a dynamically changing designated group of parties can decrypt it. Broadcast encryption plays an important role for digital right management, as we will describe later. Although broadcast encryption was first introduced by Fiat and Naor earlier [FN93], the first efficient collusion-resistant construction was presented only recently by Naor, Naor, and Lotspiech [NNL01], of which their paradigm has then been the key-stone to many subsequent works in this area.

The most breakthrough in this present era was the invention by Boneh and Franklin [BF01] of the first fully-functional *identity-based encryption* (IBE), which is the concept introduced by Shamir [Sha84] a decade ago. IBE provides a public key encryption mechanism where an arbitrary string, such as recipient's identity, can be served as a public key. Such a property is useful, *e.g.,* for simplifying public key infrastructure. Following the invention of IBE, many new kinds of public key encryption were proposed, for instance, *key-insulated encryption* [DK02], *forward-secure encryption* [CHK03], *certificate-based encryption* [Gen03], *intrusion-resilient encryption* [DFK+03], *public-key encryption with keyword search* [BDOP04], *hierarchical IBE* [HL02, GS02], *time-capsuled encryption* [MHS03], *attribute-based encryption* [SW05, GPSW06] and many more including some of their generalizations themselves. Security notions of each encryption primitives were defined based on the basic paradigm of [GM84] with adaptations to each scenario. In this thesis, we coin the term *public key encryption with high functionalities* to call these schemes.

We also note that IBE is not only already interesting for its own right, but its weak version called selective-ID IBE can also be converted to CCA-secure PKE efficiently in the standard model [CHK04, BK05, BMW05].

1.2 Motivations and Results Overview

In this thesis, we study the class of those encryption schemes with advance functionalities, which is one of the main streams of recent researches in cryptography. We concern mostly the following goals in general.

- The first goal is "practicality": we aim at constructing efficient schemes for *broadcast encryption*, which is one of few encryption primitives featuring advance functionalities that is already used in practice, such as in digital right management.
- The second goal is "unified formalization": we aim at introducing a new unified framework for the class of *public key encryption with high functionalities* to capture their theoretical essence and to have a convenient way to derive the security definition and some secure constructions for each existing or newly constructed encryption primitive from now automatically from the unified ones.

We now describe our motivations and results in more detail separated by each chapter in the thesis. In Chapter 2 we will give some basic notations and some preliminary tools to be used throughout the thesis. Main contributions of thesis are in Chapter 3,4,5.

1.2.1 A Study on Broadcast Encryption

In Chapter 3, we focus on *symmetric key broadcast encryption*. Such a scheme allows one party, called a broadcaster, to deliver an encrypted data so that only a dynamically changing designated group of parties can decrypt it.

Broadcast encryption [FN93] is motivated largely by many applications that have the nature of "conditional access". Among others, it can be best understood in the scenario of pay-TV systems where broadcast encryption can be applied to restrict access to a TV cable system's premium channels to viewers who have paid a subscription fee. Ironically, this flagship application for broadcast encryption has proven less important than another application - *media content protection*. This application has the same nature of conditional access to an encrypted broadcast as can be described in the following scenario. At first, a movie company makes an encrypted recording, such as an encrypted movie on DVD. Years later, on one hand, a new legitimate player that might not even have existed when the recording was made needs to play it back. On the other hand, we consider a player that was legitimate but by now its private key inside has been compromised and is abused in a piracy business (such as cloning a pirate decoder from the compromised one); therefore, it is important that the ability of the compromised key to decrypt must be *revoked* so that combatting the piracy is done in the sense that we render the pirate decoder cloned from them useless. In both cases, broadcast encryption can be utilized by defining a proper group of legitimate users at the present time, thanks to the way that it can work even if the group is dynamically changing.

Broadcast encryption is thus one of the central tools to the digital right management issues in the sense that it efficiently "establishes the right" of newly joined legitimate enti-

ties and "revokes the right" of identified malicious or compromised entities to decrypt the encrypted data. Media content protection essentially requires these properties.

Motivations and Our Contributions. There are some broadcast encryption schemes available in the literature (cf. Section 3.1); however, as the number of all users in the system tends to be increased, these existing solutions tend to be quite inefficient, and eventually cannot be used in the real-world application. Our focus is then to construct practical broadcast encryption schemes, which can be "scalable", in the sense that the efficiency of scheme will not be affected by the increasing number of users. As a result of the research, we achieve this goal by constructing the first schemes whose the main two parameters, namely the ciphertext size and the private key size, are independent of the number of all users, while the computational cost is semi-scalable (namely, the cost is increasing but slowly as logarithmically). Behind this scheme, we proposed a theoretical framework that can be used to construct secure and efficient schemes in a systematical way.

In particular, our proposed constructions are more efficient than the so-called Subset-Difference broadcast encryption method (proposed by Naor, Naor, and Lotspeich in 2001), which is the scheme that was recently chosen as a new standard called Advance Access Content System (AACS) and will be used in the next-generation DVD materials such as Blu-ray Disc (BD) and HD-DVD.

1.2.2 A Study on Public-Key Encryption with High Functionalities

In Chapter 4, we focus on *public key encryption with high functionalities*. Such schemes are extensions of normal public key encryption so as to strengthen the security or to achieve some useful functionalities which are specific to applications thereof. Indeed they solve some problems which basic public-key encryption cannot. Many kinds of existing schemes were proposed. For instance, encryption primitives which enable significant simplification of public key infrastructure are identity-based Encryption (IBE) [BF01], certificate-based encryption [Gen03], and certificateless PKE [AP03]. Examples of encryption primitives that cope with the vulnerability against secret key exposure are forward-secure encryption [CHK03], key-insulated encryption [DK02], intrusion-resilient encryption [DFK+03]. Other primitives are those that achieves new additional functionalities, such as public-key encryption with keyword search [BDOP04], time-capsuled PKE [MHS03]. Moreover, hierarchical versions such as hierarchical IBE (HIBE) [HL02, GS02] or some combinations such as forward-secure HIBE [YFDL04] were also reported in the literature.

Motivations and Our Contributions. The algorithm syntax and security definitions for each of existing schemes for public key encryption with high functionalities were formalized separately in each contributing paper anew every time when such a primitive was introduced. Such definitions were formalized by extending the standard notions for normal PKE, such as semantic security and security against chosen-ciphertext attack, with appropriate adaptations to the corresponding scenario. Doing such formalization each time anew is an inconvenient approach. Much worse, careless in doing so may yield only weak, incorrect, or unsubstantiated notions of security. To this end, we raise a question whether a "unified" approach for such formalizations exists. One support evidence for its existence is that all the security notions defined in those existing primitives can be traced back to the paradigm of [GM84, NY90, RS91, DDN91] with only some adaptations. Our primary motivation is thus to obtain such a unified approach.

In this thesis, we answer the motivated question in affirmative. We propose a unified framework called *Directed Acyclic Graph Encryption* (DAGE) that unifies these highly-functional encryption primitives into a unified syntax and a unified security notion with many levels of security. More precisely, we reduce a specification of such a primitive to its necessary and sufficient information, which is turned out to be its underlying graph: by specifying a graph, the definition of syntax and security notion will be automatically induced by the framework.

In the theoretical point of view, the merits of the proposed framework are direct. It helps understanding the theoretical essences of the encryption schemes with high-functionalities from our unified characterization. This result simplifies the previous complicated researches into one piece.

Our second contribution is the *primitive implication theorem*, which is central to our paradigm. It states a simple logical criterion where if a pair of encryption primitives casted as DAGE satisfies, then the existence of secure construction from one primitive implies that of the other. As a result, this reduces the proof of relations which has to be performed based on *complexity-theoretic* approaches in the previous individual researches, which is quite complicated and can be verified only by human, to the *logical-based* approach, which is simpler and can be verified automatedly by computer.

Our third contributions are generic constructions for any primitives that can be casted as DAGE. This implies the possibility result for arbitrary graphs. This has merits not only in the theoretical point of view but also in the practical point of view where the protocol designer can just specify a "tailor-made" graph for the on-purposed application and the implementation of the scheme will be prompted to use. Furthermore, any esoteric scheme featured with many combined functionalities can be directly implemented; for example, a forward-secure certificate-based public-key encryption with keyword-searchability. This is also something that previous works cannot achieve, particularly since there was no unified framework to cope with.

1.2.3 Combining Best of Both Studies

In Chapter 5, we focus on the combination of the above-mentioned two previous results: public key broadcast encryption schemes that are *simultaneously* practical and featuring high functionalities. Although, broadcast encryption is already useful in the symmetric key setting as was focused in Chapter 3, we search for schemes in the public key setting in this chapter since they give more flexibility. To be able to attain such practical broadcast schemes, it is unavoidable to focus on more specific functionalities (not generic as in the second topic above). We focused on some most useful functionalities, namely forward security and keyword-based searchability.

The first contribution is related to *forward security*. This functionality enables the private-key updating and guarantees the security of the previously-encrypted ciphertexts even when the present-time private key is exposed. We presents the most practical and scalable forward-secure broadcast encryption so far in the literature.

Another contribution is on *keyword-based searchability*. This functionality enables the search over encrypted data. It has a killer application of encrypted file sharing systems over public database. In such an application, users share encrypted files among arbitrarily specified privileged users. Due to a possible large amount of databases, a user Alice might want to retrieve only those files that contain a particular keyword of interest (among all

the files in which Alice is specified as a privileged user), but without giving the server the ability to decrypt the databases. Our keyword-searchable broadcast encryption allows to do exactly this. We presented the first such scheme in the literature.

Chapter 2

Preliminaries

2.1 Basic Notations and Terminology

We adopt some of the now-standard notations and definitions of Goldwasser, Micali, and Rackoff [GMR89] and add some new ones to be used throughout the thesis.

The set of λ-bit strings is denoted by $\{0,1\}^\lambda$. The λ-bit string consisted of only a is denoted by a^λ or $a^{(\lambda)}$. Concatenation of two string x_1, x_2 is denoted by $x_1 \| x_2$ or $x_1 \circ x_2$.

For a finite set S we use $y \leftarrow S$ or $S \rightarrow y$ to define a random variable y that picks an element of S uniformly at random. Let $A(\cdot)$ be an algorithm. By $y \leftarrow A(x)$ or $A(x) \rightarrow y$ we denote that y was obtained by running A on input x. If A is deterministic, then y is unique; if A is randomized, then y is a random variable.

Let b be a boolean function. By $(y \leftarrow A(x) : b(y))$ we denote the event that $b(y)$ is true after y was generated by running A on input x. We let

$$\Pr[y_1 \leftarrow A(x_1);\ \ldots\ ; y_t \leftarrow A(x_t) : b(y_1, \ldots, y_t)]$$

denote the probability of the event that $b(y_1, \ldots, y_t)$ is `true` after the value $y_1, \ldots, y_t$ was obtained by (orderly) running algorithm $A_1, \ldots, A_t$ on inputs $x_1, \ldots, x_t$. See, for example, Definition 2.8. We often denote by state an auxiliary output of an algorithm which describes the state information of that algorithm. When such conditional events are quite long to capture in one line, for visual ease we define and write it as an individual experimental event (see, for example, Definition 2.5); if such conditional events are even in more detail, we write it in text form (see, for example, Section 4.2.2).

We let $A^{\mathcal{O}}$, where $\mathcal{O} = \{O_1(\cdot), O_2(\cdot), \ldots, O_w(\cdot)\}$, denote the algorithm A that has access to oracle machines $O_1, \ldots, O_w$ which will answer queries to it. We let "Left-or-Right oracle" $\mathsf{LR}(x_0, x_1, b) = x_b$, and "Real-or-Random oracle" $\mathsf{RR}(x, b)$ output x if $b = 0$ or output a random string with the same length as x if $b = 1$. These last two functions will be used in the formulation of encryption schemes. [BDJR97]

A probabilistic, polynomial-time interactive Turing machine (PPT ITM) $\boldsymbol{M}$ is one for which there exists a polynomial $p(\cdot)$ such that for all inputs $x_1, \ldots, x_t$, all randomness r, and arbitrary behavior of other machines with which $\boldsymbol{M}$ is interacting, $\boldsymbol{M}(x_1, \ldots, x_t)$ with randomness r runs in time bounded by $p(|x_1 \circ \cdots \circ x_t|)$. For the detail about ITM and PPT algorithm, see [Gol01].

We say that a function $\mu : \mathbb{Z}^+ \rightarrow \mathbb{R}^+$ is negligible if for all $d > 0$ and sufficiently large k we have $0 < \mu(k) < 1/k^d$.

We denote the set $\{1, 2, ..., a\}$ by $[a]$. For any set A, we denote by 2^A the set $\{S \subseteq A\}$; and by $\binom{A}{t}$ the set $\{S \subseteq A : |S| = t\}$.

More notations related to graphs or sequences are postponed to Section 2.5.

2.2 Integer Factoring Related Assumptions

We review some well-known intractability assumptions related to integer factoring as follows.

Hardness of Factoring. This assumption states that it is infeasible for any algorithm to find the factors of a random product of two large primes. More formally, let $\mathsf{2\text{-}FACTOR}_\lambda := \{pq : p, q \in \mathtt{primes}; |p| = |q| = \lambda\}$. Then, the factoring assumption states that for every probabilistic polynomial-time algorithm $\boldsymbol{A}$, the following probability is negligible (in λ):

$$\Pr[p \cdot q = n : n \leftarrow \mathsf{2\text{-}FACTOR}_\lambda; (p, q) \leftarrow \boldsymbol{A}(n)]$$

The RSA Assumption. Informally, for a modulus $n = pq$ which is the product of two primes, a fixed e which is relatively prime to $\phi(n)$, and a random $z \in \mathbb{Z}_n^*$, the RSA assumption [RSA78] states that it is infeasible to compute $z^{1/e} \bmod n$. More formally, the RSA assumption states that for all probabilistic polynomial-time algorithms $\boldsymbol{A}$, the following probability is negligible (in λ):

$$\Pr[x^e = z \bmod n : n \leftarrow \mathsf{2\text{-}FACTOR}_\lambda; z \leftarrow \mathbb{Z}_n^*; x \leftarrow \boldsymbol{A}(n, e, z)],$$

where e is any number relatively prime to $\phi(n)$. If the factorization of n is known, then for all e relatively prime to $\phi(n)$ and for all $z \in \mathbb{Z}_n^*$, the value $z^{1/e}$ can be computed efficiently. Thus the RSA assumption is at least as strong as the assumption that factoring is hard.

The Strong RSA Assumption. Informally, the strong RSA assumption [BP97] is similar to the RSA assumption except that the problem does not fix the exponent e, i.e., it states that for a random $z \in \mathbb{Z}_n^*$, it is infeasible to find a pair (x, e) where $x \in \mathbb{Z}_n^*$ and $e \in \mathbb{Z}_{>1}$ such that $x^e = z \bmod n$. More formally the strong RSA assumption states that for all probabilistic polynomial-time algorithms $\boldsymbol{A}$, the following probability is negligible (in λ):

$$\Pr[x^e = z \bmod n : n \leftarrow \mathsf{2\text{-}FACTOR}_\lambda; z \leftarrow \mathbb{Z}_n^*; (x, e) \leftarrow \boldsymbol{A}(n, z)].$$

The strong RSA assumption is at least as strong as the RSA assumption.

Also note that if n is a safe RSA modulus (i.e.,$n = pq$ with $p = 2p' + 1, q = 2q' + 1$, and p, q, p', q' are all primes), it is a good habit to restrict operation to the subgroup of quadratic residues modulo n, i.e., the cyclic group QR_n generated by an element of order $p'q'$. This is because the order $p'q'$ of QR_n has no small factors.

2.3 Bilinear Maps and Related Assumptions

Bilinear Maps. We use the standard terminology from [Jou00, BF01]. Let $\mathbb{G}, \mathbb{G}_1$ be two groups of prime order p, written multiplicatively. Let g be a generator of $\mathbb{G}$. A bilinear map is a map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$ for which the following hold:

1. The map e is bilinear, that is, for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, we have

$$e(u^a, v^b) = e(u, v)^{ab}.$$

2. The map e is non-degenerate: $e(g, g) \neq 1$.

We say that $\mathbb{G}$ is a bilinear group if the group action in $\mathbb{G}$ can be computed efficiently and there exists $\mathbb{G}_1$ for which the bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is efficiently computable.

For typical implementations, $\mathbb{G}$ is a subgroup of the group of points on an elliptic curve over a finite field and $\mathbb{G}_1$ is a subgroup of the multiplicative group of a related finite field. We simply use bilinear maps in a black-box manner throughout the thesis. See, for example, [BSS05] for detailed implementations of bilinear maps and selection of curves with suitable properties.

We now state intractability assumptions related to bilinear maps that we will use in this thesis.

Decision BDH Assumption. Let $\mathbb{G}$ be a bilinear group of prime order p. The decision n-BDH (Bilinear Diffie-Hellman) problem [BF01] in $\mathbb{G}$ is stated as follows: given a vector

$$\left(g, g^a, g^b, g^c, Z\right) \in \mathbb{G}^4 \times \mathbb{G}_1$$

as input, determine whether $Z = e(g, g)^{abc}$. An algorithm $\mathcal{A}$ that outputs $\beta \in \{0, 1\}$ has advantage ϵ in solving decision n-BDH in $\mathbb{G}$ if

$$|\Pr\left[\mathcal{A}\left(g, g^a, g^b, g^c, e(g, g)^{abc}\right) = 0\right] - \Pr\left[\mathcal{A}\left(g, g^a, g^b, g^c, Z\right) = 0\right]| \geq \epsilon,$$

where the probability is over the random choice of generators $g \in \mathbb{G}$, the random choice of $a, b, c \in \mathbb{Z}_p$, the random choice of $Z \in \mathbb{G}_1$, and the randomness of $\mathcal{A}$. We refer to the distribution on the left as $\mathcal{P}_{BDH}$ and the distribution on the right as $\mathcal{R}_{BDH}$. We say that the decision (t, ϵ, n)-BDH assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ in solving the decision n-BDH problem in $\mathbb{G}$. We sometimes drop the t and ϵ and refer it as the decision n-BDH assumption in $\mathbb{G}$.

Decision BDHE Assumption. Let $\mathbb{G}$ be a bilinear group of prime order p. The decision n-BDHE (Bilinear Diffie-Hellman Exponent) problem [BGW05, BBG05] in $\mathbb{G}$ is stated as follows: given a vector

$$\left(g, h, g^{\alpha}, g^{(\alpha^2)}, \ldots, g^{(\alpha^n)}, g^{(\alpha^{n+2})}, \ldots, g^{(\alpha^{2n})}, Z\right) \in \mathbb{G}^{2n+1} \times \mathbb{G}_1$$

as input, determine whether $Z = e(g, h)^{(\alpha^{n+1})}$. We denote $g_i = g^{(\alpha^i)} \in \mathbb{G}$ for shorthand. Let $\vec{y}_{g,\alpha,n} = (g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$. An algorithm $\mathcal{A}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving decision n-BDHE in $\mathbb{G}$ if

$$|\Pr\left[\mathcal{A}\left(g, h, \vec{y}_{g,\alpha,n}, e(g_{n+1}, h)\right) = 0\right] - \Pr\left[\mathcal{A}\left(g, h, \vec{y}_{g,\alpha,n}, Z\right) = 0\right]| \geq \epsilon,$$

where the probability is over the random choice of generators $g, h \in \mathbb{G}$, the random choice of $\alpha \in \mathbb{Z}_p$, the random choice of $Z \in \mathbb{G}_1$, and the randomness of $\mathcal{A}$. We refer to the

distribution on the left as $\mathcal{P}_{BDHE}$ and the distribution on the right as $\mathcal{R}_{BDHE}$. We say that the decision (t, ϵ, n)-BDHE assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ in solving the decision n-BDHE problem in $\mathbb{G}$. We sometimes drop the t and ϵ and refer it as the decision n-BDHE assumption in $\mathbb{G}$.

It is shown in [BBG05] that this assumption holds in the generic bilinear group model [Sho97] with the computational lower bound of $\Omega(\sqrt{p/n})$ on the difficulty of breaking (cf.[BBG05]). Cheon [Che06] recently showed a concrete attack with roughly the same complexity. It is recommended to either increase p to compensate the security loss appropriately or use p of a special form where $p-1$ and $p+1$ have no small divisor greater than $\log^2 p$ to avoid the attack.

Decision Linear Assumption. Let $\mathbb{G}$ be a bilinear group of prime order p. The decision Linear problem [BBS04] in $\mathbb{G}$ is stated as follows: given a vector

$$(g, g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_2 z_4}, Z) \in \mathbb{G}^6$$

as input, determine whether $Z = g^{z_3+z_4}$. The decision (t, ϵ)-Linear assumption posits the hardness of this problem and can be formally defined in an analogous way as above.

2.4 Cryptographic Tools

2.4.1 One-way Function and Pseudo-random Sequence Generator.

Definition 2.1 (OWF)**.** A polynomial time computable function $f : \{0,1\}^* \to \{0,1\}^*$ is called a one-way function if for any PPT adversary $\boldsymbol{A}$, the following probability is negligible (in λ):

$$\Pr[f(z) = y : x \leftarrow \{0,1\}^*; y \leftarrow f(x); z \leftarrow \boldsymbol{A}(1^\lambda, y)].$$

In other words, f is easy to compute but hard to invert on a random input x.

Definition 2.2 (PRSG)**.** A deterministic polynomial time computable function $G : \{0,1\}^k \to \{0,1\}^{l(k)}$ is called a pseudo random sequence generator (PRSG) stretching from k to $l(k)$ bits where $l(k) > k$ if for any PPT adversary $\boldsymbol{A}$, the following is negligible (in $l(k)$):

$$|\Pr[\boldsymbol{A}(t) = 1 : x \leftarrow \{0,1\}^k; t \leftarrow G(x)] - \Pr[\boldsymbol{A}(t) = 1 : t \leftarrow \{0,1\}^{l(k)}]|.$$

In other words, $G(r)$ for a random $r \in \{0,1\}^k$ (this r is called a seed of G) is indistinguishable to any PPT algorithm from a truly random $R \in \{0,1\}^{l(k)}$. The following important result was proved by [ILL89].

Theorem 2.3. OWF*'s exist* $\iff$ PRSG*'s stretching to* $k+1$ *bits exist* $\iff$ PRSG*'s stretching to* $l(k)$ *bits exist for any polynomial* $l(k) > k$.

One of the consequences is that we can talk about PRSG's in general without worrying about the particular stretching factor.

2.4.2 Public-key and Symmetric-key Encryption Schemes.

Encryption is the main theme of this thesis, where we consider encryption with various functionalities. Formalizations of basic public-key and symmetric-key encryption schemes are thus crucial foundations to our theme. We elaborate such formalizations in this section.

The widely accepted notion of security for encryption is semantic security, introduced by Goldwasser and Micali [GM84]; this definition states (informally) that anything which can be efficiently computed about a plaintext message when given access to the encryption of that message can be efficiently computed without access to the encryption of the message (in particular, this implies that the message itself cannot be determined without the decryption key). A second definition of security is that of indistinguishability [GM84]; here, an adversary outputs two messages x_0, x_1 and is then given an encryption of one of them (chosen at random). The adversary succeeds if he can determine which message was encrypted. An encryption scheme is indistinguishable if the success probability of any PPT adversary is negligibly close to 1/2 (the adversary can always succeed half the time by guessing randomly). The basic definition of indistinguishability given above is equivalent to that of semantic security [GM84]. Under the basic definition, however, the adversary is given only the ciphertext. Subsequent work has considered stronger attacks in the public-key [NY90, RS91, BDPR98] and symmetric-key [BDJR97] settings. The strongest attack is the so-called chosen-ciphertext attack. It is relaxed to a weaker but considered useful notion in [ADR02, CKN03]. The syntax of both public-key and private-key encryption scheme can be unified as follows.

Definition 2.4 (Encryption Scheme)**.** An encryption scheme E is a 3-tuple of polynomial-time algorithms $(\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt})$:

$\mathsf{Setup}(1^\lambda)$**:** Takes as input a security parameter 1^λ. It outputs keys sk and pk.

$\mathsf{Enc}(\mathsf{pk}, M)$**:** Takes as input the key pk, and a message M. It outputs a ciphertext C.

$\mathsf{Dec}(\mathsf{sk}, C)$**:** Takes as input the key sk, and a ciphertext C. It returns M or $\perp$ by the following consistency: $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{Encrypt}(\mathsf{pk}, M)) = M$

A private-key (or symmetric-key) encryption scheme SE is defined as one which for all $(\mathsf{pk}, \mathsf{sk})$ output by Setup, we have $\mathsf{pk} = \mathsf{sk}$. A public-key encryption scheme PE will have $\mathsf{pk} \neq \mathsf{sk}$; note that this is not implied by the definition above, yet will be implied by the definition of security below.

Definition 2.5 ($\mathsf{IND\text{-}ATK}$ Security of Symmetric-key Encryption)**.** A symmetric-key encryption scheme SE is secure in the sense of $\mathsf{IND\text{-}ATK}$ where $\mathsf{ATK} \in \{\mathsf{COA}, \mathsf{CPA}, \mathsf{CCA1}, \mathsf{CCA2}\}$ (Indistinguishability against Ciphertext-Only, Chosen-Plaintext and Non-adaptive/ Adaptive Chosen-Ciphertext Attack resp.) if for all polynomial time adversaries $\boldsymbol{S} = (\boldsymbol{S}_{\mathsf{find}}, \boldsymbol{S}_{\mathsf{guess}})$, its advantage $|\Pr[\mathbf{Exp}^{\mathsf{IND-ATK}}(\mathsf{SE}, \boldsymbol{S}) = 1] - \frac{1}{2}|$ is a negligible function in λ where we define

$$
\begin{array}{rcl}
\text{Experiment } \mathbf{Exp}^{\mathsf{IND-ATK}}(\mathsf{SE}, \boldsymbol{S}) & & \\
\mathsf{Setup}(1^\lambda) & \rightarrow & \mathsf{sk}; \\
(M^*, \mathsf{state}) & \leftarrow & \boldsymbol{S}_{\mathsf{find}}^{\mathcal{O}_1}(1^\lambda) \\
\{0,1\} & \rightarrow & b; \\
\mathsf{RR}(M^*, b) & \rightarrow & M^\dagger; \\
\mathsf{Enc}(\mathsf{sk}, M^\dagger) & \rightarrow & C^*; \\
b' & \leftarrow & \boldsymbol{S}_{\mathsf{guess}}^{\mathcal{O}_1}(C^*, \mathsf{state}) \\
\text{return 1 iff } b = b' & &
\end{array}
$$

where

- If $\mathsf{ATK} = \mathsf{COA}$, then $\mathcal{O}_1 = \mathcal{O}_2 = \emptyset$,

- If $\mathsf{ATK} = \mathsf{CPA}$, then $\mathcal{O}_1 = \mathcal{O}_2 = \{\mathsf{Enc}(\mathsf{sk}, \cdot)\}$,
- If $\mathsf{ATK} = \mathsf{CCA1}$, then $\mathcal{O}_1 = \{\mathsf{Enc}(\mathsf{sk}, \cdot), \mathsf{Dec}(\mathsf{sk}, \cdot)\}$, $\mathcal{O}_2 = \{\mathsf{Enc}(\mathsf{sk}, \cdot)\}$,
- If $\mathsf{ATK} = \mathsf{CCA2}$, then $\mathcal{O}_1 = \mathcal{O}_2 = \{\mathsf{Enc}(\mathsf{sk}, \cdot), \mathsf{Dec}(\mathsf{sk}, \cdot)\}$ with the restriction that $\boldsymbol{S}_{\mathsf{guess}}$ cannot ask the decryption of C^*.

It is known [DDN00] that PRSG is necessary and sufficient to construct symmetric-key encryption in the strongest notion (IND-CCA2). From the previous section, we thus have that one-way function is also necessary and sufficient.

Definition 2.6 (IND-ATK Security of Public-key Encryption)**.** A public-key encryption scheme PE is secure in the sense of IND-ATK where $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA1}, \mathsf{RCCA2}, \mathsf{gCCA2}, \mathsf{CCA2}\}$ (Indistinguishability against Non-adaptive / Relaxed Adaptive / Generalized Adaptive / Adaptive Chosen-Ciphertext Attack resp.) if for all polynomial time adversaries $\boldsymbol{P} = (\boldsymbol{P}_{\mathsf{find}}, \boldsymbol{P}_{\mathsf{guess}})$, its advantage $|\Pr[\mathbf{Exp}^{\mathsf{IND-ATK}}(\mathsf{PE}, \boldsymbol{P}) = 1] - \frac{1}{2}|$ is a negligible function in λ where we define

$$
\begin{array}{rcl}
\text{Experiment } \mathbf{Exp}^{\mathsf{IND-ATK}}(\mathsf{PE}, \boldsymbol{P}) & & \\
\mathsf{Setup}(1^\lambda) & \rightarrow & (\mathsf{pk}, \mathsf{sk}); \\
(M_0, M_1, \mathsf{state}) & \leftarrow & \boldsymbol{P}_{\mathsf{find}}^{\mathcal{O}_1}(1^\lambda, \mathsf{pk}) \\
\{0,1\} & \rightarrow & b; \\
\mathsf{LR}(M_0, M_1, b) & \rightarrow & M^\dagger; \\
\mathsf{Enc}(\mathsf{sk}, M^\dagger) & \rightarrow & C^*; \\
b' & \leftarrow & \boldsymbol{P}_{\mathsf{guess}}^{\mathcal{O}_1}(C^*, \mathsf{state}) \\
\text{return 1 iff } b = b' & &
\end{array}
$$

where if $\mathsf{ATK} = \mathsf{CPA}$, then $\mathcal{O}_1 = \mathcal{O}_2 = \emptyset$; if $\mathsf{ATK} = \mathsf{CCA1}$, then $\mathcal{O}_1 = \mathsf{Dec}(\mathsf{sk}, \cdot)$, $\mathcal{O}_2 = \emptyset$; otherwise $\mathcal{O}_2$ is also $\mathsf{Dec}(\mathsf{sk}, \cdot)$ with some restrictions imposed:

- if $\mathsf{ATK} = \mathsf{RCCA2}$, $\boldsymbol{P}_{\mathsf{guess}}$ does not allow to ask C such that $\mathsf{Dec}(\mathsf{sk}, C) \in \{M_0, M_1\}$,
- if $\mathsf{ATK} = \mathsf{gCCA2}$, $\boldsymbol{P}_{\mathsf{guess}}$ does not allow to ask C such that $R(C, C^*) = 1$ for some efficiently computable relation R,
- if $\mathsf{ATK} = \mathsf{CCA2}$, $\boldsymbol{P}_{\mathsf{guess}}$ does not allow to ask C^*.

2.4.3 Signature Schemes

In a digital signature scheme, a signer publishes a public verification key and keeps secret a signing key; a signer then uses a signing algorithm to associate a signature to a message; this signature can be validated by anyone who knows the verification key.

Definition 2.7 (Digital Signature Scheme)**.** A signature scheme SIG is a triple of algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ such that for some polynomial $p(\cdot)$:

$\mathsf{Gen}(1^\lambda)$ Takes as input a security parameter 1^λ. It outputs verification key V_{SIG} and signing key K_{SIG}.

$\mathsf{Sign}(K_{\mathsf{SIG}}, M)$ Takes as input a signing key K_{SIG}, and a message $M \in \{0,1\}^{\leq p(\lambda)}$. It outputs a signature M.

$\mathsf{Vrfy}(V_{\mathsf{SIG}}, M, \sigma)$ Takes as input a verification key V_{SIG}, a message M, and a signature σ. It outputs a single bit b.

We require that for all λ, all $(V_{\mathsf{SIG}}, K_{\mathsf{SIG}})$ output by $\mathsf{Gen}(1^\lambda)$, all $M \in \{0,1\}^{\leq p(\lambda)}$ and all σ output by $\mathsf{Sign}(K_{\mathsf{SIG}}, M)$, we have that $\mathsf{Vrfy}(V_{\mathsf{SIG}}, M, \sigma) = 1$.

A signature scheme is secure if any adversary is unable to forge a valid message/signature pair. Security definitions for signature schemes are formalized by Goldwasser, Micali, and Rivest [GMR88]. As is true for encryption schemes, notions for signature can be classified by the combination of security goals and attack types. We consider the strongest type: (strong) existentially unforgeable against chosen message attack.

Definition 2.8 (EUF-CMA Security of Signature Scheme)**.** A signature scheme is secure in the sense of EUF-CMA or sEUF-CMA (weakly /strongly existentially unforgeable against chosen message attack) if for all polynomial time adversaries $\mathcal{A}$, the following probability is a negligible function in λ:

$$\Pr[\ (V_{\mathsf{SIG}}, K_{\mathsf{SIG}}) \leftarrow \mathsf{Gen}(1^\lambda); (M^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(K_{\mathsf{SIG}}, \cdot)}(1^\lambda, V_{\mathsf{SIG}}) : \mathsf{Vrfy}(V_{\mathsf{SIG}}, M^*, \sigma^*) = 1\ ],$$

where the restriction for each notion is as follows.

- For notion EUF-CMA: $M^* \notin \{M_1, \ldots, M_q\}$.
- For notion sEUF-CMA: $(M^*, \sigma^*) \notin \{(M_1, \sigma_1), \ldots, (M_q, \sigma_q)\}$.

where (M_i, σ_i) is the i-th queried message and answer pair.

It is known that the existence strongly unforgeable signature scheme is equivalent to one-way function [NY89, Rom90].

A weaker notion is that of a *one-time signature scheme* [Lam79, Mer89], in which the adversary is allowed to request only one signature from the Sign oracle before attempting a forgery. Although secure signature schemes and one-time signature schemes may both be constructed from one-way functions, known constructions of one-time signature schemes are more efficient.

2.4.4 Notions of Access Structures and Secret Sharing Schemes

In order to formalize "accessability" to some target information in full generality, the notions of access structures are useful and will appear as a recurring theme throughout the thesis. We thus capture it in this preliminary chapter.

Access Structures over Sets. We now describe some terminology on access structures over sets. Consider a (possibly infinite) set N. We call the subsets in N which are allowed to derive some sort of secret *qualified*, and the subsets in N who should not be able to obtain any information about the secret *forbidden*. We call $\Theta \in 2^N$ a *qualified access structure* if Θ is monotone non-decreasing, *i.e.,* for each $\boldsymbol{v} \in \Theta$ we have that if $\boldsymbol{v} \subseteq \boldsymbol{w}$ then $\boldsymbol{w} \in \Theta$. We denote by $\Theta^{(\mathsf{min})}$ the collection of *minimal sets* in Θ. We call $\Psi \in 2^N$ a *forbidden access structure* if Ψ is monotone non-increasing, *i.e.,* for each $\boldsymbol{v} \in \Theta$ we have that if $\boldsymbol{w} \subseteq \boldsymbol{v}$ then $\boldsymbol{w} \in \Theta$. We denote by $\Psi^{(\mathsf{max})}$ the collection of *maximal sets* in Ψ. The tuple (Θ, Ψ) is called an access structure pair on N if $\Theta \cap \Psi = \emptyset$. If $\Theta \cup \Psi = 2^N$, then we say that (Θ, Ψ) is complete.

Since we often refer to the qualified one throughout the chapter, when we refer to access structure we mean a qualified access structure, unless otherwise is specified. For an access structure Θ, we call $\bigcup_{\boldsymbol{w} \in \Theta} \boldsymbol{w}$ the support of Θ and denote it by $\mathsf{Sup}(\Theta)$.

Secret Sharing Schemes. A secret sharing scheme permits a secret to be shared among participants of a base group N in such a way that only qualified subsets of participants can recover the secret. Such qualified subsets are captured by a qualified access structure.

Definition 2.9 (Secret Sharing Scheme)**.** A secret sharing scheme for access structure Θ over $N = \{1, \ldots, n\}$ consists of two algorithms $\mathsf{Share}_{n,\Theta}, \mathsf{Recon}_{n,\Theta}$:

$\mathsf{Share}_{n,\Theta}(M)$ A probabilistic algorithm for sharing that takes as input a secret M. It outputs the set of shares $(s_1, \ldots, s_n)$ and a public information pub.

$\mathsf{Recon}_{n,\Theta}(\mathbf{X}, (s_j)_{j\in\mathbf{X}}, \mathsf{pub})$ A deterministic algorithm for reconstructing that takes as input a set $\mathbf{X} \in \Theta$, the set of shares $(s_j)_{j\in\mathbf{X}}$, and the public information pub. It outputs the secret M.

The correctness property states that for all M, for all $(s_1, \ldots, s_n, \mathsf{pub})$ output from $\mathsf{Share}_{n,\Theta}(M)$ it holds that $\mathsf{Recon}_{n,\Theta}(\mathbf{X}, (s_j)_{j\in\mathbf{X}}, \mathsf{pub}) = M$.

Definition 2.10 (Computational Security of Secret Sharing Scheme)**.** A secret sharing scheme for access structure Θ over set N is computationally secure against a forbidden structure $\Psi \subseteq 2^N \setminus \Theta$ if for all polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ the following is negligible:

$$\Pr[(M_0, M_1, \mathbf{Y}) \leftarrow \mathcal{A}_1(1^\lambda, \Theta, \Psi);\ b \leftarrow \{0,1\};\ (s_1, \ldots, s_n, \mathsf{pub}) \leftarrow \mathsf{Share}_{n,\Theta}(M_b) : \mathcal{A}_2((s_j)_{j\in\mathbf{Y}}, \mathsf{pub}) = b \quad \wedge \quad \mathbf{Y} \in \Psi].$$

2.5 Graphs and Posets

Graph. A *graph* is a pair $G = (V, E)$ of sets satisfying $E \subseteq \binom{V}{2}$; thus, the elements of E are 2-element subsets of V. The elements of V are the *vertices* (or *nodes*) of the graph G, the elements of E are its *edges*. When $G = (V, E)$ we sometimes refer $V = V_G, E = E_G$ for clarity. We will consider only a *simple* graph, where E is threaten as a set without repetition, *i.e.,* not multi-set. If $V' \subseteq V$ and $E' \subseteq E$ then $G' = (V', E')$ is a *subgraph* of $G = (V, E)$, written as $G' \subseteq G$. Less formally, we say that G' is in G. We say that two graphs are disjointed if their sets of nodes are disjointed.

A *path* is a graph $P = (V, E)$ of the form $V = \{x_0, ..., x_l\}$, and $E = \{\{x_0, x_1\}, ..., \{x_{l-1}, x_l\}\}$ where the x_i are all distinct. We often refer to a path by the sequence of its nodes, writing, say, $P = (x_0, x_1, ..., x_l)$. A graph G is called *connected* if any two of its nodes are linked by a path in G. A sequence $(x_0, x_1, ..., x_l, x_0)$ where $((x_0, x_1, ..., x_l)$ is a path is called a *cycle*. An *acyclic* graph, one not containing any cycles, is called a *forest*. A connected forest is called a *tree*. A tree with one node fixed as a special node is called a *rooted tree*, and the special node is called the *root* of tree. A node of degree 1 in the tree is called a *leaf*. A forest in which all maximal connected trees are rooted trees is called *rooted forest*.

A *directed graph* is a graph $G = (V, E)$ together with two maps $\mathsf{init} : E \to V$ and $\mathsf{ter} : E \to V$ assigning to every edge $e \in E$ an initial vertex $\mathsf{init}(e)$ and a terminal vertex $\mathsf{ter}(e)$. The edge is said to be directed from $\mathsf{init}(e)$ to $\mathsf{ter}(e)$, written as $e = (\mathsf{init}(e), \mathsf{ter}(e))$. A *directed path* or *chain* is a path of sequence $(x_0, ..., x_l)$ with the edge $\{x_{j-1}, x_j\}$ being directed from x_{j-1} to x_j for all $j = 1, ..., l$. A directed cycle is defined analogously. A *directed acyclic graph* is a directed graph with no directed cycle in it.

A directed rooted tree such that for each edge $e \in E$, $\mathsf{init}(e)$ is nearer to the root than $\mathsf{ter}(e)$ is called *root-ordered tree*. A *root-ordered forest* is defined analogously for a directed rooted forest.

Decomposition of Graph. A node-partitioned decomposition of a graph G is a family of subgraphs $G_1, ..., G_k$ whose sets of nodes partition V_G.

- When G is a directed acyclic graph and each G_j is a chain, we have a chain-decomposition.
- When G is an undirected graph and each G_j is a tree, we have a tree-decomposition (for undirected graph).
- When G is a directed acyclic graph and each G_j is a root-ordered tree, we have a tree-decomposition (for directed acyclic graph).

Partially Ordered Set. A partially ordered set is a set with partial order relation. It is defined as follows.

Definition 2.11 (Partially Ordered Set, Poset)**.** A poset $\Gamma = (\mathcal{S}, \preccurlyeq)$ is a set $\mathcal{S}$ on which an order relation $\preccurlyeq$ is defined, satisfying

1. $x \preccurlyeq x$ for all $x \in \mathcal{S}$,
2. if $x \preccurlyeq y$ and $y \preccurlyeq x$ then $x = y$,
3. if $x \preccurlyeq y$ and $y \preccurlyeq z$ then $x \preccurlyeq z$.

We say that x and y are comparable if $x \preccurlyeq y$ or $y \preccurlyeq x$. If $x \preccurlyeq y$ and $x \neq y$ we write $x \prec y$. If $x \prec y$ and there is no z such that $x \prec z \prec y$ we say that y covers x, written as $x \prec_{\mathsf{c}} y$. Any x with no y such that $y \prec x$ is called a minimal element. An example of poset is inclusion poset, where $\mathcal{S} \subseteq 2^N$ with relation $\subseteq$.

One can represent a poset $\Gamma = (\mathcal{S}_\Gamma, \prec)$ as a directed acyclic graph G by setting

$$V_G = \mathcal{S}_\Gamma, \qquad E_G = \{(x, y) : x \prec y\}. \tag{2.1}$$

This is the maximal representation since it shows all the order relations in the poset. On the other hand, the minimal representation is the one with $E_G = \{(x, y) : x \prec_{\mathsf{c}} y\}$. A chain (tree, resp.) decomposition of poset is a chain (tree) decomposition of its maximal representation graph. For clarity, we call each edge in the maximal representation graph a *ordering-edge*; and each edge in the minimal one a *covering-edge*.

Chapter 3

Practical Symmetric-key Broadcast Encryption

3.1 Introduction

Broadcast encryption (BE) involves 1 broadcaster and n receivers. Each receiver is given a unique private key. The broadcaster is given a private broadcaster key. The broadcaster wishes to broadcast messages to a designated set $P \subseteq N = \{1, ..., n\}$ of receivers. Any receivers in P should be able to decrypt the broadcast message using only its private key while a coalition $F \subseteq N \setminus P$ (revoked users) should not be able to do so. As mentioned in Chapter 1, such a scheme is motivated largely by pay-TV systems, the distribution of copyrighted materials such as CD/DVD.

Broadcast encryption schemes were first formalized by Fiat and Naor [FN93]. Since then, many variants of the basic problem were proposed. The arguably most challenging variant is the case where it satisfies the following properties.

Arbitrary Broadcast The designated subset P can be an arbitrary subset of N.

Fully Collusion-Resistant Security of the scheme is guaranteed even all the users outside P are colluded, *i.e.,* the revoked user set equals to $F = N \setminus P$.

Stateless Receiver The private key stored by each user is fixed from the initialization.

The main goal for the research in this area is to construct efficient schemes that satisfy the above variant and require only small size of both the header of broadcast (communication cost) and the private key at each user (storage cost) as a function of the number of users n (and/or the number of revoked users, $r := n - |P|$). Here, the *header* is the encapsulation of session key that is used to encrypt data.

A scheme which solves above mentioned variant problem and satisfies good efficiency in only one side is trivial. The two trivial schemes are as follows.

- On one side, consider the broadcast encryption in which each user has one unique user key of constant bit length. To broadcast an encrypted data, a center encrypts the session key with each key of privileged users and then transmits them all as the header. This construction yields only one key at each user but a large number of ciphertexts in the header as linear to $|P|$.

- On the opposite side, each user has all keys of subsets in which he is one of the members. To broadcast an encrypted data, a center just let the session key be the key corresponding to a privileged subset, hence yields no header. However each user is required to store a large amount of keys, which equals to 2^{n-1}.

Depending on applications, we may categorize to two following cases.

Large Broadcast, Small Revocation This is the case where $r \ll n$. One of the most important example of applications in this case is copy protection for DVD, where typically the number of illegal keys to be revoked should be small.

Small Broadcast, Large Revocation This is the case where $r \approx n$. An example of applications maybe irregular broadcast of TV programs where only small number of users are interested in.

For the case of large revocation, it turns out that the first trivial scheme described above is the only efficient solution (except for one recent scheme by Boneh, Gentry, Waters [BGW05], see below). On the other hand, there are many proposed schemes which are useful in the small revocation case. As being attractive by important applications, we also focus on this case.

An efficient solution for the case of small revocation which is considered a ground work to many consequences is the Complete (binary) Subtree scheme (CS) by Naor et al. [NNL01]. Schemes which were considered the current state of the art (before two very recent works, see below) are: (i) Pseudo-random sequences generator (PRSG) based schemes such as the Subset Difference scheme (SD) [NNL01], its refinement–the Layered SD scheme (LSD) [HS02], and their somewhat generalizations in [AKI03a]. (ii) RSA accumulator based schemes such as Asano's scheme [Asa02], and its optimal generalizations in [AKI03b, GR04]. See Table 1 for the efficiency comparison. No scheme above could achieve simultaneously small header size independent of n, small key size of $O(\log n)$, while keeping computational cost and all other costs grow only sub-linear in n.

More recently, Goodrich et al. [GST04] and Wang et al. [WNR04] independently propose more efficient schemes that break the above barrier. In particular, they achieve simultaneously header size of $O(r)$ and key size of $O(\log n)$, and computational cost of $O(n^{1/k})$ for arbitrary constant k. (In fact, in [WNR04] only the case when $k = 1, 2$ is considered).

In this chapter, we propose generic frameworks for constructing broadcast encryption and give some efficient instantiations. One of our instantiations (Instantiation 2 in Table 1) achieves not only small header size of $O(r)$ but also small key size of $O(1)$ with no extra non-secret storage, while keeping computational cost $O(n^{1/k} \log^2 n)$, which grows only sub-linear in n. Thus this is the first scheme that achieves header and private key size independent of n while keeping computational cost sub-linear in n, with no extra non-secret storage. The contributions in more detail are described below.

3.1.1 Our Contributions

In the general subset-cover paradigm of [NNL01], which includes almost all of the above schemes, it has been *implicitly* understood that one can separate the design of such a scheme into two seemingly orthogonal problems namely: designing combinatorial set system which enables subset covering (this step determines the header size), and defining computational key derivation (this step determines the private key size and computational cost). This is

Table 3.1: Comparison among symmetric-key broadcast encryption schemes. (k, a, c are arbitrary constant parameters. negl is for negligible.)

		Header size Complexity	$\leq$	Priv. key size	Comp. cost (bit complexity) Prime-gen	Others	Non-secret Storage
CS	[NNL01]	$O(r\log(\frac{n}{r}))$		$\log n+1$	-	$O(\log\log n)$	negl
PRSG or OWF -based ↓							
SD	[NNL01]	$O(r)$	$2r-1$	$O(\log^2 n)$	-	$O(\log n)$	negl
LSD	[HS02]	$O(r)$	$2kr-k$	$O(\log^{1+1/k} n)$	-	$O(\log n)$	negl
GST	[GST04]	$O(r)$	$4kr$	$2\log n$	-	$O(n^{1/k})$	negl
WNR	[WNR04]	$O(r)$	$4r$	$2\log n$	-	$O(n^{1/2})$	negl
HLL_1	[HLL05]	$O(r)$	$2kr$	$\log n+k$	-	$O(n^{1/k})$	negl
Instantiation 1		$O(r)$	$2kr$	$\leq \log n+1$	-	$O(n^{1/k})$	negl
Instantiation 4		$O(r)$	$2r$	$\leq k(\log n+1)$	-	$O(kn^{1/k})$	negl
HL	[HL06]	$O(r)$	$2r$	$k(\log n+1)$	-	$O(kn^{1/k})$	negl
$\mathsf{JHC}+_2$	[JHC+05]	$O(r)$	r/c	$O(n^c)$	-	$O(n-r)$	negl
RSA Accumulator -based ↓							
Asano	[Asa02]	$O(r\log_a(\frac{n}{r})+r)$		1	$O(2^a\log_a^5 n)$	$O(2^a\log_a^2 n)$	negl
GR	[GR04]	$O(r\log_a(\frac{n}{r})+r)$		1	$O(a\log_a^5 n)$	$O(a\log_a^2 n)$	negl
Instantiation 3		$O(r\log_a(\frac{n}{r})+r)$		1	$O(1)$	$O(a\log n)$	negl
Instantiation 2		$O(r)$	$2kr$	1	$O((\log^5 n)/k^5)$	$O((n^{1/k}\log^2 n)/k)$	negl
Bilinear Map -based ↓							
BGW	[BGW05]	$O(1)$	1	1	-	$O(n-r)$	$O(n)$

first explicitly characterized in [GR04] for the case of Akl-Taylor's RSA based key derivation [AT83].

Framework. In this paper, we characterize the two orthogonal components in general. We then explicitly present three generic sub-frameworks for computational key derivation component (*generic* as arbitrary set systems are applicable): (1) PRSG based technique (re-formalizing from [AKI03a] so as to be consistent with presentations here), (2) non-trapdoor-RSA accumulator based, and (3) trapdoor-RSA accumulator based techniques. The non-trapdoor RSA based one is a generalization and also an optimization of Akl-Taylor's technique. The trapdoor RSA based framework is an extension of the non-trapdoor RSA based one and can achieve better efficiency performance, albeit its description is somewhat more complicated.

The main issue is that we characterize three sub-frameworks in such a way that such instantiations in these frameworks and their resulting efficiencies will depend solely on properties related to *graph decompositions* of the set systems being instantiated; while in the same time the security will be guaranteed *automatically* from the general frameworks. The PRSG based framework will be based on *tree decomposition*, and the two RSA based frameworks will be based on *chain decomposition*; both are purely combinatorial. Therefore the whole paradigm abstracts away the computational security issues and reduces the problem to only pure combinatorics. Moreover it allows modularity in designing a scheme: it is a matter of finding a set system which yields a good header size in the first step, and then finding a graph decomposition of that set system that yields a good private key size and computational cost.

As for the generic efficiency characterization, both RSA based frameworks achieve key size of $O(1)$ for all instances. One generic property of the trapdoor based framework that makes it superior to the non-trapdoor based one is that when restricting to the same asymp-

totic resources and instantiating the same set system[1], if the non-trapdoor based one allows n users in the scheme, then the trapdoor based one will allow n^k users for any (arbitrary large) constant k. Indeed, the costs due to prime generation are exactly the same (not only asymptotically).

Efficient Instantiations. For the combinatorial set system component, all of our schemes are based on new set systems we call Subset Incremental chain (SIC) and Layered-SIC (LSIC) which are designed so to achieve small header size as being $O(r)$ while intrinsically have graph decompositions with good properties. For the computational key derivation component, we instantiate the LSIC set system by presenting their graph decompositions, resulting in various concrete schemes upon each sub-framework as follows. We use the notation $(\mathsf{X})^{\mathsf{y}}$ to denote an instantiation of the set system X using the y-based framework. Denote $\mathsf{LSIC}[k]$ as LSIC with parameter k. Note that $\mathsf{LSIC}[1] = \mathsf{SIC}$.

Instantiation 1 : $(\mathsf{LSIC}[k])^{\mathsf{prsg}}$. This scheme directly improves the scheme of [GST04, WNR04] (and it is fair to compare with since the same assumption, PRSG, or equivalently one-way function, was used). In particular it can reduce some overheads, albeit only within constant terms in the worst case: the worst-case key sizes are half of those in [GST04, WNR04]. Indeed the key size in our scheme is non-uniform among users; some users are even required to store only constant-size keys (cf. Theorem 3.12, 3.15, and Eq.(3.4)). Our scheme also reduces the computational cost from [GST04], but only in the average case (the worst-case costs are asymptotically the same).

Instantiation 2 : $(\mathsf{LSIC}[k])^{\mathsf{acc}}, (\mathsf{LSIC}[k])^{\mathsf{tacc}}$. Note that (t)acc is for (trapdoor) accumulator. The performance of this scheme is as mentioned previously. It is the first scheme that achieves header and private key size independent of n while keeping computational cost sub-linear in n, with no extra non-secret storage. The number of primes used per user is optimal as being $O(\log n)$ for $(\mathsf{LSIC}[\mathsf{k}])^{\mathsf{acc}}$ and further reduced to $O((\log n)/k)$ for $(\mathsf{LSIC}[\mathsf{k}])^{\mathsf{tacc}}$ (so that the on-the-fly prime generation cost is $O((\log^5 n)/k^5)$). Had one used the non-optimal Akl-Taylor's framework as put forth to the context of BE by [Asa02, AKI03b, GR04], it would be $O(n^{1/k} \log n)$ which is super-logarithmic (and the prime generation cost would be $O(n^{1/k} \log^5 n)$).

Instantiation 3 : $(\mathsf{LSIC}[\log_a n])^{\mathsf{tacc}}$. This scheme improves Gentry and Ramzan's scheme [GR04], which itself is more efficient than the above schemes in the aspect of asymptotic computational cost. Our scheme reduces *poly-logarithmic* cost due to prime generation, which was the dominant cost, to only a *constant* one without affecting the other parameters. Among the *constant-key-size* schemes with header size $O(r \log_a(n/r) + r)$ and no extra non-secret storage, this is the first one in the literature that achieves $O(\log n)$ overall computational cost. (And in fact, ours uses only a constant number of primes). The previous improvement for this class of schemes was done by [GR04] to improve [Asa02] but only in the constant term involving a. (cf. Table 1).

[1]To be more precise, we indeed do not instantiate the same set system in this comparison; instead, we instantiate a set system, say A, in the non-trapdoor based framework and a *hierarchical version* of A in the trapdoor based framework. See for more details in Section 3.6.1.

Instantiation 4 : $(\mathsf{SIC})^{\mathsf{prsg,new}}$. By using a new tree decomposition different from the one used in the first instantiation, we achieve a new scheme that trades the private key size for smaller header size. Analogously to the instantiation 1, the private key size is non-uniform: it varies from 1 to $k(\log n + 1)$.

As a nutshell guide for selecting schemes, if overall good performances are required, the instantiation 2 is recommended; if the computational cost is more restricted, use the instantiation 3; if the underlying cryptographic primitive is restricted to only weak one (namely PRSG, and not RSA or bilinear map), use the instantiation 1 for smaller key size or use the instantiation 4 for smaller header size.

3.1.2 Recent Related Works

At the same time as when we publicly announced our first preliminary version [AI05b] (in January 2005), Boneh et.al. [BGW05] presented a *public-key* broadcast encryption scheme which achieves size $O(1)$ for both header and private key. However, the computational cost is $O(n - r)$ (which, in particular, is not sub-linear in n) and the size of the public key to be used by an encrypter, which is also the non-secret storage needed for the decrypter, is $O(n)$. The second scheme in [BGW05] reduces these computational cost and non-secret storage size to $O(\sqrt{n})$ but with the price of the increased header size as $O(\sqrt{n})$, which is not independent of n anymore.

Also at almost the same time, Jho et.al. [JHC+05] proposed some efficient schemes with small header size when r is not too small. However, their schemes do not enjoy practical *asymptotic* performances as either the header size is $c_1 r + c_2 n = O(n)$ (for some constant c_1, c_2) or the key size is $\binom{n-1}{c} = O(n^c)$ (for some $c \geq 2$) for their best two schemes.

More recently, in August 2005, Hwang et al. (scheme $\overline{\mathsf{B1}}$ in [HLL05]) independently proposed a similar scheme as our instantiation 1, but the private key size is $\log n + k$, which is larger than ours.

We presented our fourth instantiation [AI05c] first in November 2005. Independently, Hwang and Lee [HL06] (in March 2006) presented a scheme with a similar performance, albeit with the user key size being uniformly $k(\log n + 1)$, larger than ours on average.

Finally, it is worth noting that Boneh and Silverberg [BS03] showed that n-linear maps can be used to construct an optimal public-key scheme with constant private key, public key, and header size. However, there are currently no known constructions for such a map for $n > 2$.

3.1.3 Organization of the Chapter

In Section 3.1.4, we exhaustively survey earlier works in the area related to broadcast encryption; this subsection can be skipped without loss of understanding of the main context. We then start the main context by first formalizing the subset-cover framework separately into two components, combinatorial set system and computational key derivation as mentioned above, in Section 3.2.2. Some terminology is then described in Section 3.2.3. New set systems, SIC and LSIC, are presented in Section 3.3. The three frameworks for key derivation which are PRSG-based, Non-trapdoor RSA accumulators-based, and Trapdoor RSA accumulators-based, are described in Section 3.4.1, 3.5.1, and 3.6.1 resp. We then give the instantiations of the proposed set systems in Section 3.4.2, 3.5.2, and 3.6.2 from the three frameworks, respectively. We then conclude in Section 3.7.

3.1.4 Survey on Earlier Works

For completeness, we also survey earlier related works. The research area of broadcast encryption is extensive and our survey turns to be lengthy than first planning. Since this is not directly related to our proposal, the reader may just skip this part without loss of understanding.

Schemes surveyed in this section are those that were presented before 2001. All the schemes do not satisfy the above mentioned variant: they are either having limitation on privileged user sets, not fully collusion-resistant, or not stateless.

To characterize such schemes and compare previous results we should fix our notation. Two important characterizations that diverge different schemes are the first two aspects in the list concerning variants mentioned before, i.e., fixed, bounded or unbounded size of privileged and/or forbidden subsets. Here we naturally generalize it to the access structure of privileged subsets and forbidden subsets (e.g.,threshold structure is a special case). Let 2^N denote the collection of all subsets of N. We denote by $\mathcal{P} \subseteq 2^N$ a collection of all possible *privileged subsets* to which the broadcaster sends a broadcast; and $\mathcal{F} \subseteq 2^N$ a collection of all possible coalitions (called *forbidden subsets*) against which each broadcast is to remain secure. Informally, such a scheme satisfying these structures is called $\mathcal{F}$-secure $\mathcal{P}$-broadcast encryption scheme or $(\mathcal{P}, \mathcal{F})$-BE. Denote $[\leq a] = \{P \in 2^N : |P| \leq a\}$ (and similarly for $[\geq a], [= a]$). We often omit [ ] when it does not cause ambiguity.

In high-level, we categorize previous works in the literature by the lines of researches which are usually diverged by their different approaches.

Information-theoretically Secure Stateless Schemes.

The early works in the area of broadcast encryption are done in the information-theoretically secure setting. Informally, such schemes are secure against any adversaries with unbounded resource. Consequently, such schemes require no computational intractability assumption. Also, the private keys needs to be perfectly unrelated. Such a scheme can be used only single or bounded number of time(s), since every use will leak some secret information. All information-theoretically secure schemes are naturally stateless. We exhaustively survey results as follows.

One interesting special class of these contains the schemes which have header-size = 0, which is called *zero-header* broadcast encryption. This class of schemes is equivalent to another cryptographic tool called *Key Predistribution Scheme* (KPS). KPS allows a designated group of users to establish a common key non-interactively. Intuitively, using such a common key as a session key encrypting the data in broadcast encryption, header is not needed anymore, hence its name: zero-header.

Blom obtained a $(2, \leq k)$-KPS for any $k \leq n$ in [B84] by using MDS codes. The formalization of Blom's work is done by Matsumoto and Imai in [MI87]. Blundo *et al.* obtained a $(t, \leq k)$-KPS in [BSH+92] by using symmetric polynomials. Fiat and Naor presented a $(\leq n, \leq k)$-KPS in [FN93]. Blundo *et al.* found tight lower bounds on private key size for $(t, \leq k)$-KPS's [BSH+92] and for $(\leq n, \leq k)$-KPS's [BC94]. Stinson shows that general $(\mathcal{P}, \mathcal{F})$-KPS can be constructed from a simple combinatorial set system called $(\mathcal{P}, \mathcal{F})$-Key Distribution Pattern [Sti96]. Desmedt and Viswanathan presented a bound and an optimal construction for $(\leq n, \leq n)$-KPS [DV98]; this can be considered as a compliment of the Fiat and Naor $(\leq n, \leq n)$-KPS, this is indeed the second trivial construction mentioned before.

We now turn to general broadcast encryption schemes (not only zero-header). After Fiat-Naor's zero-header scheme is published, there are several subsequent works [BMS96, Sti96, SW98, KYDB98, LS98]. Blundo *et al.* gave a non-tight concrete lower bound regarding tradeoff between private key size and header size and a construction for $(t, \leq k)$-BE's from KPS as sub-schemes of smaller access structure [BMS96]. Stinson also gave a construction for $(\leq n, \leq k)$-BE's based on KPS [Sti96]. The implication from BE to KPS is proved by Kurosawa *et al.* [KYDB98], hence by using lower bounds of KPS in [BSH+92, BC94] they obtain directly a lower bound for $(t, \leq k)$-BE's. Luby and Staddon found some more-tight asymptotic bounds and constructions for $(= n - k, \leq k)$-BE's [LS98]. Kumar *et al.* [KRS99] proposed $(\geq n - k, \leq k)$-BE based on cover-free families and error correcting codes; their scheme which underlying cover-free families based on polynomials require private-key-size $= \mathcal{O}(k \log n)$, header-size $= \mathcal{O}(k \log n)$; while the one on algebraic-geometric codes require private-key-size $= \mathcal{O}(k \log n)$, header-size $= \mathcal{O}(k^2)$. Anzai *et al.* [AMM99] and Naor-Pinkas [NP00] independently proposed a $(\geq n - k, \leq k)$-BE based on Shamir's secret sharing scheme [Sha79]. Their schemes performs permanent type of revocation; while n does not have to be determined in advance! (we can add new users by not affecting present users). The resulting scheme has private-key-size $= \mathcal{O}(1)$, header-size $= \mathcal{O}(k)$. This scheme can be turned to computationally secure one by using secret sharing in the exponent of an generator in the group of prime order instead.

Note that in general, one can turn an information-theoretically secure broadcast encryption which the revocation is not permanent to a computationally secure one. This makes us possible to use the resulting scheme polynomially-many times, instead of single (or few bounded) time(s). This is done by changing the underlying encryption scheme used for encrypting a session key from an information-theoretically secure one (one-time pad) to a computationally secure one. Intuitively, this is since the private keys are still *perfectly unrelated*.

Computationally Secure Stateless Schemes.

Informally, this setting models the adversaries as ones with bounded resources, in particular as polynomial-time algorithms. Such schemes require computational intractability assumption in order to base the security on. Basically, computational security trades the security with a better efficiency. One of the main idea is to generate keys that are *computationally unrelated* to the view of forbidden users, instead of perfectly unrelated as in the previous setting.

Fiat-Naor in their seminal paper [FN93] presents several constructions of stateless $(\leq n, \leq k)$-BE; their best result requires private-key-size $= \mathcal{O}(k(\log k)(\log n))$, header-size $= \mathcal{O}(k^2(\log^2 k)(\log n))$. The multiple-times-use version of those perfectly-unrelated-key schemes such as Luby-Staddon [LS98], Kumar *et al.* [KRS99], Naor-Pinkas [NP00] described above improves this scheme. However, there were no scheme that achieves the access structure $(\leq n, \leq n)$ as desired. This was not until the scheme of Naor-Naor-Lotspiech [NNL01] is published, which is indeed inspired from efficient stateful schemes which utilize the simple binary tree structure.

Stateful Schemes and Multicast Security.

The first efficient stateful scheme appeared in the context of *multicast security*. It is not clear how to distinguish the meaning of multicast security and broadcast encryption exactly. Very

roughly speaking, multicast security considers the scenario of *re-keying* a group key, which is known by *all the members at the present time*, say m members, when some users outside those members joins the group or some users in the present time leaves (or, is revoked from) the group. The efficiency is considered as a function of m, rather than the number of all users, n, as in broadcast encryption, since such n is not determined in advance for multicast. Put in another word, while the goal is the same, they differ as multicast security deals with the set of present-time users while broadcast encryption deals with the privileged subset of the set of all users. Due to the fact that the group key seamlessly relies on the status relative to the present-time, such approaches to multicast security are usually inevitably stateful.

The first efficient construction was proposed independently by Wallner *et al.* [WHA97] and Wong *et al.* [WGL98]. Their methods define a logical tree and a node key for each node of the tree. Each receiver is assigned to a leaf of the tree and given a set of node keys defined for the nodes on the path from the leaf to the root. Therefore, private-key-size $= \log m + 1$, assuming that the system uses a binary tree. All of these keys except one are shared by other receivers. This method revokes one receiver at a time, and updates all keys stored by non-revoked receivers, which have also been owned by the revoked receiver. This requires header-size $= 2 \log m$ for one-user revocation. If the system needs to revoke r users by repeating the single revocation, it requires header-size $= 2r \log m$.

Since then, many modifications of these have been proposed in [MS98, CMN99]. Some of them reduce header-size to $\log m$ for single revocation. McGrew *et al.* [MS98] used a one-way function, Canetti *et al.* [CMN99] used a pseudo-random generator. The private-key-size remains the same, while their methods increase the computational overhead at a receiver, namely, each receiver needs to perform the computation of such a technique at most $\log m$ times.

3.2 Framework and Some Preliminaries

3.2.1 Definitions and Security Notions of Broadcast Encryption

We formulate broadcast encryption as a key encapsulation mechanism (KEM)as follows.

Definition 3.1. (Private-key Broadcast Encryption) A private-key broadcast encryption scheme consists of three polynomial-time algorithms (Setup, Enc, Dec):

$\mathsf{Setup}(1^\lambda, n)$ Takes as input a security parameter 1^λ; and the number of users n. It outputs private keys $\mathsf{sk}_1, ..., \mathsf{sk}_n$, a broadcaster secret key bk, and some public parameter pub.

$\mathsf{Enc}(\mathsf{bk}, S)$ Takes as input the broadcaster secret key bk and a privileged subset S. It outputs a message encryption key K and a header hdr.

$\mathsf{Dec}(\langle u, \mathsf{sk}_u \rangle, S, \mathsf{hdr}, \mathsf{pub})$ Takes as input a receiver key $\langle u, \mathsf{sk}_u \rangle$, the full header (S, hdr), and pub. If $u \in S$ then it outputs K or a special symbol $\perp$ otherwise.

In usage, the ciphertext is $\mathsf{hdr} || E_K(M)$ where E is a symmetric encryption scheme and M is a plaintext.

Definition 3.2. (IND-CCA Security of BE). A private-key broadcast encryption $\mathsf{BE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ is secure in the sense of IND-ATK where $\mathsf{ATK} \in \{\mathsf{CCA1}, \mathsf{CCA2}\}$ if for

all polynomial time adversaries $\boldsymbol{K}$, its advantage $|\Pr[\mathbf{Exp}^{\mathsf{IND-ATK}}(\mathsf{BE}, \boldsymbol{K}) = 1] - \frac{1}{2}|$ is a negligible function in λ where we define

$$\begin{aligned}
&\text{Experiment } \mathbf{Exp}^{\mathsf{IND-ATK}}(\mathsf{BE}, \boldsymbol{K}) \\
&\quad \mathsf{Setup}(1^\lambda) \rightarrow (\mathsf{sk}_1, ..., \mathsf{sk}_n, \mathsf{bk}, \mathsf{pub}); \\
&\quad (S^*, \mathsf{state}) \leftarrow \boldsymbol{K}_{\mathsf{find}}^{\mathcal{O}_1}(1^\lambda, \mathsf{pub}) \\
&\quad \mathsf{Enc}(\mathsf{bk}, S^*) \rightarrow (K^*, \mathsf{hdr}^*); \\
&\quad \{0,1\} \rightarrow b; \\
&\quad \mathsf{RR}(K^*, b) \rightarrow K^\dagger; \\
&\quad b' \leftarrow \boldsymbol{K}_{\mathsf{guess}}^{\mathcal{O}_2}(K^\dagger, \mathsf{hdr}^*, \mathsf{state}) \\
&\text{return 1 iff } b = b'
\end{aligned}$$

where $\mathcal{O}_1$ denotes the oracles $\mathsf{Corrupt}(\cdot)$, $\mathsf{Enc}(\mathsf{bk}, \cdot)$, $\mathsf{Dec}(\langle \cdot, \mathsf{sk}_{(\cdot)} \rangle, \cdot, \cdot, \mathsf{pub})$ while $\mathcal{O}_2$ is the same if $\mathsf{ATK} = \mathsf{CCA2}$ or nothing otherwise. $\mathsf{Corrupt}(S)$ takes as input $S \subset N$ and returns all the secret keys for users in S. It is required that $F \cap S^* = \emptyset$, where F denotes the set of all corrupted identities of users. Also it is restricted that $\boldsymbol{K}_{\mathsf{guess}}$ does not ask the decryption of the challenge ciphertext hdr^*.

3.2.2 Framework

Now we formalize the subset-cover framework [NNL01] separately into two independent components as follows.

Combinatorial Set System Component

We first redefine a set system which is useful for such a scheme in this framework called complement-cover set system. Such a set system is a family of subsets of a universe with the property that every subset of the universe can be efficiently partitioned to a union of some collection of subsets in the family.

Definition 3.3. (Complement-Cover Set System) For a map $\mathsf{c} : \mathbb{Z}_{>0}^2 \rightarrow \mathbb{Z}_{>0}$, a set system $\mathcal{S} = \{S_1, ..., S_m\}$ over a base set $N = \{1, ..., n\}$ is c-*complement-cover* if there is a polynomial-time algorithm $\mathsf{Cover}(\cdot)$ such that upon input any subset $R \subset N$, outputs $\{S_{i_1}, ..., S_{i_t}\}$ for some $1 \leq i_1, \ldots, i_t \leq m$ such that $N \setminus R = \bigcup_{j=1}^{t} S_{i_j}$ and that $t \leq \mathsf{c}(n, |R|)$. □

As usual n, r is the number of all users and revoked users respectively. Such a $\mathsf{c}(n, r)$-complement-cover set system yields a broadcast encryption scheme in the subset-cover framework with the header size $\mathsf{c}(n, r)$. The scheme is described formally in 3.2.2 and sketched here as follows. The broadcaster defines a subset key for each subset in the family. Each user stores a set of keys in such a way that he can derive all the keys of subsets (in the family) that he is a member. (Thus, the easiest way to do is to store them all. However to reduce the storage of keys, it would be better to store only some and derive the others from those stored keys on the fly. Such derivation patterns are predefined by the broadcaster.) To revoke the set R of users, the broadcaster just let a header to be a session key encrypted with each key of subsets in the partition of $N \setminus R$. Thus the header size is $\mathsf{c}(n, r)$. We often denote $\mathsf{c}_{\mathsf{X}}(n, r)$ for $\mathsf{c}(n, r)$ of the set system $\mathcal{S}_{\mathsf{X}}$, where X is the name of that set system.

Computational Key Derivation Component

We formalize the specification on key derivations in the context of access control scheme as the following. Denote by $\mathsf{k}(S)$ the subset key for $S \in \mathcal{S}$ and $\mathsf{p}(u)$ the private key of $u \in N$. Informally, the security of such a scheme requires that with $\mathsf{p}(u)$, one can derive $\mathsf{k}(S)$ if and only if $u \in S$; moreover, the collusion $N \setminus S$ cannot derive it.

Definition 3.4 (Access Control Scheme, AC)**.** An Access Control Scheme AC for a set system $\mathcal{S}$ over a base set N is a 2-tuple of polynomial-time algorithms $(\mathsf{Keygen}, \mathsf{Derive})$, where:

$\mathsf{Keygen}(1^\lambda)$**:** Takes as input a security parameter 1^λ. It returns all $\mathsf{k}(S_i)$'s, all $\mathsf{p}(u)$'s, and public parameter pub.

$\mathsf{Derive}(\langle u, \mathsf{p}(u)\rangle, S_i, \mathsf{pub})$**:** Takes as input $u \in N$, the key $\mathsf{p}(u)$, $S_i \in \mathcal{S}$, and pub. It returns $\mathsf{k}(S_i)$ if $u \in S_i$, or special symbol $\perp$ otherwise. □

The security notions are formalized as Key-Indistinguishability (following [NNL01]) and Key-Intract-ability. In what follows, we denote by $\mathsf{RR}(x, b)$ the real-or-random oracle, which outputs x if $b = 1$, otherwise a random string with the same length as x.

Definition 3.5 (Key-Indistinguishability of AC)**.** An access control scheme AC for a set system $\mathcal{S}$ is secure in the sense of Key-Indistinguishability (KIND-1 or KIND-2) if for all polynomial-time randomized adversary $\boldsymbol{A}$, its advantage $|\Pr[\mathbf{Exp}^{\mathsf{KIND}\text{-}l}(\mathsf{AC}, \boldsymbol{A}) = 1] - \frac{1}{2}|$ is a negligible function in λ where we define

$$
\begin{array}{rcl}
\text{Experiment } \mathbf{Exp}^{\mathsf{KIND}\text{-}l}(\mathsf{AC}, \boldsymbol{A}) & & \\
\mathsf{Keygen}(1^\lambda) & \rightarrow & (\{\mathsf{k}(S)\}_{s\in\mathcal{S}}, \{\mathsf{p}(u)\}_{u\in N}, \mathsf{pub}); \\
(S^*, \mathsf{state}) & \leftarrow & \boldsymbol{A}_{\mathsf{find}}^{\mathsf{Corrupt}(\cdot)}(1^\lambda, \mathsf{pub}, \mathcal{S}); \\
\{0,1\} & \rightarrow & b; \\
b' & \leftarrow & \boldsymbol{A}_{\mathsf{guess}}(\mathsf{RR}(\mathsf{k}(S^*), b), \mathsf{state}); \\
\text{return 1 iff } b = b' \text{ and } S^{\mathsf{ask}} \cap S^* = \emptyset & &
\end{array}
$$

Where $\mathsf{Corrupt}(S)$ (can be asked once) returns $\{\langle u, \mathsf{p}(u)\rangle : u \in S\}$ and S^{ask} is the query made to $\mathsf{Corrupt}(\cdot)$. If $l = 1$, it is restricted that $S^* = N \setminus S^{\mathsf{ask}}$.

Definition 3.6 (Key-Intractability of AC)**.** An access control scheme AC for a set system $\mathcal{S}$ is secure in the sense of Key-Intractability (KINT-1 or KINT-2) if for all polynomial-time randomized adversary $\boldsymbol{A}$, its advantage $\Pr[\mathbf{Exp}^{\mathsf{KINT}\text{-}l}(\mathsf{AC}, \boldsymbol{A}) = 1]$ is a negligible function in λ where we define

$$
\begin{array}{rcl}
\text{Experiment } \mathbf{Exp}^{\mathsf{KINT}\text{-}l}(\mathsf{AC}, \boldsymbol{A}) & & \\
\mathsf{Keygen}(1^\lambda) & \rightarrow & (\{\mathsf{k}(S)\}_{s\in\mathcal{S}}, \{\mathsf{p}(u)\}_{u\in N}, \mathsf{pub}); \\
(S^*, K) & \leftarrow & \boldsymbol{A}^{\mathsf{Corrupt}(\cdot)}(1^\lambda, \mathsf{pub}, \mathcal{S}); \\
\text{return 1 iff } K = \mathsf{k}(S^*) \text{ and } S^{\mathsf{ask}} \cap S^* = \emptyset & &
\end{array}
$$

Where $\mathsf{Corrupt}(S)$ (can be asked once) returns $\{\langle u, \mathsf{p}(u)\rangle : u \in S\}$ and S^{ask} is the query made to $\mathsf{Corrupt}(\cdot)$. If $l = 1$, it is restricted that $S^* = N \setminus S^{\mathsf{ask}}$.

In 3.2.2, we present a simple conversion that converts any KINT-secure scheme to a KIND-secure scheme in the random oracle model. Note that although KIND-2 is seemingly

stronger than KIND-1 (since in the former the adversary has a freedom to choose the target subset S^* after the corrupt query while the latter does not allow to do so), both notions are indeed obviously equivalent up to polynomial reduction, of factor $O(|\mathcal{S}|)$. The same result holds for KINT.

Subset-Cover Broadcast Encryption

In this section, we state the formal description of the generic subset-cover broadcast encryption from [NNL01]. From a compliment-cover set system $(N, \mathcal{S})$ where $N = \{1, \ldots, n\}$, an access control scheme $(\mathsf{Keygen}, \mathsf{Derive})$ for this set system, and a symmetric encryption scheme (E, D) whose key space is the key space of the access control scheme, we construct a broadcast encryption (to be more precise, a private-key broadcast KEM) as follows.

Construction 3-1. Subset-Cover Broadcast Encryption

$\mathsf{Setup}(1^\lambda, n)$ Run $\mathsf{Keygen}(1^\lambda)$ to obtain $\{\langle S, \mathsf{k}(S)\rangle : S \in \mathcal{S}\}$, $\{\langle u, \mathsf{p}(u)\rangle : u \in N\}$, and pub. Let bk consist of all these elements. Let $\mathsf{sk}_u = \mathsf{p}(u)$.

$\mathsf{Enc}(\mathsf{bk}, S)$ Run $\mathsf{Cover}(N \setminus S)$ to obtain the cover of S say $\{S_{i_1}, \ldots, S_{i_t}\}$. It randomly chooses K from the message space of the symmetric encryption. It then lets

$$\mathsf{hdr} = \left(\langle i_1, \mathsf{E}_{\mathsf{k}(S_{i_1})}(K)\rangle, \ldots, \langle i_t, \mathsf{E}_{\mathsf{k}(S_{i_t})}(K)\rangle\right)$$

It outputs a message encryption key K and a header hdr.

$\mathsf{Dec}(\langle u, \mathsf{sk}_u\rangle, S, \mathsf{hdr}, \mathsf{pub})$ Parse $\mathsf{hdr} = (\langle i_1, C_1\rangle, \ldots, \langle i_t, C_t\rangle)$. If $u \notin S$ then outputs $\perp$. Otherwise there must be k such that $u \in S_{i_k}$. Run $\mathsf{Derive}(\langle u, \mathsf{sk}_u\rangle, S_{i_k}, \mathsf{pub})$ to obtain $\mathsf{k}(S_{i_k})$. It then outputs $\mathsf{D}_{\mathsf{k}(S_{i_k})}(C_k)$.

Naor et al. [NNL01] proved that broadcast encryption in the subset-cover paradigm, whose the access control component is secure in the sense of KIND1 is secure in the standard notion, namely IND-CCA1.

Theorem 3.7. *([NNL01]) Let w be the maximum cover size for the complement-cover set system. Suppose that* $(\mathsf{Keygen}, \mathsf{Derive})$ *is KINT-1-secure with the maximum adversarial advantage ϵ_{AC} and the symmetric encryption scheme is IND-CCA1-secure with the maximum adversarial advantage ϵ_{SE}. Then the private-key broadcast KEM above is IND-CCA1-secure with the maximum adversarial advantage $2w \cdot |\mathcal{S}| \cdot (\epsilon_{\mathsf{SE}} + 2|\mathcal{S}|\epsilon_{\mathsf{AC}})$.*

Without going into detail, we note that either starting with KIND1 or KIND2, the reduction cost to IND-CCA1 of BE is the same (cf. Theorem 3.7 in 3.2.2). Intuitively, this is since the IND-CCA game of BE allows the corrupt query to be asked polynomially many times but both KIND1 and KIND2 allow only once. Therefore, proving KIND2 security is not so interesting in this context. We defined the two adaptivity levels merely for an independent interest.

It is also worth noting that Dodis and Katz [DK05] use the technique involving multiple encryption to obtain a generic scheme which is IND-CCA2-secure.

Denote $(\mathsf{X})^{\mathsf{y}}$ to be the access control scheme for set system $\mathcal{S}_{\mathsf{X}}$ that is constructed via AC framework y. Denote $\mathsf{KeySize}_{(\mathsf{X})^{\mathsf{y}}}(u)$ to be the number of keys of u (i.e., $|\mathsf{p}(u)|$, when $\mathsf{p}(u)$

is treated as a set) and $\mathsf{CompCost}_{(\mathsf{X})^{\mathsf{y}}}$ to be the worst-case computational cost for Derive. Also denote $\mathsf{PubSize}_{(\mathsf{X})^{\mathsf{y}}}$ to be the public storage required exactly in the number of bits. We also refer $(\mathsf{X})^{\mathsf{y}}$ as a BE scheme via the complement-cover set system $\mathcal{S}_{\mathsf{X}}$. For any y, $\mathsf{HeaderSize}_{(\mathsf{X})^{\mathsf{y}}}(n, r) = \mathsf{c}_{\mathsf{X}}(n, r)$.

Conversion KINT $\Rightarrow$ KIND

In this section, we state the conversion that compiles a KINT-secure access control scheme to a new KIND-secure one and prove its security.

From an access control scheme (Keygen, Derive), we construct a new access control scheme $(\mathsf{Keygen}^{\diamond}, \mathsf{Derive}^{\diamond})$ as follows. Let $H : \mathcal{S} \times \{0,1\}^* \to \{0,1\}^{\mathsf{poly}(\lambda)}$ for some polynomial poly be a hash function.

Construction 3-2. Generic Conversion KINT $\Rightarrow$ KIND

$\mathsf{Keygen}^{\diamond}(1^{\lambda})$**:** Run $\mathsf{Keygen}(1^{\lambda})$ to obtain $\{\langle S, \mathsf{k}(S)\rangle : S \in \mathcal{S}\}$, $\{\langle u, \mathsf{p}(u)\rangle : u \in N\}$, and pub. Define $\mathsf{k}^{\diamond}(S) = H(S, \mathsf{k}(S))$ for all $S \in \mathcal{S}$ and $\mathsf{p}^{\diamond}(u) = \mathsf{p}(u)$ for all $u \in N$. It then outputs $\{\langle S, \mathsf{k}^{\diamond}(S)\rangle : S \in \mathcal{S}\}$, $\{\langle u, \mathsf{p}^{\diamond}(u)\rangle : u \in N\}$, and $\mathsf{pub}^{\diamond} = (\mathsf{pub}, H)$.

$\mathsf{Derive}^{\diamond}(\langle u, \mathsf{p}^{\diamond}(u)\rangle, S, \mathsf{pub}^{\diamond})$**:** It first runs the algorithm $\mathsf{Derive}(\langle u, \mathsf{p}^{\diamond}(u)\rangle, S, \mathsf{pub})$. If $\perp$ is returned, it also outputs $\perp$, else $\mathsf{k}(S)$ is returned. In the latter case, it then outputs $\mathsf{k}^{\diamond}(S) = H(S, \mathsf{k}(S))$.

Theorem 3.8. *Suppose that* (Keygen, Derive) *is KINT-ℓ-secure for some $\ell \in \{1, 2\}$. Then* $(\mathsf{Keygen}^{\diamond}, \mathsf{Derive}^{\diamond})$ *is KIND-ℓ-secure if H is modeled as a random oracle.*

Proof. It is sufficient to prove the case when $\ell = 2$ since the other case follows as a special case. Let q_H be the number of hash queries.

Let $\boldsymbol{A}$ be an adversary which breaks KIND-2-security of $(\mathsf{Keygen}^{\diamond}, \mathsf{Derive}^{\diamond})$ with advantage ϵ. We construct an algorithm $\boldsymbol{B}$ which breaks KINT-2-security of (Keygen, Derive). The construction is as follows.

The adversary $\boldsymbol{A}$ first asks the corrupt query, $\boldsymbol{B}$ just forwards this to its corrupt oracle and then forwards back the result to $\boldsymbol{A}$. This simulation is perfect since $\mathsf{p}(u)^{\diamond} = \mathsf{p}(u)$ for all $u \in N$. At anytime, $\boldsymbol{A}$ may ask a H-query for $(S, R) \in \mathcal{S} \times \{0,1\}^*$. It just returns a random value in the range $\{0,1\}^{\mathsf{poly}(\lambda)}$ for each new query and records the input-output pairs in a H-list which was empty at first. If the query was asked before, it returns the corresponding output that had been recorded in the H-list.

$\boldsymbol{A}$ then eventually outputs S^*. $\boldsymbol{B}$ then collects all the tuples of the form $\langle (S^*, \star), \star\rangle$ appeared in the H-list to a new H^*-list. (Here $\star$ is for indeterminant). Let q^* be the size of the H^*-list at this point and let q' be the number of hash queries so far. Let $\tilde{q} = q^* + q_H - q'$. $\boldsymbol{B}$ randomly chooses $i \in \{1, \ldots, \tilde{q}\}$. If $i \leq q^*$ then let $W^* = W_i$ where $\langle (S^*, R_i), W_i\rangle$ is the i-th tuple in the H^*-list. Otherwise ($i > q^*$) randomly choose $W^* \in \{0,1\}^{\mathsf{poly}(\lambda)}$. In both cases, $\boldsymbol{B}$ then randomly chooses $b \in \{0,1\}$ and returns $\mathsf{RR}(W^*, b)$ to $\boldsymbol{A}$.

$\boldsymbol{A}$ may continue to ask H-queries. For the case $i \leq q^*$, $\boldsymbol{B}$ responds exactly as before. Otherwise ($i > q^*$) $\boldsymbol{B}$ responds as before except the i-th query of the form $(S^*, \star)$, in which case $\boldsymbol{B}$ returns W^*. In both cases, $\boldsymbol{B}$ appends the H^*-list appropriately. At the end, $\boldsymbol{A}$ will output b'.

Finally $\boldsymbol{A}$ outputs (S^*, R_i) to its challenger in the KINT game, i.e., R_i is the guess of $\mathsf{k}(S^*)$.

It is not difficult to see that $\boldsymbol{A}$'s view is identical to its view in the real attack. Let H be the event that $\boldsymbol{A}$ asks a query for $H(S^*, \mathsf{k}(S^*))$ at some point (this implies that at the end of simulation $(\langle S^*, \mathsf{k}(S^*)\rangle, \star)$ appears in the final H^*-list). We claim that $\Pr[\mathsf{H}] \geq 2\epsilon$. This will prove that $\boldsymbol{B}$ guesses $\mathsf{k}(S^*)$ correctly with probability at least $2\epsilon/\tilde{q} \geq 2\epsilon/q_H$.

It remains to prove the claim. If $\boldsymbol{A}$ does not issue a query for $H(S^*, \mathsf{k}(S^*))$ then $\mathsf{k}^\diamond(S^*)$ is independent of $\boldsymbol{A}$'s view. Hence, $\Pr[b = b'|\overline{\mathsf{H}}] = 1/2$. From definition of $\boldsymbol{B}$ we have $|\Pr[b = b'] - 1/2| = \epsilon$. Using these two facts we have the following.

$$\begin{aligned}
\Pr[b = b'] &= \Pr[b = b'|\overline{\mathsf{H}}]\Pr[\overline{\mathsf{H}}] + \Pr[b = b'|\mathsf{H}]\Pr[\mathsf{H}] \\
&\leq \Pr[b = b'|\overline{\mathsf{H}}](1 - \Pr[\mathsf{H}]) + \Pr[\mathsf{H}] \\
&= \frac{1}{2} + \frac{1}{2}\Pr[\mathsf{H}], \\
\Pr[b = b'] &\geq \Pr[b = b'|\overline{\mathsf{H}}]\Pr[\overline{\mathsf{H}}] = \frac{1}{2} - \frac{1}{2}\Pr[\mathsf{H}].
\end{aligned}$$

Hence $\epsilon = |\Pr[b = b'] - 1/2| \leq \frac{1}{2}\Pr[\mathsf{H}]$ as claimed. □

3.2.3 Some Terminology

Viewing Set system as Poset. A set system is partially ordered by the inclusion relation ($\subset$). Interpreting a set system as a partially ordered set (poset) is useful when defining key derivations in AC. Intuitively, Derive algorithm implies that whenever $S_i \subset S_j$, anyone who can access $\mathsf{k}(S_i)$ is allowed to access $\mathsf{k}(S_j)$.

Terminology for Posets, Graphs. The terminology for posets and graphs used in this paper is quite standard one (cf.[D00]) (with some exceptions, see below). Here we review some. A graph is a pair $G = (V, E)$ of sets satisfying $E \subseteq \binom{V}{2}$. V is the set of vertices (or nodes), usually denoted $V(G)$, E is the set of edges, usually denoted $E(G)$. Often, we abuse notation $v \in G$ to mean $v \in V(G)$. A tree is a connected acyclic graph. Let $x = \mathsf{parent}_T(y)$ if x is the parent of a non-root node y in tree T. A directed graph is a pair $G = (V, E)$ of sets satisfying $E \subseteq V \times V$, i.e., an edge is an ordered pair. A directed acyclic graph (DAG) is a directed graph with no directed cycle in it. A notation of chain $x \to y \to z$ means a directed graph which $V = \{x, y, z\}, E = \{(x, y), (y, z)\}$ and is generalized naturally.

An inclusion poset $\mathcal{S}$ can be represented by a DAG G by setting $V = \mathcal{S}$, $E = \{(S, S') : S \subset S';\ S, S' \in \mathcal{S}\}$. This is called the maximal representation, denoted $\mathsf{DAG}_{\mathsf{max}}(\mathcal{S})$. The minimal representation, denoted $\mathsf{DAG}_{\mathsf{min}}(\mathcal{S})$, is the one with $E = \{(S, S') : S \subset_c S';\ S, S' \in \mathcal{S}\}$ where we say $S \subset_c S'$ iff there is no $S'' \in \mathcal{S}$ such that $S \subset S'' \subset S'$.

In our context[1], a graph decomposition (often denoted $\mathcal{G}$) of a poset $\mathcal{S}$ is a family of connected subgraphs whose sets of nodes partition the set of all nodes in the $\mathsf{DAG}_{\mathsf{max}}(\mathcal{S})$. (Thus we sometimes say $\mathcal{G}$ is a graph decomposition of $\mathsf{DAG}_{\mathsf{max}}(\mathcal{S})$). When each subgraph is a tree whose edges are directed away from the root, we call it a tree decomposition (often denoted $\mathcal{T}$). When each graph is a directed chain whose edges are directed in the same direction, we call it a chain decomposition (often denoted $\mathcal{C}$). An induced graph decomposition is one in which each subgraph is an induced subgraph.

[1]Our notions for tree and chain decompositions are *not* standard ones (cf.[D00]). Instead the notions introduced here could have been named as *tree cover* and *path cover*, resp.

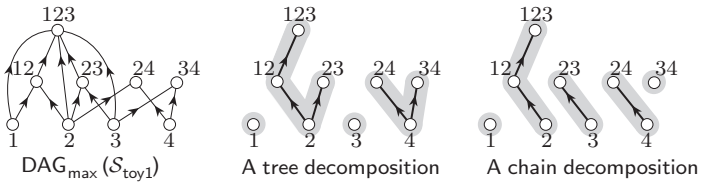


Figure 3-1: Toy example 1 and its graph decompositions

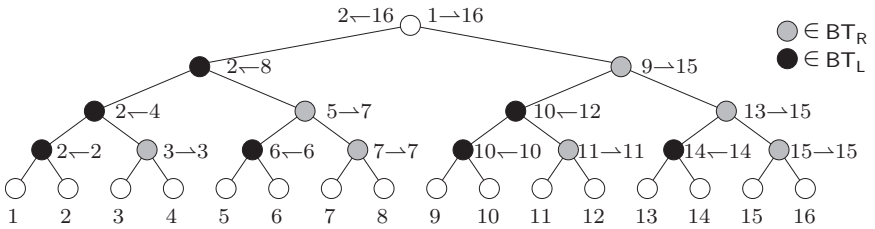


Figure 3-2: Set system SIC defined by the union of all the collections written at each node

Fig.3-1 shows graph decompositions of the set system for toy example 1, $\mathcal{S}_{\mathsf{toy1}} = \{\{1\},\{2\},\{3\},\{4\},\{1,2\},\ \{2,3\},\ \{2,4\},\{3,4\},\{1,2,3\}\}$. From now we abuse some notations, often in figures, e.g., writing 12 or $1,2$ instead of $\{1,2\}$ if it causes no confusion. Note that every chain decomposition is a tree decomposition.

We will fix BT to be the complete binary tree of n leaves labeled $1,...,n$ from left to right. The level of node in BT is the distance from root to it. For a fixed node, its left (resp., right) nodes are those nodes with the same level and appear on the left (resp., right). BT will be used only to help defining set systems and should not be confused with the graph representations of posets of set systems.

3.3 New Set Systems

3.3.1 Subset Incremental Chain (SIC) Set System

The SIC Set System. For $i,j \in N = \{1,...,n\}$ and $i < j$, denote

$$\begin{aligned} i \rightharpoonup j &:= \{\{i\},\{i,i+1\},\ldots,\{i,\ldots,j\}\},\\ i \leftharpoonup j &:= \{\{j\},\{j,j-1\},\ldots,\{j,\ldots,i\}\}, \end{aligned}$$

and $(i \rightharpoonup i) = (i \leftharpoonup i) := \{\{i\}\}$. Consider the binary tree BT. For a node v in BT, let l_v (resp., r_v) be the leftmost (resp., rightmost) leaf under v. We define the set system SIC (of n users) by letting

$$\mathcal{S}_{\mathsf{SIC}} = \bigcup_{v\in \mathsf{BT_L}} (l_v+1 \leftharpoonup r_v) \cup \bigcup_{v\in \mathsf{BT_R}} (l_v \rightharpoonup r_v-1) \cup (1 \rightharpoonup n) \cup (2 \leftharpoonup n), \tag{3.1}$$

where $\mathsf{BT_L}$ (resp., $\mathsf{BT_R}$) are the set of internal nodes which are left (resp., right) children. An informal visual view of $\mathcal{S}_{\mathsf{SIC}}$ is shown in Fig.1, where the union of all the collections written there is the only important information.

Theorem 3.9. *$\mathcal{S}_{\mathsf{SIC}}$ is $(2r)$-complement-cover set system.*

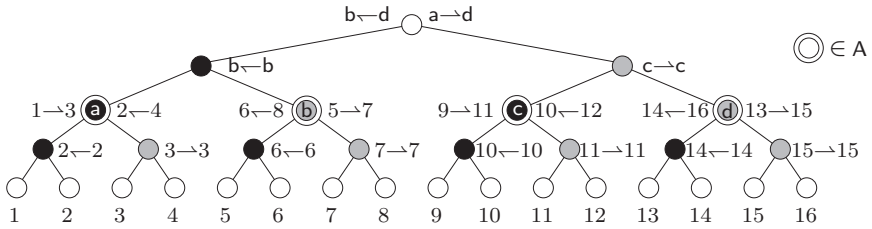


Figure 3-3: Set system $\mathsf{LSIC}[k]$, $k = 2$, as the union of all collections written at each node

Proof. We call a set of the form $\{i, i+1, \ldots, j\}$ for some $i \leq j$ a consecutive set. We first claim that any consecutive set, say $A = \{i, \ldots, j\}$, can be partitioned to no more than 2 sets in $\mathcal{S}_{\mathsf{SIC}}$; then prove it as follows. Let a be the least common ancestor node of the leaves i and j in BT, denoted $\mathsf{lca}(i, j) = a$. Let s be the least ancestor of a which is in BT_{L} if $a \in \mathsf{BT}_{\mathsf{R}}$ and which is in BT_{R} if $a \in \mathsf{BT}_{\mathsf{L}}$. Let x, y be the left and right children of a. First if $i = 1$ then $A \in (1 \rightharpoonup n) \subseteq \mathcal{S}_{\mathsf{SIC}}$; else if $j = n$ then $A \in (2 \leftharpoonup n) \subseteq \mathcal{S}_{\mathsf{SIC}}$ (since $2 \leq i$). Now assume $i \neq 1$, $j \neq n$. We list all possible cases of (i, j) as follows. Let $*$ be an unspecified value.

1. If $(i = l_a; j = *; a \in \mathsf{BT}_{\mathsf{L}})$ then $A \in (l_s \rightharpoonup r_s - 1) \subseteq \mathcal{S}_{\mathsf{SIC}}$ (since $i = l_s$; $j < r_s - 1$; and $s \in \mathsf{BT}_{\mathsf{R}}$),
2. If $(i = *; j = r_a; a \in \mathsf{BT}_{\mathsf{R}})$ then $A \in (l_s + 1 \leftharpoonup r_s) \subseteq \mathcal{S}_{\mathsf{SIC}}$ (since $j = r_s$; $l_s + 1 < i$; and $s \in \mathsf{BT}_{\mathsf{L}}$),
3. If $(i = l_a; j \neq r_a; a \in \mathsf{BT}_{\mathsf{R}})$ then $A \in (l_a \rightharpoonup r_a - 1) \subseteq \mathcal{S}_{\mathsf{SIC}}$ (since $j \leq r_a - 1$),
4. If $(i \neq l_a; j = r_a; a \in \mathsf{BT}_{\mathsf{L}})$ then $A \in (l_a + 1 \leftharpoonup r_a) \subseteq \mathcal{S}_{\mathsf{SIC}}$ (since $l_a + 1 \leq i$),
5. If $(i \neq l_a; j \neq r_a; a \in *)$ then $A = P \cup Q$; $P = \{i, \ldots, r_x\}$, $Q = \{l_y, \ldots, j\}$, and we have $P, Q \in \mathcal{S}_{\mathsf{SIC}}$ (since
   - $\mathsf{lca}(i, r_x) = x$, thus (i, r_x) will fall to the case 4 and $P \in \mathcal{S}_{\mathsf{SIC}}$;
   - $\mathsf{lca}(l_y, j) = y$, thus (l_y, j) will fall to the case 3 thus $Q \in \mathcal{S}_{\mathsf{SIC}}$).

These proved the claim. Now we are back to the proof, it is obvious that $N \setminus R$ can be partitioned to no more than r consecutive sets if 1 or $n \in R$; or to no more than $r + 1$ such sets otherwise. In the former case, the partition size to sets in $\mathcal{S}_{\mathsf{SIC}}$ is $\leq 2r$; while in the latter case (where $\{1, ..., s\}$ and $\{t, ..., n\}$ for some s, t are included in the partition), it is $\leq 1(1) + 2(r - 1) + 1(1) = 2r$. □

Intuitively, SIC has graph decompositions with good properties since each collection in the union of Eq.(3.1) forms a chain of subset. This will become clearer in the next section. The set system LSIC below generalizes SIC.

3.3.2 Layered SIC (LSIC) Set Systems

The $\mathsf{LSIC}[k]$ Set System. We view BT consisting of subtrees (also binary and complete) of $n^{1/k}$ leaves so that there are exactly k layers of such subtrees, where $k | \log n$. We will call such subtree an "atomic" subtree (to distinguish from other kinds of subtrees in BT). Informally, each atomic subtree contributes sets to $\mathcal{S}_{\mathsf{LSIC}}$ as in the SIC set system for that

subtree, albeit each leaf in the subtree represents all the leaves under it in BT. More formally, for node z in BT, let $A_z := \{l_z, l_z + 1, ..., r_z\}$ (i.e., all the leaves under z). In particular, if z is a leaf then $A_z = \{z\} = \{l_z\} = \{r_z\}$. Let us consider the leaves u, v in an atomic subtree where v is some node on the right of u. We denote $u^{(+1)}, u^{(+2)}$ (and so on) be the next one, two (and so on) *right* leaves to u in that atomic subtree. Denote $u^{(-1)}, u^{(-2)}$ analogously. Denote

$$\begin{aligned} u \rightharpoonup v &:= \{A_u, A_u \cup A_{u^{(+1)}}, \ldots, A_u \cup \cdots \cup A_v\}, \\ u \leftharpoonup v &:= \{A_v, A_v \cup A_{v^{(-1)}}, \ldots, A_v \cup \cdots \cup A_u\}. \end{aligned}$$

Let l'_w, r'_w be the leftmost and rightmost leaves under w *in the atomic subtree* and not w itself; for example, $l'_{\mathsf{root}} = a, r'_{\mathsf{root}} = d$ and $l'_a = 1, r'_a = 4$ in Fig.3. Let A be the set of all nodes which are the roots of atomic subtrees but excluding the root of BT. We define $\mathsf{LSIC}[k]$ analogously to Eq.(3.1) by letting

$$\mathcal{S}_{\mathsf{LSIC}[k]} = \bigcup_{v \in \mathsf{BT_L} \cup \mathsf{A}} ({l'_v}^{(+1)} \leftharpoonup r'_v) \cup \bigcup_{v \in \mathsf{BT_R} \cup \mathsf{A}} (l'_v \rightharpoonup {r'_v}^{(-1)}) \cup (l'_{\mathsf{root}} \rightharpoonup r'_{\mathsf{root}}) \cup (l'^{(+1)}_{\mathsf{root}} \leftharpoonup r'_{\mathsf{root}}). \quad (3.2)$$

Observe that each $v \in \mathsf{A}$ has two collections $({l'_v}^{(+1)} \leftharpoonup r'_v), (l'_v \rightharpoonup {r'_v}^{(-1)})$ attached since intuitively it is a kind of "root" (i.e., of an atomic subtree) and at a root, the structure of SIC introduces two collections, one for left and one for right direction (cf. Eq.(3.1) and Fig.1).

Theorem 3.10. *$\mathcal{S}_{\mathsf{LSIC}[k]}$ is $(2kr)$-complement-cover set system for a constant k ; and $\mathcal{S}_{\mathsf{LSIC}[\log_a n]}$ is $O(r \log_a(n/r) + r)$-complement-cover set system for a constant a.*

Note that when $k = \log_a n$, from the former claim we already have that $\mathcal{S}_{\mathsf{LSIC}[\log_a n]}$ is $(2r \log_a n)$-complement-cover, but the claim above gives a sharper bound.

Proof. First we will prove that $\mathcal{S}_{\mathsf{LSIC}[k]}$ is $(2kr)$-complement-cover. Let ST_R denote the Steiner tree of a set of leaves $R \subseteq N$, i.e., the subtree of BT that consists of all paths from the root to each leaf in R. We call a node v special if $v \in \mathsf{A}$ or v is a leaf in BT. We "color" a node if it is special but is not in ST_R and all of its special ancestors are in ST_R. Denote C the set of all color nodes. Hence $N \setminus R = \bigcup_{v \in \mathsf{C}} A_v = \bigcup_{j=1}^{k} \bigcup_{v \in L_j \cap \mathsf{C}} A_v$ where we denote L_j to be the set of all special nodes in the j-th special layer away from root (i.e., at distance $j(\log n)/k$ from the root). It suffices to prove that for each special layer j, the set $Y_j := \bigcup_{v \in L_j \cap \mathsf{C}} A_v$ can be partitioned to at most $2r$ sets in the family $\mathcal{S}_{\mathsf{LSIC}}$. Denote x_i to be the number of uncolored special nodes in the i-th atomic subtrees from left to right in this j-th layer. From Theorem 3.9, it is easy to deduce that Y_j can be partitioned to at most $2(x_1 + x_2 + \cdots + x_p)$ sets in $\mathcal{S}_{\mathsf{LSIC}}$, where p is the last atomic subtree in this layer (in fact, $p = n^{(j-1)/k}$). But we have $x_1 + \cdots + x_p \leq r$ since the Steiner tree of r leaves passes through all these uncolored special nodes. This proves the claim.

Next we will prove that $\mathcal{S}_{\mathsf{LSIC}[\log_a n]}$ is $O(r \log_a(n/r) + r)$-complement-cover. We first give the definition of Stratified Subset-Difference set system with each atomic subtree of a leaves (SSD_a): $\mathcal{S}_{\mathsf{SSD}_a} = \{A_u \setminus A_v : \ u \text{ is an ancestor of } v \text{ in the same atomic subtree}\}$. (Note that the name SSD is given in [GST04], but is independently presented in [GR04]). It is known [GR04] that $\mathcal{S}_{\mathsf{SSD}_a}$ is $(O(r \log_a(n/r) + r))$-complement-cover. Next, we claim

that each $A_u \setminus A_v$ can be partitioned to at most 2 sets in $\mathcal{S}_{\mathsf{LSIC}[\log_a n]}$. Before proving this, we conclude that $\mathsf{LSIC}[\log_a n]$ has $\mathsf{c}_{\mathsf{LSIC}[\log_a n]}(n,r) = 2\mathsf{c}_{\mathsf{SSD}_a}(n,r) = O(r\log_a(n/r)+r)$.

It is left to prove the claim. Let u be an ancestor of v in the same atomic subtree. Thus $l_u \le l_v \le r_v \le r_u$ (where the first and the third equalities will not occur simultaneously). If $l_u < l_v \le r_v < l_v$, then $A_u \setminus A_v = \{l_u, l_u+1, \ldots, l_v-1\} \cup \{r_v+1, r_v+2, \ldots, r_u\}$. If $l_u = l_v \le r_v < l_v$, then only the last term in union appears. If $l_u < l_v \le r_v = l_v$, then only the first term in union appears. From the fact that the two terms in the union are in $(l_u \rightharpoonup r_u)$ and $(l_u \leftharpoonup r_u)$ respectively, we thus completes the proof of the claim. □

3.4 Key Derivation based on PRSG

3.4.1 PRSG based Framework

Framework Idea. In this framework, we use pseudo-random sequence generators to derive keys from one subset to another. The correctness of access control schemes allows this to be done only if the first set is included in the latter (e.g.,$\{1\} \subset \{1,2\}$). Thus such derivations can be defined in correspondence with directed edges in a graph decomposition of $\mathsf{DAG}_{\mathsf{max}}(\mathcal{S})$, in which all the inclusion relations in $\mathcal{S}$ are included. One exception is that there should be no node with $\mathsf{indegree} > 1$ in any graph in the decomposition since it would imply a collision of PRSG, which should be computable by neither broadcasters nor adversaries. Therefore, all the valid decompositions are *tree decompositions*, of which the class includes all graph decompositions of the poset that allow $\mathsf{indegree} \le 1$ for all nodes. Each user then stores keys for subsets which he is in and are closest to the root of that tree. For the toy example 1 in Fig.3-1, our paradigm with the tree decomposition in the figure namely $\mathcal{T}_{\mathsf{toy1}}$ allows the user 2 to store only the keys at $2, 24$.

Note that in order to be provably secure in the KIND sense, it is mandatory to make an adaptation so that keys are not derived from another key *directly*. Instead, one should use intermediate keys denoted $\mathsf{t}(S)$ for $S \in \mathcal{S}$; how to use this is explained in the construction. This was neglected in many recent schemes that use similar one-way derivation approaches.

We now give the formal description. Such a scheme is based solely on a tree decomposition, say $\mathcal{T}$, of the poset $\mathcal{S}_{\mathsf{X}}$. The scheme applies to an arbitrary complement-cover set system X.

Construction 3-3. PRSG-based Access Control Scheme: $(\mathsf{X})^{\mathsf{prsg}}$

Keygen : ($\mathsf{Subset\ keys}$) At the root S of a tree in $\mathcal{T}$, let $\mathsf{t}(S) \leftarrow \{0,1\}^\lambda$. For each node S (either root or non-root of a tree in $\mathcal{T}$) whose all children are $S_{i_1}, ..., S_{i_d}$ where d is the outdegree of S, we define the following recurrence relation:

$$\mathsf{t}(S_{i_1}) \| \cdots \| \mathsf{t}(S_{i_d}) \| \mathsf{k}(S) \leftarrow \mathsf{PRSG}_{d+1}(\mathsf{t}(S)), \tag{3.3}$$

where $|\mathsf{t}(S_{i_1})| = \cdots = |\mathsf{t}(S_{i_d})| = |\mathsf{k}(S)| = \lambda$ bits; $\mathsf{PRSG}_j : \{0,1\}^\lambda \to \{0,1\}^{j\lambda}$.

($\mathsf{User\ keys}$) For a tree $G \in \mathcal{T}$, let $\mathsf{parent}_G(S)$ be the parent of S in G if S is not the root of G or be $\emptyset$ otherwise.

For $u \in N$, we define its key as

$$\mathsf{p}(u) = \{\mathsf{t}(S) | u \in S; u \notin \mathsf{parent}_G(S), G \in \mathcal{T}\}.$$

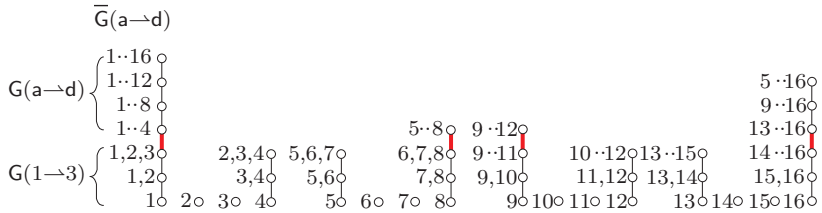


Figure 3-4: The tree decomposition $\mathcal{T}_{\mathsf{LSIC}[k]}$ of the set system $\mathsf{LSIC}[k]$ (see Fig.3). A simpler decomposition $\mathcal{T}'_{\mathsf{LSIC}[k]}$ is the one without the thick red edges.

Derive : Find the tree where S is in and then use Eq.(3.3) to derive $\mathsf{k}(S)$.

Characterizing Efficiency. Let $\mathsf{RS}_{\mathcal{T}}(u) = \{S \mid u \in S; u \notin \mathsf{parent}_G(S), G \in \mathcal{T}\}$ and $\mathsf{RN}_{\mathcal{T}}(u) = |\mathsf{RS}_{\mathcal{T}}(u)|$ and call them the *reachability set* and *reachability number* of u in $\mathcal{T}$ (since it is the minimal set of sufficient nodes such that when traversing from these nodes in the edge direction we meet all $S \in \mathcal{S}$ such that $u \in S$). Let $\mathsf{DD}_{\mathcal{T}}$ = the depth of the deepest trees. We have

$$\mathsf{KeySize}_{(\mathsf{X})^{\mathsf{prsg}}}(u) = \mathsf{RN}_{\mathcal{T}}(u), \quad \mathsf{CompCost}_{(\mathsf{X})^{\mathsf{prsg}}} = \mathsf{DD}_{\mathcal{T}}. \tag{3.4}$$

Theorem 3.11. ([AKI03a]) $(\mathsf{X})^{\mathsf{prsg}}$ *is secure in the sense of KIND assuming secure* PRSG.

3.4.2 PRSG based Instantiation for SIC, LSIC

First PRSG-based Instantiation for SIC. It suffices to define a tree decomposition of $\mathcal{S}_{\mathsf{SIC}}$ and the concrete scheme will follow automatically from the general construction of the framework. We choose the following natural one and prove that it is the optimal decomposition for SIC. For $i \le j \in N$, define graphs

$$\mathsf{G}(i \rightharpoonup j) = \{i\} \to \{i, i+1\} \to \cdots \to \{i, ..., j\};$$
$$\mathsf{G}(i \leftharpoonup j) = \{j\} \to \{j, j-1\} \to \cdots \to \{j, ..., i\}.$$

We will use the following tree decomposition.

$$\mathcal{T}_{\mathsf{SIC}} = \{\mathsf{G}(l_v + 1 \leftharpoonup r_v) | v \in \mathsf{BT_L}\} \cup \{\mathsf{G}(l_v \rightharpoonup r_v - 1) | v \in \mathsf{BT_R}\} \cup \{\mathsf{G}(1 \rightharpoonup n), \mathsf{G}(2 \leftharpoonup n)\} \tag{3.5}$$

Let $\langle x \rangle$ denotes the binary representation of x. We have the following theorem.

Theorem 3.12. *The tree decomposition* $\mathcal{T}_{\mathsf{SIC}}$ *yields minimal* $\max_{u \in N} \mathsf{RN}_{\mathcal{T}}(u)$*, indeed we have*

$$\mathsf{RN}_{\mathcal{T}_{\mathsf{SIC}}}(u) = \begin{cases} \log n + 2 - f(\langle u - 1 \rangle) & ; 2 \le u \le n \\ 1 & ; u = 1, \end{cases}$$

where $f(y) :=$ *the number of the same consecutive least significant bits of* y*. In particular,* $\max_{u \in N} \mathsf{RN}_{\mathcal{T}_{\mathsf{SIC}}}(u) = \log n + 1$*. We also have* $\mathsf{DD}_{\mathcal{T}_{\mathsf{SIC}}} = n - 1$*.*

Proof. We define $F_v = l_v + 1 \leftharpoonup r_v$ if $v \in \mathsf{BT_L}$ and $l_v \rightharpoonup r_v - 1$ if $v \in \mathsf{BT_R}$. $\mathcal{T}_{\mathsf{SIC}}$ is really a tree decomposition since $\{F_v : v \in \mathsf{BT_L} \cup \mathsf{BT_R}\} \cup \{(1 \rightharpoonup n), (2 \leftharpoonup n)\}$ can be proved to be a pairwise non-intersecting family (somewhat straightforwardly). Next we prove the

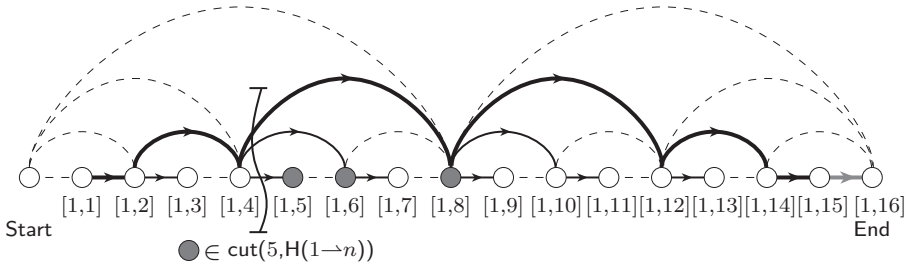


Figure 3-5: Showing $\mathsf{H}(1 \rightharpoonup n)$ when $n = 16, k = 4$ and also showing $\mathsf{cut}(5, \mathsf{H}(1 \rightharpoonup 16))$.

formula for $\mathsf{RN}_{\mathcal{T}_{\mathsf{SIC}}}(u)$. For $u \in N \setminus \{1\}$, only possible trees in $\mathcal{T}_{\mathsf{SIC}}$ that u appears are those graphs $\mathsf{G}(F_v)$ for internal nodes v on the path from the leaf u to the root in BT, and $\mathsf{G}(1 \rightharpoonup n), \mathsf{G}(2 \leftharpoonup n)$. Each graph $\mathsf{G}(\cdot)$ that u appears contribute one key for u. Thus $\mathsf{RN}_{\mathcal{T}_{\mathsf{SIC}}}(u)$ is at most $(\log n - 1) + 2$. Let $u, w_1, ..., w_{\log n}$, root be the nodes on that path. Due to symmetry, we assume w.l.o.g. that $w_1, ..., w_{z-1} \in \mathsf{BT_L}$ and $w_z \in \mathsf{BT_R}$. Now it is easy to see that

1. For $1 \leq j \leq z-1$, we have $\mathsf{G}(F_{w_j}) = \mathsf{G}(l_{w_j} + 1 \leftharpoonup r_{w_j})$ thus u does not appear in these graphs.

2. For $j = z$, we have $\mathsf{G}(F_{w_j}) = \mathsf{G}(l_{w_z} \rightharpoonup r_{w_z} - 1)$ thus u appears in these graph.

3. For $z < j \leq \log n$, we have that u appears in $\mathsf{G}(F_{w_j})$ since $l_{w_j} < u < r_{w_j}$.

We also have that $z = f(\langle u - 1 \rangle)$. Thus $\mathsf{RN}_{\mathcal{T}_{\mathsf{SIC}}}(u) = (\log n - 1) + 2 - (f(\langle u - 1 \rangle) - 1)$ as desired. Now we prove that $\mathcal{T}_{\mathsf{SIC}}$ is optimal (obtaining minimal $(\max_{u \in N} \mathsf{RN}_{\mathcal{T}}(u))$ among all $\mathcal{T}$ of SIC). Observe that for all $\mathcal{T}$ of SIC, $\sum_{u \in N} \mathsf{RN}_{\mathcal{T}}(u) = \sum_{S \in \mathcal{S}_{\mathsf{SIC}}} |\{u : u \in S, u \notin \mathsf{parent}_G(S), G \in \mathcal{T}\}| \geq |\mathcal{S}_{\mathsf{SIC}}| = n \log n + 1$. Hence $\max_{u \in N} \mathsf{RN}_{\mathcal{T}}(u) \geq \lceil \frac{n \log n + 1}{n} \rceil = \log n + 1$. Our decomposition matches this bound. □

The number of keys at each user is not uniform as recorded in the corollary below. While sharing some similarities with our scheme, the basic schemes in [GST04, WNR04] assign one-way chains in both left and right directions at each node in BT while we use only one direction and exploit some symmetries. This can be an intuition as to why we can reduce key size at least 2 times (and up to $\log n$ in the best case, user 1). Those schemes can be considered as instantiations in our framework, but with storage-redundancies in the sense that the set systems extracted from their schemes are sets with repetition. Moreover, the scheme of [GST04] can also be shown to be derivation-redundant when exposed in our framework.

Corollary 3.13. *In the scheme* $(\mathsf{SIC})^{\mathsf{prsg}}$*, there are exactly* 2^x *users who store exactly* $x + 2$ *keys for* $0 \leq x \leq (\log n) - 1$ *and exactly 1 user who stores 1 key.*

Second PRSG-based Instantiation for SIC. We now give an instantiation of the set system SIC in the PRSG-based framework but this time is done via a new tree decomposition $\mathcal{T}_{\mathsf{SIC,new}}$ which is defined exactly the same as in Eq.(3.5) except changing G to H, where we define H as follows. $\mathcal{T}_{\mathsf{SIC,new}}$ will depend on k which we assume that $k | \log n$ for simplicity.

We first define $\mathsf{H}(l_v \rightharpoonup r_v - 1)$ for $v \in \mathsf{BT_R} \cup \{\mathsf{root}\}$. Let the node set $V(\mathsf{H}(l_v \rightharpoonup r_v - 1)) = (l_v \rightharpoonup r_v - 1)$ and define the edge set $E(\mathsf{H}(l_v \rightharpoonup r_v - 1))$ as follows.

1. Write the elements of $(l_v \rightharpoonup r_v - 1)$ from left to right on a horizontal line in the increasing order of inclusion. In addition, put two dummy nodes Start at the leftmost and End at the rightmost. The length of the whole line will be $L_v := r_v - l_v + 1$. Let x be the integer such that $n^{(x-1)/k} < L_v \leq n^{x/k}$. Note that $1 \leq x \leq k$.

2. For $0 \leq i \leq x-1$ do the following. Start with Start, jump consecutively to its next node that is at distance $n^{i/k}$ away from it until either it meets End or the next jump would exceed End. For every jump, write the corresponding directed edge.

3. Remove all the edges of the form $(\mathsf{Start}, *)$ and $(*, \mathsf{End})$.

4. Remove all the edges (S, T) if there is (S', T) where the distance between nodes S and T is shorter than nodes S' and T.

Define $\mathsf{H}(l_v + 1 \leftharpoondown r_v)$ for $v \in \mathsf{BT_L} \cup \{\mathsf{root}\}$ analogously. Finally, define $\mathsf{H}(1 \rightharpoonup n)$ by letting $E(\mathsf{H}(1 \rightharpoonup n)) = E(\mathsf{H}(1 \rightharpoonup n-1)) \cup \{([1, n-1], [1, n])\}$, where we let $[i, j] := \{i, ..., j\}$.

Fig. 3-5 shows $\mathsf{H}(1 \rightharpoonup n)$ for $n = 16, k = 4$. The non-dotted lines comprise the tree. The dotted lines are those edges which have been removed in the step 3 and 4 of the construction. The gray line is edge $([1, n-1], [1, n])$.

Theorem 3.14. *The tree decomposition $\mathcal{T}_{\mathsf{SIC,new}}$ yields*

$$\begin{aligned} \max_{u \in N} \mathsf{RN}_{\mathcal{T}_{\mathsf{SIC,new}}}(u) &\leq k(\log n + 1) = O(k \log n) \\ \mathsf{DD}_{\mathcal{T}_{\mathsf{SIC,new}}} &= (2k-1)(n^{1/k} - 1) = O(kn^{1/k}). \end{aligned}$$

Proof. For $l_v < u < r_v$ let $\mathsf{cut}(u, \mathsf{H}(l_v \rightharpoonup r_v - 1))$ be the set of all edges crossing the vertical line between $[l_v, u-1]$ and $[l_v, u]$ in $\mathsf{H}(l_v \rightharpoonup r_v - 1)$ and let $\mathsf{cut}(u, \mathsf{H}(l_v + 1 \leftharpoondown r_v))$ be the set of all edges crossing the vertical line between $[u, r_v]$ and $[u-1, r_v]$ in $\mathsf{H}(l_v + 1 \leftharpoondown r_v)$. Define $F_v = l_v + 1 \leftharpoondown r_v$ if $v \in \mathsf{BT_L}$ and $l_v \rightharpoonup r_v - 1$ if $v \in \mathsf{BT_R}$.

It is not hard to see that $\mathsf{RS}_{\mathcal{T}_{\mathsf{SIC,new}}}(u)$ is the set of all the terminated nodes of edges in $\bigcup_{v \in \mathsf{path}_u} \mathsf{cut}(u, \mathsf{H}(F_v)) \cup \mathsf{cut}(u, \mathsf{H}(1 \rightharpoonup n)) \cup \mathsf{cut}(u, \mathsf{H}(2 \leftharpoondown n))$; where path_u is the set of all internal nodes on the path from u to root in BT. Also observe that $|\mathsf{cut}(u, \mathsf{H}(F_v))| \leq x$ where $n^{(x-1)/k} < L_v \leq n^{x/k}$ since there are x iterations in the step 2 of the construction. Also $|\mathsf{cut}(u, \mathsf{H}(1 \rightharpoonup n))|, |\mathsf{cut}(u, \mathsf{H}(2 \leftharpoondown n))| \leq k$. Hence we have

$$\mathsf{RN}_{\mathcal{T}_{\mathsf{SIC,new}}}(u) \leq \sum_{x=1}^{k} \left(\sum_{v: \left\{ \begin{smallmatrix} v \in \mathsf{path}_u \\ n^{(x-1)/k} < L_v \leq n^{x/k} \end{smallmatrix} \right\}} x \right) + 2k$$

Simplifying this we have $\mathsf{RN}_{\mathcal{T}_{\mathsf{SIC,new}}}(u) \leq \sum_{x=1}^{k-1} x(\frac{\log n}{k}) + k(\frac{\log n}{k} - 1) + 2k = \frac{(k+1)}{2} \log n + k$ and the result follows.

Now we prove $\mathsf{DD}_{\mathcal{T}_{\mathsf{SIC,new}}}$. By inspection, the longest path is in the tree $\mathsf{H}(1 \rightharpoonup n)$. Fix $t = (n^{1/k} - 1)$. Denoted by $\mathsf{J}(a, b)$ the action of taking a times of b-length jumps. This path starts from $\{1\}$ and takes the following sequence of jumps:

$$\mathsf{J}(t, 1), \mathsf{J}(t, n^{1/k}), ..., \mathsf{J}(t, n^{(k-2)/k}), \mathsf{J}(t-1, n^{(k-1)/k}),$$
$$, \mathsf{J}(t, n^{(k-2)/k}), ..., \mathsf{J}(t, n^{1/k}), \mathsf{J}(t+1, 1).$$

This is shown by the thick lines in Fig. 3-5. Hence it takes $2(k-1)(n^{1/k}-1)+n^{1/k}-2+1 = (2k-1)(n^{1/k}-1)$ jumps, this completes the proof. □

PRSG-based Instantiation for LSIC. Before describing our default tree decomposition of $\mathcal{S}_{\mathsf{LSIC}}$, denoted $\mathcal{T}_{\mathsf{LSIC}[k]}$, we first describe a more straightforward one, denoted $\mathcal{T}'_{\mathsf{LSIC}[k]}$, which is constructed, informally, as the union of all $\mathcal{T}_{\mathsf{SIC}}$ applied to each atomic subtree in BT. More formally, we can define $\mathsf{G}(u \rightharpoonup v)$ for u, v which are leaves in the same atomic subtree, analogously as before, by letting $\mathsf{G}(u \rightharpoonup v) = A_u \to A_u \cup A_{u^{(+1)}} \to \cdots \to (A_u \cup \cdots \cup A_v)$, and analogously for $\mathsf{G}(u \leftharpoonup v)$. Without going into details, we can define $\mathcal{T}'_{\mathsf{LSIC}[k]}$ from Eq.(3.2) in an analogous way when we defined $\mathcal{T}_{\mathsf{SIC}}$ in Eq.(3.5) from Eq.(3.1).

Now $\mathcal{T}_{\mathsf{LSIC}[k]}$ is constructed by an observation that $\mathsf{G}(l'_v \rightharpoonup r'^{(-1)}_v)$ and $\mathsf{G}(v \rightharpoonup *)$ can be combined into one chain (and in particular, one tree) since the maximum element in the former, $A_{l'_v} \cup \cdots \cup A_{r'^{(-1)}_v}$, is included in A_v, the minimum element of the latter. For $v \in \mathsf{BT_R} \cup \{\mathsf{root}\}$, let $w_1, ..., w_m$ be the sequence of nodes in $\mathsf{BT_L} \cap \mathsf{A}$ such that $w_1 = l'_v$; for $1 \le i \le m-1$, $w_{i+1} = l'_{w_i}$; and $l_v = l'_{w_m}$, then define $\bar{\mathsf{G}}(l'_v \rightharpoonup x) := \mathsf{G}(l'_{w_m} \rightharpoonup r'^{(-1)}_{w_m}) \to \cdots \to \mathsf{G}(l'_{w_1} \rightharpoonup r'^{(-1)}_{w_1}) \to \mathsf{G}(l'_v \rightharpoonup x)$ where x is some right node of l'_v. (Here, '$\to$' means to connect the chains). The definition for $\bar{\mathsf{G}}(x \leftharpoonup r'_v)$ for $v \in \mathsf{BT_L} \cup \{\mathsf{root}\}$ can be done analogously. Now we define

$$\begin{aligned}\mathcal{T}_{\mathsf{LSIC}[k]} = \{\bar{\mathsf{G}}(l'^{(+1)}_v \leftharpoonup r'_v) | v \in \mathsf{BT_L}\} \\ \cup \{\bar{\mathsf{G}}(l'_v \rightharpoonup r'^{(-1)}_v) | v \in \mathsf{BT_R}\} \\ \cup \{\bar{\mathsf{G}}(l'_{\mathsf{root}} \rightharpoonup r'_{\mathsf{root}}), \bar{\mathsf{G}}(l'^{(+1)}_{\mathsf{root}} \rightharpoonup r'_{\mathsf{root}})\}. \quad (3.6)\end{aligned}$$

The abstraction of this decomposition may disguise the simplicity of the scheme; in Fig.3-4 we thus give an explicit example when $n = 16$ and $k = 2$ (cf. Fig.3).

The following theorem and corollary can be proved by an elementary counting argument based on Theorem 3.12.

Theorem 3.15. *The tree decomposition $\mathcal{T}_{\mathsf{LSIC}[k]}$ yields*

$$\mathsf{RN}_{\mathcal{T}_{\mathsf{LSIC}[k]}}(u) = \log n + 1 + k - g_k(\langle u-1 \rangle)$$

where $g_k(\langle x \rangle) := f(0||\langle x_1 \rangle) + f(b_1||\langle x_2 \rangle) \cdots + f(b_{k-1}||\langle x_k \rangle)$ where we parse $\langle x \rangle$, with padding of 0s on the left so to have length $\log n$ bits, as $\langle x_1 \rangle || \cdots || \langle x_k \rangle$ so that each $\langle x_i \rangle$ has length $(\log n)/k$ bits; b_j is the least significant bit of $\langle x_j \rangle$. In particular, $\max_{u \in N} \mathsf{RN}_{\mathcal{T}_{\mathsf{LSIC}[k]}}(u) = \log n + 1$. We also have $\mathsf{DD}_{\mathcal{T}_{\mathsf{LSIC}[k]}} = kn^{1/k}$.

As an example, user 4 will store 2 keys: $\mathsf{k}(1234), \mathsf{k}(4)$ (see Fig.3-4). This can be calculated as $|\mathsf{p}(4)| = 4 + 1 + 2 - (f(0||00) + f(0||11)) = 2$ (Note $\langle 4-1 \rangle = 0011$).

Corollary 3.16. *In $(\mathsf{LSIC}[\mathsf{k}])^{\mathsf{prsg}}$, exactly $\sum_{j=0}^{x-1} \binom{k}{j} C(x-1, j, (\log n)/k) 2^{x-1-j}$ users store exactly x keys for $2 \le x \le (\log n) + 1$ and exactly 1 user stores 1 key where $C(a, b, c)$ is the number of integer compositions (ordered partitions) of a into b positive integers, each $\le c$.*[2]

[2]For example $C(5, 3, 2) = 3$ since $5 = 1+2+2 = 2+1+2 = 2+2+1$. The exact formula of $C(a, b, c)$ is quite complicated and is shown in [S76].

3.5 Key Derivation based on Non-Trapdoor RSA

3.5.1 Non-Trapdoor RSA based Framework

Framework Idea. We first briefly review the access control scheme of Akl-Taylor [AT83]. There, each $S \in \mathcal{S}$ is assigned a publicly known prime. The key of S is defined as $\mathsf{k}(S) = s^{\prod_{T:S \not\rightarrow T} p_T}$ modulo an RSA modulus, where s is a secret; and $S \not\rightarrow T$ means (S,T) is not an edge in $\mathsf{DAG}_{\mathsf{max}}(\mathcal{S})$. Each user u just stores $\mathsf{k}(\{u\})$. The terms in the exponents are arranged so that even any collusion cannot compute keys that are not supposed to be computable by them. However, the number of primes used in the above schemes are too large as $|\mathcal{S}|$. Such primes will be stored as non-secret storage or derived on-the-fly.[2] We propose a new paradigm which makes uses of *prime powers* so that the number of primes used becomes optimal. We will see shortly that assigning prime powers depends essentially on a *chain decomposition* of $\mathsf{DAG}_{\mathsf{max}}(\mathcal{S})$. Indeed, the number of primes used will be exactly the number of chains; and each node in the same chain will correspond to the same prime but with a distinct power. For the toy example 1 in Fig.3-1, our new paradigm with the chain decomposition $\mathcal{C}_{\mathsf{toy1}}$ will result in only 5 primes used while the Akl-Taylor's needs 9 primes. We will describe how to assign those powers over primes by an incidence matrix. We formalize the notion of incidence matrices that admit a secure scheme as *maximin matrix*:

Maximin Matrix. An $n \times m$ matrix $\{a_{ij}\}$ where $a_{ij} \in \mathbb{Z}_{\geq 0}$ is called a maximin matrix for set system X if for all $S \in \mathcal{S}_{\mathsf{X}}$, there exists j: $1 \leq j \leq m$ such that $\boxed{\max_{i \in S} a_{ij} < \min_{i \in N \setminus S} a_{ij}.}$ We give a formal treatment of RSA functions as accumulators and our construction first, then explain later.

RSA Accumulators. We fix a function $\mathsf{f} : \mathsf{U_f} \times \mathsf{E_f} \rightarrow \mathsf{U_f}$ to be an RSA function: $\mathsf{f}(x,e) := x^e \bmod n$ where $n = pq, p = 2p'+1, q = 2q'+1$ and p, q, p', q' are distinct odd primes. We restrict that $\mathsf{U_f}$ is the set of quadratic residues and $\mathsf{E_f}$ is the set of primes not equal to p', q'. We say f is generated from an RSA function generator $\mathsf{G_{RSA}}(1^\lambda)$. The function f is an instance of RSA accumulators, first proposed in [BD94], which has a quasi-commutative property: for all $x \in \mathsf{U_f}$, and $e_1, e_2 \in \mathsf{E_f}$, $\mathsf{f}(\mathsf{f}(x,e_1),e_2) = \mathsf{f}(\mathsf{f}(x,e_2),e_1)$. If $E = \{e_1, ..., e_h\}$ where each $e_i \in \mathsf{E_f}$, then we denote $\mathsf{f}(x,E) := \mathsf{f}(\mathsf{f}(...\mathsf{f}(x,e_1),...),e_h)$. Note that a set E is threaten as a multi-set, where the repetition of members is important. We thus denote a repetition of a member e which occurs t_e times as $t_e \lhd e$. For example, $\mathsf{f}(x, \{s \lhd e_1, t \lhd e_2\}) = x^{(e_1^s \cdot e_2^t)}$.

Construction 3-4. Non-Trapdoor-RSA-based Access Control Scheme: $(\mathsf{X})^{\mathsf{acc}}$

Keygen : Run a $\mathsf{G_{RSA}}$ to obtain a description of $\mathsf{f} : \mathsf{U_f} \times \mathsf{E_f} \rightarrow \mathsf{U_f}$. Pick a random secret $s \in \mathsf{U_f}$. For $1 \leq j \leq m$, pick an element $p_j \in \mathsf{E_f}$. Let pub consist of all p_j's and $\{a_{ij}\}$; indeed we let user derive prime p_j only when necessary by predetermining the intervals of those primes (see below). Let

$$\begin{array}{rcl} \mathsf{p}(u) & = & \mathsf{f}(s, \{a_{uj} \lhd p_j : 1 \leq j \leq m\}), \\ \mathsf{k}(S) & = & \mathsf{f}(s, \{(\max_{i \in S} a_{ij}) \lhd p_j : 1 \leq j \leq m\}). \end{array} \tag{3.7}$$

[2] In the latter, a sequence of integers $\{x_j\}$ is pre-specified by the broadcaster and p_i is defined to be the first prime in $[x_i, x_{i+1})$; the program to recognize $\{x_j\}$ has negligible size (cf. [Asa02]). More primes imply more computational cost on-the-fly.

for user $u \in N$ and set $S \in \mathcal{S}_{\mathsf{X}}$.

Derive : Compute $\mathsf{k}(S) = \mathsf{f}(\mathsf{p}(u), \{(\max_{i \in S} a_{ij} - a_{uj}) \lhd p_j : 1 \leq j \leq m\})$.

First it is easy to see that the correctness holds: Derive is computable. Next we will give an intuition as to why for each $S \in \mathcal{S}$, the collusion of all users from $N \setminus S$ cannot compute the key of S. Informally, the best they can do is to obtain the value with the same base s and the exponent term being GCD of all the exponent terms of the keys for users in $N \setminus S$, which is $\prod_{j=1}^{m} p_j^{\min_{i \in N \setminus S} a_{ij}}$ (by the well-known trick involving using the extended Euclid's algorithm). To be able to compute the key of S, it must divide $\prod_{j=1}^{m} p_j^{\max_{i \in S} a_{ij}}$. But this will not happen due to the property of the maximin matrix.

Theorem 3.17. $(\mathsf{X})^{\mathsf{acc}}$ *is KINT-2-secure assuming the strong RSA assumption.*

Before proving the theorem we note that the essential property of RSA accumulator to enable the access control mechanism has been already captured into a useful primitive called decomposably-one-way accumulators which is introduced in [AKI03b]. We restate its definition and a lemma as the following. We will then return to prove the security of $(\mathsf{X})^{\mathsf{acc}}$ based on the notion of accumulators instead of directly proving from the strong RSA assumption.

Definition 3.18 (dOW Security of Accumulators)**.** ([AKI03b]) Let G be an accumulator generator. We say that G is secure in the sense decomposably one-wayness (dOW) if for all polynomial-time randomized adversary $\boldsymbol{F}$ its success probability $\Pr[\mathbf{Exp}^{\mathsf{dOW}}(\mathsf{G}, \boldsymbol{F}) = 1]$ is a negligible function in λ where we define

$$
\begin{array}{rcl}
\text{Experiment } \mathbf{Exp}^{\mathsf{dOW}}(\mathsf{G}, \boldsymbol{F}) & & \\
\mathsf{G}(1^{\lambda}) & \rightarrow & \mathsf{f}; \\
\mathsf{U}_{\mathsf{f}} & \rightarrow & x; \\
(E_1, ..., E_t, \mathsf{state}) & \leftarrow & \boldsymbol{F}_{\mathsf{find}}(\mathsf{description}_{\mathsf{f}}); \\
(E^*, w) & \leftarrow & \boldsymbol{F}_{\mathsf{guess}}(\mathsf{f}(x, E_1), ..., \mathsf{f}(x, E_t), \mathsf{state}) \\
\text{return 1 iff } w = \mathsf{f}(x, E^*) \text{ and } \bigcap_{i=1}^{t} E_t \not\subseteq E^* & &
\end{array}
$$

Lemma 3.19. *([AKI03b]) The RSA function generator* $\mathsf{G}_{\mathsf{RSA}}$ *is secure in the sense of* dOW *assuming the strong RSA assumption.*

Now we return to the proof of the theorem.

Proof. (of Theorem 3.17) Let $\boldsymbol{A}$ be an adversary which breaks KINT-2 security of $\mathsf{X}^{\mathsf{acc}}$ with probability ϵ. We present an algorithm $\boldsymbol{F}$ which breaks the dOW security of accumulators with equal probability (thus tight reduction). The construction is as follows:

Find Stage. First $\boldsymbol{F}_{\mathsf{find}}$ upon input the description $\mathsf{desc}_{\mathsf{f}}$ of accumulators from the challenger randomly chooses m elements from E_{f} namely $p_1, ..., p_m$. (m is the number of columns of maximin matrix for the set system X.) It then runs $\boldsymbol{A}$ which consequently output a set of corrupt users $C = \{i_1, ..., i_z\} \subset N$ at once. To provide the answer, $\boldsymbol{F}_{\mathsf{find}}$ just outputs to its challenger $E_k = \{a_{i_k j} \lhd p_j : 1 \leq j \leq m\}$ for $1 \leq k \leq z$, and waits for the result while goes to the next stage.

Guess Stage. The adversary $\boldsymbol{F}_{\mathsf{guess}}$, upon a response $\mathsf{f}(x, E_1), ..., \mathsf{f}(x, E_z)$ from the challenger, can now answer to $\boldsymbol{A}$ by just forwarding these values. These perfectly simulate keys for $i_1, ..., i_z$. $\boldsymbol{A}$ then outputs $(S^*, \mathsf{k}(S^*))$ where $S^* \cap C = \emptyset$. Now $\boldsymbol{F}_{\mathsf{guess}}$ just outputs $(E^*, w) = (\{(\max_{i \in S^*} a_{ij}) \lhd p_j : 1 \leq j \leq m\}, \mathsf{k}(S^*))$.

If $\boldsymbol{A}$ wins its game then it must be that $w = \mathsf{f}(x, E^*)$. To see that $\boldsymbol{F}$ wins the game, we only have to show that $\bigcap_{k=1}^{z} E_k \not\subseteq E^*$, that is:

$$\bigcap_{i \in C} \{a_{ij} \lhd p_j : 1 \leq j \leq m\} \not\subseteq \{(\max_{i \in S^*} a_{ij}) \lhd p_j : 1 \leq j \leq m\}.$$

This follows from the fact that there is j' such that $\max_{i \in S^*} a_{ij'} < \min_{i \in N \setminus S^*} a_{ij'} \leq \min_{i \in C} a_{ij'}$. (Due to the property of the maximin matrix and that $C \subseteq N \setminus S^*$ respectively). Consider the element $p_{j'}$, there are $\min_{i \in C} a_{ij'}$ such elements in the left-hand-side while only $\max_{i \in S^*} a_{ij'}$ such elements in the right-hand-side. Therefore the uninclusion holds. □

Constructing a Maximin Matrix. Consider a chain decomposition $\mathcal{C} = \{G_1, ..., G_m\}$ of $\mathcal{S}_{\mathsf{X}}$. For each chain $G_j : S_1 \to \cdots \to S_l$, construct j-th column by letting

$$a_{ij} := \begin{cases} 0 & \text{if } i \in S_1 \\ w & \text{if } i \in S_{w+1} \setminus S_w \\ l & \text{otherwise} \end{cases} \tag{3.8}$$

Proposition 3.20. *The above construction is a maximin matrix. Moreover, $\mathcal{C}$ with the minimum number of chains will imply the maximin matrix with the minimum m, the number of all primes used.*

Proof. We will prove that the construction by Eq.(3.8) is a maximin matrix for X. Consider arbitrary $S \in \mathcal{S}$, observe that there is a chain $G_j : S_1 \to \cdots \to S_l$ and some w, $0 \leq w \leq l-1$, such that $S = S_{w+1}$ (since $\mathcal{C}$ is a chain decomposition). For all $i \in S$ we have $0 \leq a_{ij} \leq w$ by the construction. For all $i' \in N \setminus S$ we have $w < a_{i'j}$ also by the construction. This implies $\max_{i \in S} a_{ij} \leq w < \min_{i' \in N \setminus S} a_{i'j}$ which is what we wanted to prove.

To prove the second claim, it is sufficient to prove the converse of the first claim: from any maximin matrix for X one can construct a a chain decomposition in which the number of chains is less than or equal to the number of columns of the matrix. To do this we fix a maximin matrix. We delete some columns so that the existence of a column j for inequality of maximin matrix is unique for each $S \in \mathcal{S}$. We then minus all the entries of the matrix by the minimum entry(entries) so that the minimum entry(entries) becomes 0. Note that the modified matrix is still a maximin matrix. Let m be the number of columns. For the j-th column we construct a chain $G_j : S_1 \to \cdots \to S_l$ by letting $S_{w+1} := \{i : a_{ij} \leq w\}$. Here G_j is indeed a chain since $S_w \subset S_{w+1}$ for $w \geq 0$. By the definition, for each $S \in \mathcal{S}$ there is a unique column j' such that $\max_{i \in S} a_{ij'} < \min_{i \in N \setminus S} a_{ij'}$, thus $S = S_{\max_{i \in S} a_{ij'}}$ due to our chain construction. Therefore $\{G_j\}_{1 \leq j \leq m}$ is indeed a chain decomposition. □

Characterizing Efficiency. We will generate primes on the fly using the technique in [Asa02] (cf. Footnote 2 in Section 3.5.1). Without going into detail, this technique requires computational cost $O(\log^4 P)$ to generate one prime, and produces each prime of size $O(P \log P)$, where P is the number of all primes needed in such a scheme. In our scheme, $P = m$. Note that only when $P = O(1)$, it is worthless to use this technique; we

just store the least P primes (which requires only negligible storage) so the cost for prime generation in this case is $O(1)$.

Using the notation defined earlier, we have that $\mathsf{RN}_{\mathcal{C}}(u)$ represents the number of chains in $\mathcal{C}$ that u appears; and $\mathsf{DD}_{\mathcal{C}}$ represents the length of the longest chain in $\mathcal{C}$. The number of all chains in $\mathcal{C}$ is $|\mathcal{C}|$ (and$= m$). We obtain:

$$\begin{aligned}\mathsf{KeySize}_{(\mathsf{X})^{\mathsf{acc}}}(u) &= 1,\\ \mathsf{CompCost}_{(\mathsf{X})^{\mathsf{acc}}} &= O(\mathsf{MC}_{\mathcal{C}}^{\mathsf{acc}} + \mathsf{PC}_{\mathcal{C}}^{\mathsf{acc}}),\\ \mathsf{PubSize}_{(\mathsf{X})^{\mathsf{acc}}} &= |\mathcal{C}| \cdot n \log n \ \ \text{bits}^3,\end{aligned}$$

where $\mathsf{MC}_{\mathcal{C}}^{\mathsf{acc}}(u), \mathsf{PC}_{\mathcal{C}}^{\mathsf{acc}}(u)$ are the cost due to Modular exponentiation and on-the-fly Prime generation for user u respectively and $\mathsf{MC}_{\mathcal{C}}^{\mathsf{acc}} := \max_{u\in N} \mathsf{MC}_{\mathcal{C}}^{\mathsf{acc}}(u)$, $\mathsf{PC}_{\mathcal{C}}^{\mathsf{acc}} := \max_{u\in N} \mathsf{PC}_{\mathcal{C}}^{\mathsf{acc}}(u)$. Such costs depend solely on $\mathcal{C}$ and can be characterized as:

$$\begin{aligned}\mathsf{MC}_{\mathcal{C}}^{\mathsf{acc}}(u) &= O(\mathsf{DD}_{\mathcal{C}} \cdot (\log|\mathcal{C}|) \cdot \mathsf{RN}_{\mathcal{C}}(u)),\\ \mathsf{PC}_{\mathcal{C}}^{\mathsf{acc}}(u) &= \begin{cases} O(1) & \text{if } |\mathcal{C}| = O(1),\\ O((\log^4|\mathcal{C}|) \cdot \mathsf{RN}_{\mathcal{C}}(u)) & \text{otherwise.}\end{cases}\end{aligned}$$

The analysis are as follows. The cost of modular exponentiation for computing Derive is logarithm in the exponent term which is $\prod_{j=1}^{m} p_j^{(\max_{i\in S} a_{ij} - a_{uj})}$. To determine its complexity, observe that $\max_{i\in S} a_{ij} = a_{uj}$ for all but only $\mathsf{RN}_{\mathcal{C}}(u)$ terms of j due to Eq.(3.8) and the fact that u appears only $\mathsf{RN}_{\mathcal{C}}(u)$ chains. Also, observe that $\max_{i\in S} a_{ij} - a_{uj} \leq \mathsf{DD}_{\mathcal{C}}$ due to Eq.(3.8). Each p_j is $O(m \log m)$, hence has bit length $O(\log m)$. Combining these, we get $\mathsf{MC}_{\mathcal{C}}^{\mathsf{acc}}(u)$ as above. The cost for prime-generation above follows from the fact that the number of primes to be generated when deriving keys are $\mathsf{RN}_{\mathcal{C}}(u)$.

Remark 3.21. The MC of our scheme is asymptotically optimal among all non-trapdoor RSA-accumulator based paradigms (if there are any others) since it matches the lower bound in [GR04], which states that the optimal MC is of the same order as the number of subsets (in the set system) that one user is in, albeit here we calculate in bit complexity which includes the size of primes.

Remark 3.22. The Akl-Taylor's scheme [AT83] is a special case of our framework where the trivial chain decomposition (the collection of all one-node chains) is used.

Remark 3.23. Instead of storing the matrix $\{a_{ij}\}$ directly, one can just store the description of the chain decomposition, which can be embedded in the description of $\mathcal{S}$ by describing $\mathcal{S}$ as an *ordered set* with some special symbol to indicate the end of each chain, instead of an unordered set as usual. Since $\mathcal{S}$ is already the necessary information for any scheme (as the whole BE scheme), the public storage overhead due to this matrix can be thus considered negligible.

Even better, if the chain decomposition admits a simple formula, we just store the description of that formula and let the index of the chain be ordered lexicographically. This is the case for our instantiations of SIC,LSIC (cf. Eq.(3.5),(3.6)).

[3] We discuss how to reduce this to negligible in Remark 3.23.

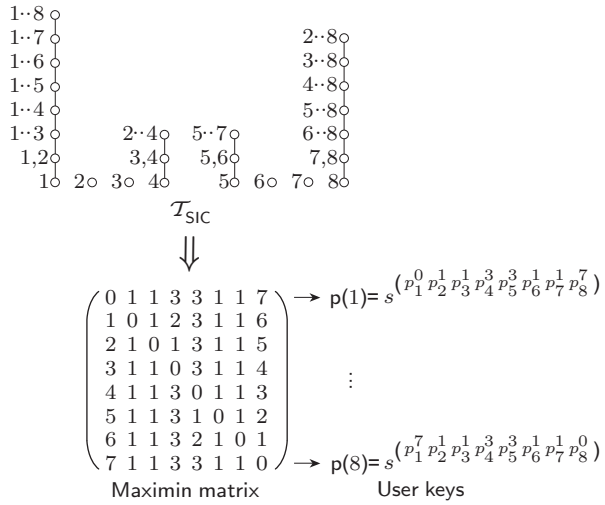


Figure 3-6: Instantiating SIC ($n = 8$) by the non-trapdoor RSA accumulator based framework

3.5.2 Non-Trapdoor RSA based Instantiation for SIC, LSIC

Non-Trapdoor-RSA-based Instantiation for SIC, LSIC. We will state the result for LSIC so that the result for SIC can be obtained by setting $k = 1$. It suffices to define a chain decomposition of $\mathcal{S}_{\mathsf{LSIC}[k]}$ and the concrete scheme will follow automatically. We choose a chain decomposition $\mathcal{C}_{\mathsf{LSIC}[k]} = \mathcal{T}_{\mathsf{LSIC}[k]}$ defined in Eq.(3.6). (Note that it is obvious that $\mathcal{T}_{\mathsf{LSIC}[k]}$ was also a chain decomposition). A concrete example for $(\mathsf{SIC})^{\mathsf{acc}}$ is shown in Fig.3-6 for $n = 8$. As an example, the subset key $\mathsf{k}(567) = s^{(p_1^6 p_2^1 p_3^1 p_4^3 p_5^2 p_6^1 p_7^1 p_8^3)}$.

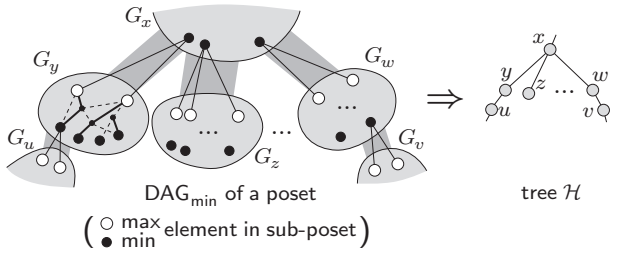
The following result follows directly from Theorem 3.12, 3.15 and the generic efficiency characterization of the framework with the fact that $|\mathcal{C}_{\mathsf{LSIC}[k]}| = n$.

Corollary 3.24. *We have that*

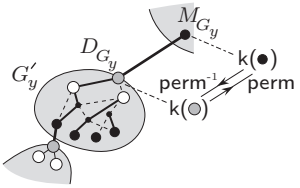
$$\begin{aligned} \mathsf{MC}^{\mathsf{acc}}_{\mathcal{C}_{\mathsf{LSIC}[k]}} &= O(kn^{1/k}\log^2 n), \\ \mathsf{PC}^{\mathsf{acc}}_{\mathcal{C}_{\mathsf{LSIC}[k]}} &= O(\log^5 n). \end{aligned}$$

Scheme $(\mathsf{LSIC}[k])^{\mathsf{acc}}$ has computational cost $O(\max\{kn^{1/k}\log^2 n, \log^5 n\})$. For trillion users ($n = 10^{12}$), choose k as low as 4 we have $4n^{1/4}\log^2 n < \log^5 n$ so that the computational cost is dominant by the latter, which is roughly as in Asano's scheme (but ours enjoy exceptionally lower header size).

Remark 3.25. If we instantiate with Akl-Taylor's, its chain decomposition has $\max_{u\in N} h_u = O(n^{1/k}\log n)$, and $m = O(2^k \cdot n^{1/k}(\log n)/k)$. Thus $\mathsf{PC} = O(n^{1/k}\log^5(n))$, which is much worse than ours, $O(\log^5 n)$. Moreover, this cost always dominates over the optimal MC for LSIC, $O(n^{1/k}\log^2 n)$.



(a) Tree-stratifiability



(b) Key derivation between subposets

Figure 3-7: The underlying idea for the trapdoor RSA based framework

3.6 Key Derivation based on Trapdoor RSA Accumulator

3.6.1 Trapdoor RSA based Framework

Framework Idea. The framework in this section is applicable to a class of posets that we call *tree-stratifiable posets*. Informally, such a poset of this type is defined as one which can be considered as formed by a tree hierarchy of atomic posets (not necessarily homogeneous), as shown in Fig.3-7. There, the graph decomposition $\mathcal{G} = \{G_x, G_y, G_z, ...\}$ is said to form a hierarchy represented by tree $\mathcal{H}$ where $V(\mathcal{H}) = \{x, y, z, ...\}$. Intuitively, such a graph decomposition is said to form a hierarchy if all the inclusion relations from every node in a lower subgraph (one with a lower index in the hierarchy), say G_y in the figure, to the next upper one in the hierarchy, G_x, are via a unique minimal node in that upper subgraph. Denote this minimal node as M_{G_y}. We will put a "dummy node" in each subgraph so that it will be the "representative" of that poset to reach that unique minimal node in the upper poset. (In the figure, the dummy node is D_{G_y} for subgraph G_y to reach M_{G_y}).

The idea for key derivations are as follows. First we define the key for each node in the highest sub-poset in the hierarchy by using the RSA-based framework in the last section. Recursively in a top-down fashion, we will define the set of keys corresponding to each lower sub-poset in the hierarchy. At some point, the set of keys for the nodes in G_x are defined. Then we define the "dummy key" for the dummy node in a next lower level sub-poset by applying a random permutation perm (w.l.o.g we will use the reverse direction) to the key of the minimal element in that upper sub-poset that it connects, that is, $\mathsf{k}(D_{G_y}) = \mathsf{perm}^{-1}(\mathsf{k}(M_{G_y}))$. To define keys for the other nodes in this lower sub-poset (at G_y), we will again use the RSA-based framework for that sub-poset. However, this time the key for the dummy node has been already determined, while all the keys must agree with the relations of $(G'_y)^{\mathsf{acc}}$, where G'_y is the modified subgraph that includes the dummy node, i.e., the relation of keys as defined in Eq.(3.7) instantiated to a poset that has G'_y as its representation. To solve this, it suffices to use the *trapdoor* of RSA. In this way, we can define keys recursively until reaching the lowest sub-posets. Users, on the other hand, do not have to use trapdoor since they only compute keys in the bottom-up fashion. Note that $(\mathsf{perm}, \mathsf{perm}^{-1})$ is a public permutation, such as any block cipher with a fixed known

key. We will model perm as an ideal random permutation in the security proof (the random permuation model).

The idea of reducing the whole poset by instantiating RSA-based framework in each sub-poset results in the use of only small number of primes for the overall scheme since the same set of primes can be used across different instantiations for different sub-posets.

To formalize this, we first define some more notations. For a directed graph G, denote $V_{\mathsf{min}}(G)$ the set of all minimal elements of poset $\mathcal{S}$ such that $\mathsf{DAG}_{\mathsf{min}}(\mathcal{S}) = G$. $V_{\mathsf{max}}(G)$ is defined analogously. The definition below captures what we have explained in the framework idea. Essentially, the bijection π below maps $G_x \mapsto x$.

Definition 3.26. (Tree-Stratifiable Poset) An inclusion poset $\mathcal{S}$ is called *tree-stratifiable poset* iff there exist an induced graph decomposition $\mathcal{G}$ of $\mathcal{S}$ and a tree $\mathcal{H}$ with a bijection $\pi : \mathcal{G} \to V(\mathcal{H})$ such that for each $G \in \mathcal{G}$ if we define G' by letting $V(G') = V(G) \cup \{D_G\}$ and $E(G') = E(G) \cup \{(S, D_G) : S \in V_{\mathsf{max}}(G)\}$ where D_G is a dummy node; define $M_G := \bigcup_{S \in V_{\mathsf{max}}(G)} S$; and define a graph $\mathcal{W}$ by letting $V(\mathcal{W}) = \bigcup_{G \in \mathcal{G}} V(G')$ and $E(\mathcal{W}) = \bigcup_{G \in \mathcal{G}} \big(E(G') \cup \{(D_G, M_G)\}\big)$, then we have that (1) for all $G \in \mathcal{G}$, $M_G \in V_{\mathsf{min}}(\pi^{-1}(\mathsf{parent}_{\mathcal{H}}(\pi(G))))$ and (2) $E(\mathsf{DAG}_{\mathsf{min}}(\mathcal{S})) \subseteq E(\mathsf{DAG}_{\mathsf{max}}(\mathcal{W}))$. □

Trapdoor RSA Accumulators. A trapdoor RSA function generator $\mathsf{G}_{\mathsf{tRSA}}$ is the one that works exactly the same as $\mathsf{G}_{\mathsf{RSA}}$ but in addition also outputs the *trapdoor* td which is $\phi(n)$ where ϕ is the Euler's phi function. With td, given the description of f, any $y \in \mathsf{U}_{\mathsf{f}}$, and a (multi-)set of accumulated values E, one can efficiently compute $x \in \mathsf{U}_{\mathsf{f}}$ such that $\mathsf{f}(x, E) = y$. Denote such x by $\mathsf{f}_{\mathsf{td}}(y, E^{-1})$.

Towards formalizing the construction, we "normalize" each sub-poset $G \in \mathcal{G}$ so that its base set will be $B_G = \{1, ..., |V_{\mathsf{min}}(G')|\}$ as follows. Construct $\gamma : V(G') \to 2^{B_G}$ by first picking an injective map $\tilde{\gamma} : V_{\mathsf{min}}(G) \to B_G$ then define for $S \in V(G')$, $\gamma : S \mapsto \{\tilde{\gamma}(U) : U \in V_{\mathsf{min}}(G),\ U \subseteq S\}$. Let $\mathcal{S}_G = \gamma(V(G'))$ (the set of all images by γ from $V(G')$) be the set system with the base set B_G.

We now give the formal description for the construction. For simplicity we will consider homogeneously stratifiable poset, i.e., each $\mathcal{S}_G$ is isomorphic to each other (in the sense that its corresponding DAG is isomorphic), say the set system Y. Let $\{a_{ij}\}_{1 \le i \le d, 1 \le j \le m}$ be a maximin matrix for set system Y, where d is the cardinality of its base set.

Construction 3-5. Trapdoor-RSA-based Access Control Scheme: $(\mathsf{X})^{\mathsf{tacc}}$

Keygen : Run a $\mathsf{G}_{\mathsf{tRSA}}$ to obtain a description of $\mathsf{f} : \mathsf{U}_{\mathsf{f}} \times \mathsf{E}_{\mathsf{f}} \to \mathsf{U}_{\mathsf{f}}$ and trapdoor td. For $1 \le j \le m$, pick an element $p_j \in \mathsf{E}_{\mathsf{f}}$. Let perm and perm^{-1} be a publicly available permutation mapping $\mathsf{U}_{\mathsf{f}} \to \mathsf{U}_{\mathsf{f}}$. Let pub consist of all p_j's and $\{a_{ij}\}$. Pick a random $t \in \mathsf{U}_{\mathsf{f}}$. Define keys recursively in a top-down fashion in the tree $\mathcal{H}$:

[**Top**]. At the subgraph $G_{\mathsf{root}} \in \mathcal{G}$, where root is the root of $\mathcal{H}$, by definition we have $N = M_{G_{\mathsf{root}}}$. We let $\mathsf{k}(N) = \mathsf{k}(M_{G_{\mathsf{root}}}) = t$.

[**Intermediate**]. At each atomic subgraph $G \in \mathcal{G}$, the key $\mathsf{k}(M_G)$ is previously determined. Define the key for the dummy node: $\mathsf{k}(D_G) = \mathsf{perm}^{-1}(\mathsf{k}(M_G))$. By using the trapdoor td and $\mathsf{k}(D_G)$, we solve Eq.(3.11) by setting $S = D_G$ (thus $\gamma(S) = B_G$) to determine the secret s_G, i.e.,

$$s_G = \mathsf{f}_{\mathsf{td}}(\mathsf{k}(D_G), \{(\max_{i \in B_G} a_{ij}) \lhd p_j : 1 \le j \le m\}^{-1}). \tag{3.9}$$

Then we define the key at each element in this subgraph, $S \in V(G)$, as follows.

For $S \in V_{\mathsf{min}}(G)$),

$$\mathsf{k}(S) = \mathsf{f}(s_G, \{a_{\tilde{\gamma}(S),j} \lhd p_j : 1 \leq j \leq m\}), \tag{3.10}$$

and for $S \in V(G)$),

$$\mathsf{k}(S) = \mathsf{f}(s_G, \{(\max_{i \in \gamma(S)} a_{ij}) \lhd p_j : 1 \leq j \leq m\}). \tag{3.11}$$

[**Bottom**]. For each $u \in N$, we let $\mathsf{p}(u) = \mathsf{k}(\{u\})$.

Derive : Compute from the relations given in Eq.(3.9), (3.10), (3.11) but in the *bottom-up* fashion by using applications of $\mathsf{f}(\cdot,\cdot)$, $\mathsf{perm}(\cdot)$ starting from $\mathsf{f}(\mathsf{p}(u),\cdot)$. Note that td is not required to do this.

Theorem 3.27. $(\mathsf{X})^{\mathsf{tacc}}$ *is KINT-1-secure in the random permutation model (*perm *as an ideal random permutation), assuming the strong RSA assumption.*

Proof. Let $\boldsymbol{A}$ be an adversary which breaks KINT-1 security of $\mathsf{X}^{\mathsf{tacc}}$ with probability ϵ. We present an algorithm $\boldsymbol{F}$ which breaks the dOW security of accumulators with equal probability (thus tight reduction). The construction is as follows:

Find Stage. First $\boldsymbol{F}_{\mathsf{find}}$ upon input the description $\mathsf{desc_f}$ of accumulators from the challenger randomly chooses m elements from $\mathsf{E_f}$ namely $p_1, ..., p_m$. (m is the number of columns of maximin matrix of the *atomic* set system composing X.) It then runs $\boldsymbol{A}$ which consequently output a challenge subset $S^* \in \mathcal{S}_{\mathsf{X}}$. $\boldsymbol{F}_{\mathsf{find}}$ has to provide the keys for users in $N \setminus S^*$. Let $B \setminus \gamma(S^*) = \{i_1, ..., i_z\}$. It then just outputs to its challenger $E_k = \{a_{i_k j} \lhd p_j : 1 \leq j \leq m\}$ for $1 \leq k \leq z$, and waits for the result while goes to the next stage.

Guess Stage. The adversary $\boldsymbol{F}_{\mathsf{guess}}$, upon a response $\mathsf{f}(x, E_1), ..., \mathsf{f}(x, E_z)$ from the challenger, simulates the keys and also the random permutation oracle $\mathsf{Perm}/\mathsf{Perm}^{-1}$ as follows. Denote the sub-poset which S^* is in by G^*. Beside user keys for the corrupted users in $N \setminus S^*$, $\boldsymbol{F}_{\mathsf{guess}}$ will simulate also keys which are derivable from them so as to ensure the overall consistency in derivation. Such keys are those of S such that $S \setminus S^* \neq \emptyset$. We categorize to two kinds: (1) those in G^*; (2) those outside G^*, which can be confined to those in sub-posets in the collection denoted $\mathcal{G}'$ which includes all except sub-posets below $U \in V_{\mathsf{min}}(G^*)$ such that $U \subseteq S^*$.

Simulate Keys in (1)**:** For each $S \in V_{\mathsf{min}}(G^*)$ such that $S \setminus S^* \neq \emptyset$ (this implies $S \not\subseteq S^*$) we let $\mathsf{k}(S) = \mathsf{f}(x, E_k)$ where $\tilde{\gamma}(S) = i_k$. These keys are perfectly simulated since they are in the form of Equation (3.10) where $s_{G^*} = x$ (which is unknown). For $S \in V(G^*)$ such that $S \setminus S^* \neq \emptyset$, we have that there exists $U \in V_{\mathsf{min}}(G^*)$ such that $U \not\subseteq S^*$ and $U \subseteq S$, thus $\mathsf{k}(S)$ can be derived from $\mathsf{k}(U)$ (which is previously defined only if $U \not\subseteq S^*$) by using the relation from Eq. (3.10),(3.11).

Simulate Keys in (2)**:** For each subgraph $G \in \mathcal{G}'$, $\boldsymbol{F}_{\mathsf{guess}}$ picks randomly $s_G \in \mathsf{U_f}$ and let the keys in this sub-poset be defined as in Equation (3.10),(3.11). Give the user keys $\mathsf{p}(u)$ for all $u \in N \setminus S^*$, which are simulated at the lowest sub-posets in the tree $\mathcal{H}$, to $\boldsymbol{A}$.

Simulate $\mathsf{Perm}/\mathsf{Perm}^{-1}$**:** For each $G \in \mathcal{G}' \cup \{G^*\}$, we define

$$\mathsf{Perm} : \mathsf{k}(M_G) \mapsto \mathsf{f}(s_G,\ \{(\max_{i \in B} a_{ij}) \lhd p_j : 1 \le j \le m\})$$

and collect it to the list of input-output pairs of Perm. Note that s_{G^*} is unknown but the above right-hand-side value when $G = G^*$ can be computed from $k(S)$ for any $S \in V(G^*)$ (which some are known). Queries to the random permutation oracle $\mathsf{Perm}/\mathsf{Perm}^{-1}$ will be answered according to this list in both direction. When a query is not in the list, we just output a random string in U_f (and make sure that it does not collide with some previous used strings) and record this to the list. The consistency of keys across sub-posets required from Equation (3.9) is ensured by this simulation which is perfect.

Consequently, $\boldsymbol{A}$ then outputs $\mathsf{k}(S^*)$. Now $\boldsymbol{F}_{\mathsf{guess}}$ just outputs $(E^*, w) = (\{(\max_{i \in \gamma(S^*)} a_{ij}) \lhd p_j : 1 \le j \le m\}, \mathsf{k}(S^*))$.

If $\boldsymbol{A}$ wins its game then it must be that $w = \mathsf{f}(x, E^*)$. To see that $\boldsymbol{F}$ wins the game, we only have to show that $\bigcap_{k=1}^{z} E_k \not\subseteq E^*$, that is:

$$\bigcap_{i \in B \smallsetminus \gamma(S^*)} \{a_{ij} \lhd p_j : 1 \le j \le m\} \not\subseteq \{(\max_{i \in \gamma(S^*)} a_{ij}) \lhd p_j : 1 \le j \le m\}.$$

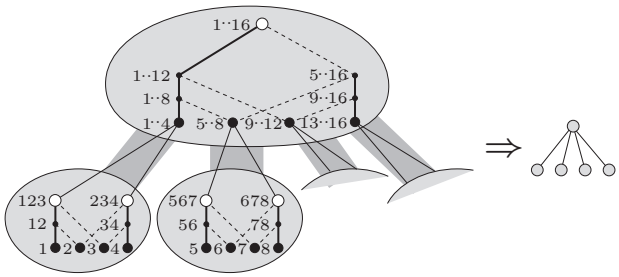
This follows from the property of the maximin matrix that there is j' such that $\max_{i \in \gamma(S^*)} a_{ij'} < \min_{i \in B \smallsetminus \gamma(S^*)} a_{ij'}$. Consider the element $p_{j'}$, there are $\min_{i \in B \smallsetminus \gamma(S^*)} a_{ij'}$ such elements in the left-hand-side while only $\max_{i \in \gamma(S^*)} a_{ij'}$ such elements in the right-hand-side. Therefore the uninclusion holds. □

Remark 3.28. Instead of KINT-2 security as in the $(\mathsf{X})^{\mathsf{acc}}$ scheme, we have proved the KINT-1 security for this the $(\mathsf{X})^{\mathsf{tacc}}$ scheme since, intuitively, the difficulty arose if it were the KINT-2 game as the simulator cannot change the sub-poset where the problem instance is embedded, after the corruption. (Such an ability to change is required since we want S^* to be in the sub-poset where the problem instance is embedded, but we do not know S^* at the corruption phase).

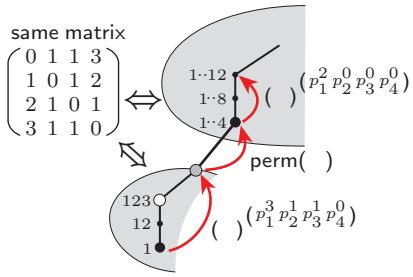
Nevertheless, if we indeed use the random self-reducibility of the accumulators (where the RSA ones have this property), the simulator can embed the problem instance at every sub-poset, and thus the KINT-2 security with tight reduction can be achieved, though. We omit the details here.

Characterizing Efficiency. If the set system X of n users is tree-stratifiable homogeneously into a set system Y of d users with the tree $\mathcal{H}$ then

$$\begin{aligned}
\mathsf{KeySize}_{(\mathsf{X})^{\mathsf{tacc}}}(u) &= 1,\\
\mathsf{CompCost}_{(\mathsf{X})^{\mathsf{tacc}}} &= O(\mathsf{MC}_{\mathsf{X}}^{\mathsf{tacc}} + \mathsf{PC}_{\mathsf{X}}^{\mathsf{tacc}}),\\
\mathsf{PubSize}_{(\mathsf{X})^{\mathsf{tacc}}} &= \mathsf{PubSize}_{(\mathsf{Y})^{\mathsf{acc}}}\ \ \mathrm{bits}^3,
\end{aligned}$$



(a) LSIC[k] is tree-stratifiable



(b) Derivation by user 1

Figure 3-8: Instantiating $\mathsf{LSIC}[k]$ ($n = 16, k = 2$, see Fig.3) by trapdoor RSA based framework

where the cost from modular exponentiation and prime generation are depended solely on both $\mathcal{H}, \mathsf{Y}$ and only Y respectively, and can be characterized as:

$$\mathsf{MC}_{\mathsf{X}}^{\mathsf{tacc}} = \mathsf{h}_{\mathcal{H}} \cdot \mathsf{MC}_{\mathcal{C}_{\mathsf{Y}}}^{\mathsf{acc}}, \quad \mathsf{PC}_{\mathsf{X}}^{\mathsf{tacc}} = \mathsf{PC}_{\mathcal{C}_{\mathsf{Y}}}^{\mathsf{acc}}, \tag{3.12}$$

where $\mathsf{h}_{\mathcal{H}}$ is the deepest depth of $\mathcal{H}$. The first claim follows from the fact that a user has to compute Eq.(3.11) for at most $\mathsf{h}_{\mathcal{H}}$ times. The second claim is from the fact that we reuse the same set of primes across sub-posets. There is also the cost due to applications of perm, which is $O(\mathsf{h}_{\mathcal{H}})$, but this is suppressed by MC.

Generic Application. We now confine our interest to the case where $\mathcal{H}$ is the balanced completed $n^{1/k}$-ary tree of depth $\mathsf{h}_{\mathcal{H}} = k$. This forces the base sets of Y and X to have cardinality $n^{1/k}$ and n respectively. In this case we say $\mathsf{X} = \mathsf{hier}_k(\mathsf{Y})$. The operation hier_k is well-defined and can be thought as the converse direction of tree-stratification; thus, from any poset Z one can construct a tree-stratifiable poset, namely $\mathsf{hier}_k(\mathsf{Z})$, by first scaling down the cardinality of the base set of Z to $n^{1/k}$. (Since usually any set system is originally defined in term of n). We write $\mathsf{Z}(n^{1/k})$ to emphasize the cardinality of base set. The point is that when k is a constant, Eq.(3.12) allows one to construct a full scheme of n users but with exactly the same asymptotic performances as those of $(\mathsf{Z}(n^{1/k}))^{\mathsf{acc}}$, which is a "scaled-down" scheme, in *both* parameters MC, PC! Moreover, if $\mathsf{c}_{\mathsf{Z}(n)}(n, r) = O(r)$ then we can show that $\mathsf{c}_{\mathsf{hier}_k(\mathsf{Z}(n^{1/k}))}(n, r) = O(kr) = O(r)$ (by exactly the same proof as that of Theorem 3.10); therefore, $\mathsf{HeaderSize}$ is also unaffected.

[3]Recall that Remark 3.23 shows how to reduce this to negligible.

3.6.2 Trapdoor RSA based Instantiation for LSIC

It is easy to see that $\mathsf{LSIC}[k]$ is tree-stratifiable since $\mathsf{LSIC}[k] = \mathsf{hier}_k(\mathsf{SIC}(n^{1/k}))$. (We could have define LSIC via hier operation rather than directly in Sec.3.3.2). An example is shown in Fig.3-8. From the efficiency characterization we have:

Corollary 3.29. *We have that*

$$\begin{array}{lll} \text{(i)} & \mathsf{MC}^{\mathsf{tacc}}_{\mathsf{LSIC}[k]} & = O(n^{1/k}(\log^2 n)/k), \\ & \mathsf{PC}^{\mathsf{tacc}}_{\mathsf{LSIC}[k]} & = O((\log^5 n)/k^5). \\ \text{(ii)} & \mathsf{MC}^{\mathsf{tacc}}_{\mathsf{LSIC}[\log_a n]} & = O(a \log a \log n), \\ & \mathsf{PC}^{\mathsf{tacc}}_{\mathsf{LSIC}[\log_a n]} & = O(1). \end{array}$$

Proof. From Corollary 3.24, we have

$$\begin{array}{ll} \mathsf{MC}^{\mathsf{acc}}_{\mathcal{C}_{\mathsf{SIC}(n^{1/k})}} & = O((n^{1/k} \log^2 n)/k^2), \\ \mathsf{PC}^{\mathsf{acc}}_{\mathcal{C}_{\mathsf{SIC}(n^{1/k})}} & = O((\log^5 n)/k^5); \\ \mathsf{MC}^{\mathsf{acc}}_{\mathcal{C}_{\mathsf{SIC}(a)}} & = O(a \log^2 a), \\ \mathsf{PC}^{\mathsf{acc}}_{\mathcal{C}_{\mathsf{SIC}(a)}} & = O(1). \end{array}$$

(In fact, for the case $\mathsf{SIC}(a)$, the maximum number of primes used per user is $\log a + 1$, a small constant). From these and from Eq.(3.12), we have the corollary statement. □

3.7 Concluding Remarks

We presented three generic frameworks for constructing broadcast encryption and give some efficient instantiations. Almost all subset-cover broadcast encryption schemes based on PRSG (or one-way function) or RSA accumulator in the literature can be rewritten as instantiations in our paradigms. In fact, [NNL01, HS02, M03, AKI03a, GST04, WNR04, JHC+05, HLL05, HL06] can be viewed as PRSG-instantiated schemes and [Asa02, AKI03b, GR04] are non-trapdoor-RSA-instantiated schemes from our frameworks.

The whole paradigm abstracts away the computational security issues and reduces the problem to only pure combinatorics. We leave as an open problem the question of showing any combinatorial bound from the efficiency characterization in each sub-framework. Note that the previous bounds for broadcast encryption [LS98] are done in the setting where no key derivation is involved.

Chapter 4

Unifying Public Key Encryption with "High Functionalities"

4.1 Introduction

Public-key encryption (PKE) is one of the most fundamental concept in modern cryptography. They allow any party to securely send a message to another securely, *i.e.,* the contents of message remain hidden from anyone intercepting the communication. This is done in such a way that there is no need for the sender Bob to share any secret information in the first place; all Bob needs to know is the public key of the intended recipient, Alice, who disseminate her public key as she likes, *e.g.,* publishing it in the internet and so forth.

In order to strengthen the security or to achieve some useful functionalities which are specific to applications thereof, the original concept of public-key encryption schemes has been extended in various ways, giving raise to many kinds of what we call "public-key encryption schemes with high functionalities". Some examples are ID-based encryption [Sha84, BF01], key-insulated encryption [DK02], forward-secure encryption [CHK03], certificate-based encryption [Gen03], and many more. Although it is well-known that these are related primitives as the approaches for constructing them are based on ID-based encryption, there was no unified systematic framework for defining or constructing them.

In brief, in this work, we unify these public-key encryption schemes with high functionalities in various aspects, namely a unified syntax, a unified security notion, and unified generic/specific constructions. More precisely, we reduce a specification of such a primitive to its necessary and sufficient information, which is its underlying graph: by specifying a graph, the definition and constructions will be automatically induced. We also give a primitive implication theorem which gives a criterion whether a primitive implies another.

In the following subsection, we explain motivations for those public-key encryption schemes with high functionalities which are useful in different scenarios. We will later describe our contributions in more details.

4.1.1 Background

Public-key encryption schemes with high functionalities solve some problems which basic public-key encryption cannot. Among existing proposals, we can classify them by their originally intended target problems to solve as follows.

Simplifying Public-key Infrastructure

Basic public-key encryption schemes, despite their great flexibility, does not address the issue of public key distribution problem. We have mentioned that Alice can publish her public key via, *e.g.,* the internet. But when Bob obtain Alice's public key, how does he know it really belongs to Alice and is not, for example, a fraud key created by a phisher Eve to deceive him? To solve problem of this kind, public key infrastructure (PKI) is utilized. The basic idea is to have a trusted authority known as CA (certification authority) digitally sign a message known as a certificate, thereby vouching that a particular key corresponds to a particular name. Bob then can check the authenticity of Alice's public key via the certificate. However, again, the management of certificates in public key directories are needed.

Shamir [Sha84] introduced the idea of *Identity-based Encryption* (IBE) to eliminate the need for PKI. The idea is very simple but elegant: it uses the identity of user to be served as the public key of that user, so that public keys do not need to be distributed at all. In order for Alice to obtain the private key corresponding to her name, she asks the private key generator (PKG), who has the master secret, to derive for it. Bob uses the global public key and the identity 'Alice' to produce an encrypted message. Although the concept was proposed two decades ago, it is only recently that the first fully functional schemes were proposed by Boneh and Franklin [BF01, BF03].

The advantage of IBE is clear: it eliminates the need for PKI. However, one unavoidable primary disadvantage is private key escrow, *i.e.,* the PKG can perform the decryption of any users (since in particular, it knows the master secret). *Certificate-based encryption* (CBE) [Gen03] and *Certificateless PKE* (CL-PKE) [AP03] are designed to preserve some advantages of IBE while eliminating this key escrow problem. Both have the same core mechanism: it allows Bob to encrypt messages by using Alice's (non-ID-based) public key as in the typical PKI but without requiring to check its authenticity, instead Alice have to know not only her private key but also the certificate (which is functioned as a secondary private key) in order to be able to decrypt.[1]

Protecting against Key Exposure

A normal public-key encryption scheme offers no security protection for any user whatsoever once his private key is compromised. As an extension to the normal variant in order to cope with the vulnerability against key exposure, the notion of forward security [Gün89] in the context of public-key encryption was first studied by Anderson [And97]. A *Forward-secure public-key encryption* (FS-PKE) allows Alice to update her private key periodically while keeping the public key unchanged. Such a scheme guarantees that even if an adversary learns the private key at some point, messages encrypted during all time periods prior to that point remain secret. The first non-trivial forward-secure PKE was proposed by Canetti, Halevi, and Katz [CHK03].

A similar scheme called *Key-insulated public-key encryption* was proposed by Dodis et al. [DK02]. In such a scheme, updating key must be done via the collaboration with the other module called helper server. *Intrusion-resilient PKE* [DFK+03] is a further strengthened

[1] For better understanding, one may think that the word 'certificate-based' is a property pointed to Alice as she needs the certificate to decrypt, while 'certificateless' is a property pointed to Bob as he need no ccertificate when encrypting.

version of Key-insulated PKE where its security is guaranteed as long as the adversary does not compromise both modules (helper and user) in the same time period.

Achieving New Additional Functions

Examples of schemes in this class vary in a wide range. *Public-key broadcast encryption*, which is a familiar theme in this thesis, provides a new function to PKE as it allows to encrypt to groups of users. *Public-key encryption with keyword search* [BDOP04], loosely speaking, allows one who possess keyword-based trapdoors to perform keyword-based search over encrypted data without decrypting it. (We will construct efficient schemes for their combination, *i.e.,* public-key broadcast encryption with keyword search, in Chapter 5). In *Time-capsuled PKE* [MHS03], one encrypts by also specifying an open time so that only from that time on, the message can be decrypt. In order to be able to do so, there is a releaser server, who periodically releases additional information to be used for decryption.

We note that many extensions such as hierarchical extensions are proposed. In *Hierarchical IBE* (HIBE) initiated by [HL02, GS02], many PKGs are incorporated in hierarchy, where upper-level PKG can derive private keys for lower-level ones, with users in the lowest level. Hierarchical version CBE is also proposed in its original paper [Gen03]. Some combinations such as Forward-secure HIBE [YFDL04] were also proposed.

4.1.2 Our Approaches and Contributions

In this work, we present a unified framework called *Directed Acyclic Graph Encryption* (DAGE). It has many consequences in the following. We first give a high-level overview of DAGE.

Intuition for Directed Acyclic Graph Encryption. Our starting point is IBE and HIBE. IBE schemes are indeed useful more than just managing identities and simplifying PKI. In particular, the "identity" is not limited only to someone's name, it can contain any specified information. Therefore we can abstract the notion of identity completely. Consequently, we can view an IBE scheme as a one-level tree hierarchy rooted at the PKG with each edge corresponding to a derivation of a secret key to a leaf which corresponds to a user. HIBE generalizes IBE to the case of arbitrary tree hierarchies.

Loosely speaking, DAGE is a generalization of HIBE to the case of directed acyclic graph (DAG) hierarchies, which also includes tree structures of HIBE as special cases. A DAG is a directed graph which contains no directed cycle in it. Another equivalent way to describe any directed acyclic graph is to consider any set of all edges that share the same end node as a hyper-edge denoted $\{\mathsf{V}_1, \ldots, \mathsf{V}_d\} \rightarrow \mathsf{V}^*$ to represent the set of all edges $\mathsf{V}_i \rightarrow \mathsf{V}^*$ in the DAG that directed to V^*. Intuitively, while every edge in tree graph abstraction of HIBE represents a derivation from the private key of *one node* to that of another, every hyper-edge in directed acyclic graph abstraction of DAGE represents a derivation from the private keys of *many nodes* to that of another node. The problem is how to define this "derivation", since it becomes involving many parent nodes. A natural way to define is to assign to this hyper-edge a "derivation rule" which is specified as an access structure Θ over $\{\mathsf{V}_1, \ldots, \mathsf{V}_d\}$. For example, $\Theta = \{\{\mathsf{V}_1, \mathsf{V}_2\}, \{\mathsf{V}_1, \mathsf{V}_3, \mathsf{V}_4\}\}$ means that the secret key of V^* can be derived only by the collaboration of node V_1 and V_2 or that of node V_1 and V_3 and V_4. The "collaboration" of nodes is done without exposing their own secret keys to each other. We formulate this by letting each node in collaboration output a "partial private

key" which then is given to V^* so that the secret key of V^* is a function of all these partial private keys, of which lacking even one it cannot derive a legitimate secret key. We call a DAG whose each hyper-edge is assigned an access structure a *Labeled Access DAG (LAD)*.

Unification. DAGE is very general since it deals with general graphs with general access structures associated; nevertheless, we are able to give a unified formal definition of syntax and its security notion for arbitrary labeled DAGs. It turns out that we can cast various public-key encryption with high-functionalities, as DAGE (varied by the underlying directed acyclic graphs), thus the security notion unifies the notions of these primitives. As examples, existing applications such as forward-secure encryption [CHK03], certificate-based encryption [Gen03], key-insulated encryption [DK02], public-key broadcast encryption [DF02] schemes and many more schemes described in the previous section are specific-case instantiations of DAGE.

The heart to our formalism of those primitives as DAGEs indeed can be thought out as to abstract away the very meaning of entity values such as identity, time, etc. with their relations such as derivation, updating, etc. and merely think of them as pure "strings" and "relations (among strings)". This directly lets us form graphs, which contain "nodes" and "hyper-edges (among nodes)", where relations are governed by the access structures assigned at hyper-edges.

Primitive Implications. We give a primitive implication theorem which provides a criterion whether a primitive implies another (both casted as DAGE). The criterion is purely *logical-based* and thus can be verified automatedly, in contrast with conventional *complexity-based* proof which has to be produced by hand and newly for any pair of primitive. This helps unifying and understanding more on properties of such primitives and simplifying many related works in this line, e.g., [BP02, DKXY03].

Constructions. We first present a generic construction of DAGE for any graph from HIBE. This demonstrates a possibility result that we can base DAGE on any HIBE. We next show that weak versions of DAGE can be constructed from PKE, which is a seemingly-weaker primitive, and a combinatorial family we call cover-admissible family. This demonstrates a possibility result that we can based DAGE on PKE, if we can construct such a family.

We show three more efficient constructions of DAGEs based on bilinear maps as follows.

- DAGE construction for any monotone OR-type of graphs with constant-size ciphertext. Combining this with AND-multiple encryption yields DAGE for general graphs.
- DAGE construction for any monotone AND-type of graphs with constant-size ciphertext. Combining this with OR-multiple encryption yields DAGE for general graphs.
- DAGE construction for what we called bounded-complete OR-type of graphs. Such a graph is analog of complete graphs in the context of directed graph.

Functionalities. DAGE itself provides a flexible way to define such a new primitive by just defining a "tailor-made" graph that fits for an application. Nevertheless, we define some prototype of functionalities for converting any DAGE to its forward-secure version, key-insulated version, time-capsuled version, or self-insulated version. This thus gives a

generic schema where we can get a scheme with combined functionalities or hierarchical version of one functionality. An example might be hierarchically-key-insulated certificateless encryption.

4.2 Definitions of DAGE

Derivability over Sets. We first review some terminology on access structures over sets. Consider a (possibly infinite) set N. We call the subsets in N which are allowed to derive some sort of secret *qualified*, and the subsets in N who should not be able to obtain any information about the secret *forbidden*. We call $\Theta \in 2^N$ a *qualified access structure* if Θ is monotone non-decreasing, *i.e.,* for each $\mathbf{V} \in \Theta$ we have that if $\mathbf{V} \subseteq \mathbf{W}$ then $\mathbf{W} \in \Theta$. We denote by $\Theta^{(\mathsf{min})}$ the collection of *minimal sets* in Θ. We call $\Psi \in 2^N$ a *forbidden access structure* if Ψ is monotone non-increasing, *i.e.,* for each $\mathbf{V} \in \Theta$ we have that if $\mathbf{W} \subseteq \mathbf{V}$ then $\mathbf{W} \in \Theta$. We denote by $\Psi^{(\mathsf{max})}$ the collection of *maximal sets* in Ψ. The tuple (Θ, Ψ) is called an access structure pair on N if $\Theta \cap \Psi = \emptyset$. If $\Theta \cup \Psi = 2^N$, then we say that (Θ, Ψ) is complete.

Since we often refer to the qualified one throughout the chapter, when we refer to access structure we mean a qualified access structure, unless otherwise is specified. For an access structure Θ, we call $\bigcup_{\mathbf{W} \in \Theta} \mathbf{W}$ the support of Θ and denote it by $\mathsf{Sup}(\Theta)$.

Labeled Access DAG. A *Labeled Access DAG* (or *LAD*) $\mathcal{G} = (\mathcal{V}, \mathcal{H})$ consists of the set $\mathcal{V}$ of all vertices and the set $\mathcal{H}$ of all *access hyper-edges*. Each vertex is uniquely specified by a *label* which is a string in $\{0,1\}^*$. An access hyper-edge $(\mathbf{V}, \mathsf{V}^*, \Theta) \in \mathcal{H}$ consists of a directed hyper-edge where $\mathbf{V} \subseteq \mathcal{V}$ is the initial node set, $\mathsf{V}^* \in \mathcal{V}$ is the terminate node, and a privileged access structure Θ over the set of initial nodes. In such a case, we also denote Θ by Θ_{V^*}.

We provide some notations here. We denote $\mathsf{W} \preccurlyeq \mathsf{V}$ if there is a directed path from W to V. For $\mathsf{W} \prec \mathsf{V}$ (that is, $\mathsf{W} \preccurlyeq \mathsf{V}$ but $\mathsf{W} \neq \mathsf{V}$), we write $\mathsf{W} \prec_{\mathsf{c}} \mathsf{V}$ if there exists no $\mathsf{V}' \in \mathcal{V}$ such that $\mathsf{W} \prec \mathsf{V}' \prec \mathsf{V}$. The indegree of node V is denoted by $\mathsf{indeg}(\mathsf{V})$. A source node is a node with indegree 0.

A Language of Propositional Logic. In the following, to formally reason about key derivability and ciphertext decryptability, we employ a language of propositional logic in which the set of connectives is $\{\mathtt{t}, \mathtt{f}, \neg, \Rightarrow, \vee, \wedge\}$ as usual and the set of symbols contains elements written in the form of $\langle\!\langle \mathsf{V} \rangle\!\rangle_{\mathcal{G}}$ where $\mathsf{V} \in \mathcal{V}_{\mathcal{G}}$ is a node in a LAD $\mathcal{G}$. If $\mathcal{G}$ is clear from the context, we abuse the notation as $\langle\!\langle \mathsf{V} \rangle\!\rangle$. The entailment, $\vdash$, denotes the syntactic consequence (or equivalently, semantic consequence) for propositional logic. For notational convenience, we define $[\mathbf{V}] = \bigwedge_{\mathsf{V} \in \mathbf{V}} \langle\!\langle \mathsf{V} \rangle\!\rangle_{\mathcal{G}}$ if $\mathsf{V} \subseteq \mathcal{V}_{\mathcal{G}}$ and $[\![\Theta]\!] = \bigvee_{S \in \Theta} \bigwedge_{\mathsf{V} \in S} \langle\!\langle \mathsf{V} \rangle\!\rangle_{\mathcal{G}}$ if Θ is an access structure over a set of node in $\mathcal{G}$.

Derivability over LAD. We now describe the derivability over LADs, which is analog to that of over sets described previously. For a LAD $\mathcal{G} = (\mathcal{V}, \mathcal{H})$ we define two sets of propositional formulae as follows.

- The *Positive Derivability* , Σ, is defined as a theory in propositional logic for capturing all the privileged accessability relations defined by $\mathcal{H}$:

$$\Sigma = \Big\{ [\![\Theta]\!] \Rightarrow \mathsf{V}^* \mid (\mathbf{V}, \mathsf{V}^*, \Theta) \in \mathcal{H} \Big\}.$$

- The *Negative Underivability* Σ^{c}, as a theory in propositional logic for capturing all forbidden accessability relations that are defined implicitly by the "complement" of $\mathcal{H}$ (in some sense), in their negated forms:

$$\Sigma^{\mathsf{c}} = \Big\{\neg\big([\mathbf{W}] \Rightarrow \mathsf{V}^*\big) \mid \Sigma \nvdash [\mathbf{W}] \Rightarrow \mathsf{V}^*,\ \mathbf{W} \subseteq \mathcal{V},\ \mathsf{V}^* \in \mathcal{V}\Big\}.$$

  Furthermore, we say that Δ is a *relaxed* negative underivability if $\Delta \subseteq \Sigma^{\mathsf{c}}$.

We call (Σ, Δ) a derivability structure pair if $\Delta \subseteq \Sigma^{\mathsf{c}}$. We also call $(\Sigma, \Sigma^{\mathsf{c}})$ complete. Intuitively, Σ describes completely the syntax of DAGE, while Σ^{c} (resp., $\Delta \subseteq \Sigma^{\mathsf{c}}$) describes completely the security notion (resp., relaxed security notion) of DAGE.

4.2.1 Syntax of DAGE

Definition 4.1 (Syntax of $\mathcal{G}$-DAGE in General Formulation)**.** A $\mathcal{G}$-DAGE scheme is specified by six polynomial-time randomized algorithms $\mathsf{GlobSetup}$, Setup, $\mathsf{Extract}$, $\mathsf{Combine}$, $\mathsf{Encrypt}$, and $\mathsf{Decrypt}$ as follows.

$\mathsf{GlobSetup}(1^\lambda) \to (\mathsf{gpk})$**:** Takes as input a security parameter 1^λ. It outputs a global public key gpk.

$\mathsf{Setup}(1^\lambda, \mathsf{gpk}, \mathsf{V}) \to (\mathsf{pk}_\mathsf{V}, \mathsf{sk}_\mathsf{V})$**:** Takes as input a security parameter 1^λ, the global public key gpk, and a *source node* $\mathsf{V} \in \mathcal{V}$ (a node of indegree 0). It outputs a public key pk_V and a secret key sk_V of node V. This algorithm takes place at only each source node to produce each corresponding public key.

$\mathsf{Extract}(\mathsf{V}, \mathsf{V}^*, \Gamma, \mathsf{pk}(\mathsf{V}^*), \mathsf{sk}_\mathsf{V}) \to (\mathsf{sk}_{\Gamma,(\mathsf{V}\rightarrowtail\mathsf{V}^*)})$**:** Takes as input a start node V, a target node V^* such that $\mathsf{V} \prec_{\mathsf{c}} \mathsf{V}^*$, an access structure Γ such that $\vdash [\![\Gamma]\!] \Rightarrow [\![\Theta_{\mathsf{V}^*}]\!]$, the set of all public keys at predecessor source nodes $\mathsf{pk}(\mathsf{V}^*)$ defined as $(\mathsf{gpk}, \{(\mathsf{W}, \mathsf{pk}_\mathsf{W}) \mid \mathsf{W} \preccurlyeq \mathsf{V}^*, \mathsf{indeg}(\mathsf{W}) = 0\})$, and the secret key sk_V. It outputs a *partial secret key* $\mathsf{sk}_{\Gamma,(\mathsf{V}\rightarrowtail\mathsf{V}^*)}$. This algorithm takes place at node V and the output will be given to node V^* in order to be used in the $\mathsf{Combine}$.

$\mathsf{Combine}(\mathsf{V}^*, \Gamma, \mathsf{pk}(\mathsf{V}^*), S_{\Gamma,\mathsf{V}^*,\mathbf{v}}) \to (\mathsf{sk}_{\mathsf{V}^*})$**:** Takes as the target node V^*, an access structure Γ such that $\vdash [\![\Gamma]\!] \Rightarrow [\![\Theta_{\mathsf{V}^*}]\!]$, the public key set $\mathsf{pk}(\mathsf{V}^*)$, and a set of partial secret keys $S_{\Gamma,\mathsf{V}^*,\mathbf{v}} = (\mathsf{sk}_{\Gamma,(\mathsf{V}\rightarrowtail\mathsf{V}^*)})_{\mathsf{V}\in\mathbf{v}}$ such that for all $\mathsf{V} \in \mathbf{V}$, $\mathsf{V} \prec_{\mathsf{c}} \mathsf{V}^*$ and that $\vdash [\mathbf{V}] \Rightarrow [\![\Gamma]\!]$. (Note that such $\mathbf{V}$ is not necessarily unique). It outputs the secret key $\mathsf{sk}_{\mathsf{V}^*}$ of node V^*. This algorithm takes place at node V^* to combine a legitimate set of partial secret keys so as to produce a secret key at node V^*.

$\mathsf{Encrypt}(\mathsf{V}, \mathsf{pk}(\mathsf{V}), M) \to (C)$**:** Takes as input a recipient node V, the public key set $\mathsf{pk}(\mathsf{V})$ corresponding to V, and a message M. It outputs the ciphertext C.

$\mathsf{Decrypt}(\mathsf{V}, \mathsf{pk}(\mathsf{V}), \mathsf{sk}_\mathsf{V}, C) \to (M)$**:** Take as input a recipient node V, the public key set $\mathsf{pk}(\mathsf{V})$ corresponding to V, the secret key sk_V, and the ciphertext C. It outputs the message M or a special symbol $\perp$ indicating an error.

Correctness. We define the correctness property of a $\mathcal{G}$-DAGE scheme as follows. Let $\mathsf{GlobSetup}(1^\lambda) \to (\mathsf{gpk})$ and for each source node V, let $\mathsf{Setup}(1^\lambda, \mathsf{gpk}, \mathsf{V}) \to (\mathsf{pk}_\mathsf{V}, \mathsf{sk}_\mathsf{V})$. The correctness holds if the following hold.

1. For all non-source node $\mathsf{V}^* \in \mathcal{V}$, we have that for all choices of $\Gamma, \mathbf{V}$ up to constraint that for all $\mathsf{V} \in \mathbf{V}$, $\mathsf{V} \prec_{\mathsf{c}} \mathsf{V}^*$ and that $\vdash [\![\Gamma]\!] \Rightarrow [\![\Theta_{\mathsf{V}^*}]\!]$ and $\vdash [\mathbf{V}] \Rightarrow [\![\Gamma]\!]$, the output $\mathsf{sk}_{\mathsf{V}^*}$ from the following process have the same distribution.

$$\begin{aligned}&\text{For all } \mathsf{V} \in \mathbf{V},\ \mathsf{Extract}(\mathsf{V}, \mathsf{V}^*, \Gamma, \mathsf{pk}(\mathsf{V}^*), \mathsf{sk}_{\mathsf{V}}) \to (\mathsf{sk}_{\Gamma,(\mathsf{V}\rightarrowtail\mathsf{V}^*)});\\ &\mathsf{Combine}(\mathsf{V}^*, \Gamma, \mathsf{pk}(\mathsf{V}^*), (\mathsf{sk}_{\Gamma,(\mathsf{V}\rightarrowtail\mathsf{V}^*)})_{\mathsf{V}\in\mathbf{V}}) \to (\mathsf{sk}_{\mathsf{V}^*}).\end{aligned}$$

2. For all $\mathsf{V} \in \mathcal{V}$, if $\mathsf{pk}(\mathsf{V})$ and $\mathsf{sk}_{\mathsf{V}^*}$ are correctly generated, then for all M,

$$\mathsf{Decrypt}\big(\mathsf{V}, \mathsf{pk}(\mathsf{V}), \mathsf{sk}_{\mathsf{V}},\ \mathsf{Encrypt}(\mathsf{V}, \mathsf{pk}(\mathsf{V}), M)\big) = M.$$

Remark 4.2. Definition 4.1 gives much generality, and thus is quite complex. We now discuss some special cases or how to set some default so as to lessen its complexity.

- For a special case when there is only one source node in $\mathcal{G}$, we collapse the algorithm $\mathsf{GlobSetup}$ and will not use gpk.
- For a special case when Θ_{V^*} is the collection of singleton sets (OR-structure), it is sufficient to set by default in $\mathsf{Extract}$ that $\Gamma = \{\{\mathsf{V}\}\}$. This also implies that in $\mathsf{Combine}$, $\mathbf{V} = \{\mathsf{V}\}$ is sufficient. Moreover, in a monotone OR-graph, we collapse the algorithm $\mathsf{Combine}$ and simply let $\mathsf{Extract}$ output $\mathsf{sk}_{\mathsf{V}^*}$ (instead of a partial key, $\mathsf{sk}_{\Gamma,(\mathsf{V}\rightarrowtail\mathsf{V}^*)}$).
- For a special case when Θ_{V^*} is the collection of one set (AND-structure), it is necessary (and sufficient) to set by default in $\mathsf{Extract}$ that $\Gamma = \Theta_{\mathsf{V}^*}$. This also implies that in $\mathsf{Combine}$, $\{\mathbf{V}\} = \Theta_{\mathsf{V}^*}$ is necessary (and sufficient).
- For general cases, in one extreme, the scheme may set default in $\mathsf{Extract}$ for Γ to be a single conjunctive access structure (*i.e.,* the collection of one set). In such a case, $\mathbf{V}$ is a unique set where $\Gamma = \{\mathbf{V}\}$.
- For general cases, in the other extreme, it may set default as $\Gamma = \Theta_{\mathsf{V}^*}$. In this case, $\mathsf{Extract}$ can be done "obliviously", *i.e.,* without the information on which access structure will really be used for deriving secret key for V^*.

The default in the last case above indeed turns out to be sufficient to capture almost all useful primitives, we thus make its explicit definition as follows. From now on, we will also confine ourselves to this formulation unless otherwise specified.

Definition 4.3 (Syntax of $\mathcal{G}$-DAGE in Oblivious Extraction Formulation)**.** A $\mathcal{G}$-DAGE (in oblivious extraction formulation) is specified by six polynomial-time randomized algorithms $\mathsf{GlobSetup}$, Setup, $\mathsf{Extract}$, $\mathsf{Combine}$, $\mathsf{Encrypt}$, and $\mathsf{Decrypt}$ with all algorithms except $\mathsf{Extract}$, $\mathsf{Combine}$ are defined exactly as in Definition 4.1. The remaining two are defined as follows.

$\mathsf{Extract}(\mathsf{V}, \mathsf{V}^*, \mathsf{pk}(\mathsf{V}^*), \mathsf{sk}_{\mathsf{V}}) \to (\mathsf{sk}_{(\mathsf{V}\rightarrowtail\mathsf{V}^*)})$**:** Takes as input a start node V, a target node V^* such that $\mathsf{V} \prec_{\mathsf{c}} \mathsf{V}^*$, the set of all public keys at predecessor source nodes $\mathsf{pk}(\mathsf{V}^*)$, and the secret key sk_{V}. It outputs a partial secret key $\mathsf{sk}_{(\mathsf{V}\rightarrowtail\mathsf{V}^*)}$.

$\mathsf{Combine}(\mathsf{V}^*, \mathsf{pk}(\mathsf{V}^*), S_{\mathsf{V}^*,\mathbf{V}}) \to (\mathsf{sk}_{\mathsf{V}^*})$**:** Takes as the target node V^*, the public key set $\mathsf{pk}(\mathsf{V}^*)$, and a set of partial secret keys $S_{\mathsf{V}^*,\mathbf{V}} = (\mathsf{sk}_{(\mathsf{V}\rightarrowtail\mathsf{V}^*)})_{\mathsf{V}\in\mathbf{V}}$ such that for all $\mathsf{V} \in \mathbf{V}$, $\mathsf{V} \prec_{\mathsf{c}} \mathsf{V}^*$ and that $\vdash [\mathbf{V}] \Rightarrow [\![\Theta_{\mathsf{V}^*}]\!]$. (Note that such $\mathbf{V}$ is not necessarily unique). It outputs the secret key $\mathsf{sk}_{\mathsf{V}^*}$ of node V^*.

4.2.2 Security Notions for DAGE

Message Indistinguishability Game. The negative underivability structure Σ^{c} of $\mathcal{G}$ naturally "captures" security aspects for a $\mathcal{G}$-DAGE since intuitively it contains all the requirements about key derivation that adversary should not be able to do in order to say that the scheme is secure. We now formalize this intuition in the standard way by defining some appropriate oracle access in the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Both $\mathcal{C}$ and $\mathcal{A}$ are given $\mathcal{G}$ as input.

Setup. The challenger $\mathcal{C}$ runs $\mathsf{GlobSetup}(1^\lambda) \to (\mathsf{gpk})$. It then gives gpk to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ adaptively issues queries where each is one of the following.

- Public key query (V) : '$\mathsf{pk}_{\mathsf{V}} =?$'. $\mathcal{C}$ responds by giving $\mathcal{A}$ the pk_{V}. If it is not defined yet, $\mathcal{C}$ runs $\mathsf{Setup}(1^\lambda, \mathsf{gpk}, \mathsf{V}) \to (\mathsf{pk}_{\mathsf{V}}, \mathsf{sk}_{\mathsf{V}})$ and gives pk_{V} to $\mathcal{A}$.
- Partial secret key query $(\mathsf{V}, \mathsf{V}^*)$: '$\mathsf{sk}_{(\mathsf{V} \rightarrowtail \mathsf{V}^*)} =?$'. $\mathcal{C}$ responds by giving $\mathcal{A}$ the $\mathsf{sk}_{(\mathsf{V} \rightarrowtail \mathsf{V}^*)}$. If it is not defined yet, $\mathcal{C}$ appropriately runs Setup, $\mathsf{Extract}$, $\mathsf{Combine}$ (but will use those already-defined terms as some inputs) to obtain $\mathsf{sk}_{(\mathsf{V} \rightarrowtail \mathsf{V}^*)}$ and gives it to $\mathcal{A}$.
- Secret key query (V) : '$\mathsf{sk}_{\mathsf{V}} =?$'. $\mathcal{C}$ responds by giving $\mathcal{A}$ the sk_{V}. If it is not defined yet, $\mathcal{C}$ appropriately runs Setup, $\mathsf{Extract}$, $\mathsf{Combine}$ (but will use those already-defined terms as some inputs) to obtain sk_{V} and gives it to $\mathcal{A}$.
- Decryption query (V, C) : '$\mathsf{Decrypt}(\mathsf{V}, \mathsf{pk}(\mathsf{V}), \mathsf{sk}_{\mathsf{V}}, C) =?$'. $\mathcal{C}$ responds by giving $\mathcal{A}$ the result of $\mathsf{Decrypt}(\mathsf{V}, \mathsf{pk}(\mathsf{V}), \mathsf{sk}_{\mathsf{V}}, C)$.

During this phase, $\mathcal{C}$ maintains the list of queries that have been asked for each oracle type: $\mathsf{L}_{\mathsf{pub}}, \mathsf{L}_{\mathsf{par}}, \mathsf{L}_{\mathsf{sec}}, \mathsf{L}_{\mathsf{dec}}$ respectively. We also denote by Der the set of secret keys that the adversary can trivially derive, *i.e.,*

$$\mathsf{Der} = \mathsf{L}_{\mathsf{sec}} \cup \left\{ \mathsf{V}^* \;\middle|\; \begin{array}{l} \text{there exists } \mathbf{V} \text{ such that } \vdash [\mathbf{V}] \Rightarrow [\![\Theta_{\mathsf{V}^*}]\!] \\ \text{and that for all } \mathsf{V} \in \mathbf{V} : \ (\mathsf{V}, \mathsf{V}^*) \in \mathsf{L}_{\mathsf{par}} \end{array} \right\}.$$

Since this is depended on and only on $\mathsf{L}_{\mathsf{sec}}, \mathsf{L}_{\mathsf{par}}$ we may write $\mathsf{Der}(\mathsf{L}_{\mathsf{sec}}, \mathsf{L}_{\mathsf{par}})$.

Challenge. Once $\mathcal{A}$ decides that Phase 1 is over, it outputs two messages $M_0, M_1 \in \mathcal{M}$ of equal length and a target node $\mathsf{V}^\star$ which it intends to attack. The only restriction is that

$$\Delta \vdash \neg\big(\, [\mathsf{Der}] \Rightarrow \langle\!\langle \mathsf{V}^\star \rangle\!\rangle \, \big).$$

The challenger $\mathcal{C}$ then picks a random bit $b \in \{0, 1\}$ and sets $C^\star = \mathsf{Encrypt}(\mathsf{V}^\star, \mathsf{pk}(\mathsf{V}^\star), M_b)$. It sends $C^\star$ as a challenge to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ issues additional queries where each is one of the following. $\mathcal{C}$ maintains the four lists as usual.

- Public key query (V) : '$\mathsf{pk}_{\mathsf{V}} =?$'. $\mathcal{C}$ responds as usual.
- Partial secret key query $(\mathsf{V}, \mathsf{V}^*)$: '$\mathsf{sk}_{(\mathsf{V} \rightarrowtail \mathsf{V}^*)} =?$'. Let $\mathsf{L}_{\mathsf{sec}}, \mathsf{L}_{\mathsf{par}}$ be the lists before recording this query. $\mathcal{C}$ responds as in phase 1 if and only if either:

$$\Delta \vdash \neg\big(\, [\mathsf{Der}(\mathsf{L}_{\mathsf{sec}}, \mathsf{L}_{\mathsf{par}})] \wedge \langle\!\langle \mathsf{V}^* \rangle\!\rangle \Rightarrow \langle\!\langle \mathsf{V}^\star \rangle\!\rangle \, \big) \quad \text{or} \quad \mathsf{Der}(\mathsf{L}_{\mathsf{sec}}, \mathsf{L}_{\mathsf{par}} \cup \{(\mathsf{V}, \mathsf{V}^*)\}) = \mathsf{Der}(\mathsf{L}_{\mathsf{sec}}, \mathsf{L}_{\mathsf{par}}).$$

- Secret key query (V) : 'sk_{V} =?'. Let $\mathsf{L}_{\mathsf{sec}}, \mathsf{L}_{\mathsf{par}}$ be the lists before recording this query. $\mathcal{C}$ responds as in phase 1 if and only if

$$\Delta \vdash \neg\big(\, [\mathsf{Der}(\mathsf{L}_{\mathsf{sec}}, \mathsf{L}_{\mathsf{par}})] \wedge \langle\!\langle \mathsf{V} \rangle\!\rangle \Rightarrow \langle\!\langle \mathsf{V}^{\star} \rangle\!\rangle \, \big).$$

- Decryption query (V, C) : '$\mathsf{Decrypt}(\mathsf{V}, \mathsf{pk}(\mathsf{V}), \mathsf{sk}_{\mathsf{V}}, C)$ =?'. $\mathcal{C}$ responds as in phase 1 if and only if $(\mathsf{V}, C) \neq (\mathsf{V}^{\star}, C^{\star})$.

Guess Finally $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$ for b and wins the game if $b = b'$.

We refer to such an adversary $\mathcal{A}$ as an IND-$(\mathsf{a}, \mathsf{f}|\ \mathcal{G}, \Delta)$-$\mathsf{CCA}$ adversary and the above game as the IND-$(\mathsf{a}, \mathsf{f}|\ \mathcal{G}, \Delta)$-$\mathsf{CCA}$ game. In the following, we will consider its variations as IND-$(\mathsf{x}, \mathsf{y}|\ \mathcal{G}, \Delta)$-$\mathsf{Z}$ notion where

- $\mathsf{x} \in \{\mathsf{a}, \mathsf{s}\}$ for adaptive/selective target node attack,
- $\mathsf{y} \in \{\mathsf{f}, \mathsf{n}\}$ for full/normal secret key exposure attack,
- $\mathsf{Z} \in \{\mathsf{CCA}, \mathsf{CPA}\}$ for chosen-ciphertext/chosen-plaintext attack.

For the 'x' dimension, a weaker notion of security can be defined by modifying the above game so that it is exactly the same but except only that it is also required that the adversary must disclose the target node $\mathsf{V}^{\star}$ before the Setup phase. We call such a notion as IND-$(\mathsf{s}, \mathsf{f}|\ \mathcal{G}, \Delta)$-$\mathsf{CCA}$. This notion is analogous to the notion of selective-identity secure HIBE, given by Canetti, Halevi, and Katz [CHK03, CHK04]. Note that it is also required that the restrictions on queries from phase 2 also hold in phase 1.

We define the advantage of the adversary $\mathcal{A}$ in attacking the $\mathcal{G}$-DAGE scheme $\mathcal{E}$ in the game IND-$(\mathsf{x}, \mathsf{f}|\ \mathcal{G}, \Delta)$-$\mathsf{CCA}$ as $\mathsf{Adv}_{(\mathcal{G},\Delta),\mathsf{x}}(\mathcal{E}, \mathcal{A}) = |\Pr[b = b'] - \frac{1}{2}|$, where the probability is over the random bits used by $\mathcal{C}$ and $\mathcal{A}$ in that game.

Definition 4.4 (Δ-Security of $\mathcal{G}$-DAGE)**.** We say that a $\mathcal{G}$-DAGE system $\mathcal{E}$ is $(t, \epsilon, q_{\mathsf{pub}}, q_{\mathsf{par}}, q_{\mathsf{sec}}, q_{\mathsf{dec}})$-$\mathsf{IND}$-$(\mathsf{x}, \mathsf{f}|\ \mathcal{G}, \Delta)$-$\mathsf{CCA}$-secure if for any t-time IND-$(\mathsf{x}, \mathsf{f}|\ \mathcal{G}, \Delta)$-$\mathsf{CCA}$ adversary $\mathcal{A}$ that makes at most q_{oracle} queries to oracle, we have that $\mathsf{Adv}_{(\mathcal{G},\Delta),\mathsf{x}}(\mathcal{E}, \mathcal{A}) < \epsilon$. Moreover, two deviations are defined as follows

- When $q_{\mathsf{dec}} = 0$, then we call that $\mathcal{E}$ is $(t, \epsilon, q_{\mathsf{pub}}, q_{\mathsf{par}}, q_{\mathsf{sec}})$-$\mathsf{IND}$-$(\mathsf{x}, \mathsf{f}|\ \mathcal{G}, \Delta)$-$\mathsf{CPA}$-secure.
- When $q_{\mathsf{par}} = 0$, then we call that $\mathcal{E}$ is $(t, \epsilon, q_{\mathsf{pub}}, q_{\mathsf{sec}}, q_{\mathsf{dec}})$-$\mathsf{IND}$-$(\mathsf{x}, \mathsf{n}|\ \mathcal{G}, \Delta)$-$\mathsf{CCA}$-secure.

Note that their combinations can also be considered straightforwardly. Furthermore, when t and q_{oracle}'s are $O(\mathsf{poly}(\lambda))$ and ϵ is $\mathsf{negl}(\lambda)$ where λ is the security parameter then we simply say that $\mathcal{E}$ is IND-$(\mathsf{x}, \mathsf{y}|\ \mathcal{G}, \Delta)$-$\mathsf{Z}$-secure (for corresponding $\mathsf{x}, \mathsf{y}, \mathsf{Z}$).

4.2.3 Multiple Encryption on DAGE

We now introduce notions of multiple encryption on DAGE, which is an important ingredient for many applications of DAGE. The functionalities of multiple encryption can be stated as three add-on algorithms to previously defined DAGE as follows.

Definition 4.5 (Multiple-node DAGE)**.** A multiple-node DAGE scheme with a collection $\mathcal{P}$ of qualified access structure over $\mathcal{V}$ consists of nine algorithms where six of which are those of normal DAGEs and three add-on algorithms are defined as follows.

$\mathsf{MEncrypt}(\Omega, \mathsf{pk}(\Omega), M) \to (C)$**:** Takes as input a recipient access structure $\Omega \in \mathcal{P}$, the public key set corresponding to source ancestors of nodes in Ω, more formally defined as

$$\mathsf{pk}(\Omega) = \Big(\mathsf{gpk}, \bigcup_{\mathsf{V} \in (\bigcup_{\mathbf{V} \in \Omega} \mathbf{V})} \{(\mathsf{W}, \mathsf{pk}_{\mathsf{W}}) \mid \mathsf{W} \preccurlyeq \mathsf{V}, \mathsf{indeg}(\mathsf{W}) = 0\} \Big),$$

and a message M. It outputs the ciphertext C.

$\mathsf{MDecrypt}(\Omega, \mathsf{pk}(\Omega), \mathsf{V}, \mathsf{sk}_{\mathsf{V}}, C) \to (M_{\mathsf{V}})$**:** Take as input a recipient access structure Ω, the public key set $\mathsf{pk}(\Omega)$, a node V and its secret key sk_{V}, and the ciphertext C. It outputs a message share M_{V} or a special symbol $\perp$ indicating an error.

$\mathsf{Gather}(\Omega, \mathsf{pk}(\Omega), \mathbf{X}, (M_{\mathsf{V}})_{\mathsf{V} \in \mathbf{X}}) \to (M)$**:** Take as input a recipient access structure Ω, the public key set $\mathsf{pk}(\Omega)$, a set of nodes $\mathbf{X} \in \Omega$, the set of all message shares corresponding to each $\mathsf{V} \in \mathbf{X}$. It outputs a message M.

The correctness can be defined straight-forwardly as usual. We now define the security notions as follows. Such notions follow the general multiple encryption [DK05].

Game for Multiple-node DAGE. The game is exactly the same as that of normal DAGE, except for the points below.

Setup. Unchanged.

Phase 1. $\mathcal{A}$ adaptively issues some adaptive queries. The public key, partial private key, and secret key oracles are unchanged. Only the decryption oracle is generalized as folllows.

- Multiple decryption query (Ω, V, C) : '$\mathsf{MDecrypt}(\Omega, \mathsf{pk}(\Omega), \mathsf{V}, \mathsf{sk}_{\mathsf{V}}, C) =?$'. $\mathcal{C}$ responds by giving $\mathcal{A}$ the result of $\mathsf{MDecrypt}(\Omega, \mathsf{pk}(\Omega), \mathsf{V}, \mathsf{sk}_{\mathsf{V}}, C)$.
- Weak multiple decryption query $(\Omega, \mathbf{X}, C)$:

$$\text{'}\mathsf{Gather}\Big(\Omega, \mathsf{pk}(\Omega), \mathbf{X}, \big(\mathsf{MDecrypt}(\Omega, \mathsf{pk}(\Omega), \mathsf{W}, \mathsf{sk}_{\mathsf{W}}, C) \big)_{\mathsf{W} \in \mathbf{X}} \Big) =?\text{'},$$

  $\mathcal{C}$ computes it and returns to $\mathcal{A}$.

During this phase, $\mathcal{C}$ maintains the list of queries as usual. $\mathsf{L}_{\mathsf{mdec}}, \mathsf{L}_{\mathsf{wmdec}}$ are the lists of two above oracles.

Challenge. Once $\mathcal{A}$ decides that Phase 1 is over, it outputs two messages $M_0, M_1 \in \mathcal{M}$ of equal length and a target access structure $\Omega^\star$ which it intends to attack. The only restriction is that

$$\Delta \vdash \neg\big([\mathsf{Der}] \Rightarrow [\![\Omega^\star]\!] \big).$$

The challenger $\mathcal{C}$ then picks a random bit $b \in \{0, 1\}$ and sets $C^\star = \mathsf{MEncrypt}(\Omega^\star, \mathsf{pk}(\Omega^\star), M_b)$. It sends $C^\star$ as a challenge to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ issues additional queries. Only the public key oracle is unchanged.

- Partial secret key query $(\mathsf{V}, \mathsf{V}^*)$: '$\mathsf{sk}_{(\mathsf{V} \to \mathsf{V}^*)} =?$'. Let $\mathsf{L}_{\mathsf{sec}}, \mathsf{L}_{\mathsf{par}}$ be the lists before recording this query. $\mathcal{C}$ responds as in phase 1 if and only if either:

$$\Delta \vdash \neg\big([\mathsf{Der}(\mathsf{L}_{\mathsf{sec}}, \mathsf{L}_{\mathsf{par}})] \wedge \langle\!\langle \mathsf{V}^* \rangle\!\rangle \Rightarrow [\![\Omega^\star]\!] \big) \quad \text{or} \quad \mathsf{Der}(\mathsf{L}_{\mathsf{sec}}, \mathsf{L}_{\mathsf{par}} \cup \{(\mathsf{V}, \mathsf{V}^*)\}) = \mathsf{Der}(\mathsf{L}_{\mathsf{sec}}, \mathsf{L}_{\mathsf{par}}).$$

- Secret key query (V) : '$\mathsf{sk}_{\mathsf{V}} =?$'. Let $\mathsf{L}_{\mathsf{sec}}, \mathsf{L}_{\mathsf{par}}$ be the lists before recording this query. $\mathcal{C}$ responds as in phase 1 if and only if

$$\Delta \vdash \neg\big([\mathsf{Der}(\mathsf{L}_{\mathsf{sec}}, \mathsf{L}_{\mathsf{par}})] \wedge \langle\!\langle \mathsf{V} \rangle\!\rangle \Rightarrow [\![\Omega^\star]\!] \big).$$

- Multiple decryption query (Ω, V, C). $\mathcal{C}$ responds as in phase 1 if and only if

$$(\Omega, C) \neq (\Omega^\star, C^\star) \quad \text{or} \quad [\{\mathsf{W} \mid (\Omega^\star, \mathsf{W}, C^\star) \in \mathsf{L}_{\mathsf{mdec}}\}] \wedge \langle\!\langle \mathsf{V} \rangle\!\rangle \rightarrow [\![\Omega^\star]\!]$$

Guess Finally $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$ for b and wins the game if $b = b'$.

We use the same terminology as that of normal DAGEs except that one changes CPA to MCPA and CCA to wMCCA or MCCA notions. The latter, MCCA, refers to the notion exactly described in the game above, while the former, wMCCA, refers to weak multiple CCA where the adversary is not allowed to ask the multiple decryption query (only the weak multiple decryption query and the others are allowed). In conclusion, the notions $\mathsf{IND}\text{-}(\mathsf{x},\mathsf{y}|\ \mathcal{G}, \Delta)\text{-}\mathsf{Z}$ for $\mathsf{x} \in \{\mathsf{a},\mathsf{s}\}$, $\mathsf{y} \in \{\mathsf{f},\mathsf{n}\}$, and $\mathsf{Z} \in \{\mathsf{MCPA}, \mathsf{wMCCA}, \mathsf{MCCA}\}$ are defined completely.

Construction for Multiple-node DAGE. An approach for CCA-secure multiple encryption was considered in [ZHSI04, DK05]. Both are generic as it applies to any encryption primitives. While the scheme by [ZHSI04] is proven in the random oracle model, the scheme by [DK05] considered stronger notions and is provably secure without random oracle. Hence it is preferable to use the latter scheme. Since we will use multiple encryption in a black-box manner, we only briefly describe such method here.

Although the scheme of [DK05] can apply to any encryption primitive, it does not apply directly. Instead a tag-based version (or called label in [Sho01, DK05]) of such primitives is considered. Any CCA-secure encryption can be used to construct CCA-secure tag-based version of it by using, for example, hybrid encryption paradigm [Sho01]. We omit the detail of these notions here though. The construction of multiple-node DAGE is described below. We will use one-time signature scheme and a set of secret sharing schemes $\{\ (\mathsf{Share}_{|\mathsf{Sup}(\Omega)|,\Omega}, \mathsf{Recon}_{|\mathsf{Sup}(\Omega)|,\Omega}) \mid \Omega \in \mathcal{P}\}$ as components. $\mathsf{Encrypt}^{\mathsf{T}}(\cdot)$ is the encryption with tag T.

Construction 4-1. Multiple-node DAGE

$\mathsf{MEncrypt}(\Omega, \mathsf{pk}(\Omega), M) \rightarrow (C)$**:** Let $S = \bigcup_{\mathbf{V} \in \Omega} \mathbf{V}$ which we parse as $S = \{\mathsf{V}_1, \ldots, \mathsf{V}_n\}$. Let $(s_{\mathsf{V}_1}, \ldots, s_{\mathsf{V}_n}, \mathsf{pub}) \leftarrow \mathsf{Share}_{n,\Omega}(M)$, and generate $(V_{\mathsf{SIG}}, K_{\mathsf{SIG}}) \leftarrow \mathsf{Gen}(1^\lambda)$. Set $C_i = \mathsf{Encrypt}^{(V_{\mathsf{SIG}})}(\mathsf{V}_i, \mathsf{pk}(\mathsf{V}_i), s_i)$. Then compute signature $\sigma = \mathsf{Sign}_{K_{\mathsf{SIG}}}(C_1, \ldots, C_n, \mathsf{pub})$. Output $C = (C_1, \ldots, C_n, \mathsf{pub}, V_{\mathsf{SIG}}, \sigma)$.

$\mathsf{MDecrypt}(\Omega, \mathsf{pk}(\Omega), \mathsf{V}, \mathsf{sk}_{\mathsf{V}}, C) \rightarrow (M_{\mathsf{V}})$**:** Parse $C = (C_1, \ldots, C_n, \mathsf{pub}, V_{\mathsf{SIG}}, \sigma)$. Then check whether $\mathsf{Vrfy}_{V_{\mathsf{SIG}}}((C_1, \ldots, C_n, \mathsf{pub}), \sigma) = 1$. If the verification fails, just output $\perp$. Note that it must be that $\mathsf{V} = \mathsf{V}_i$ for some i. Compute $s_{\mathsf{V}} = \mathsf{Decrypt}^{(V_{\mathsf{SIG}})}(\mathsf{V}_i, \mathsf{pk}(\mathsf{V}_i), \mathsf{sk}_{\mathsf{V}_i}, C_i)$ and output $M_{\mathsf{V}} = (s_{\mathsf{V}}, \mathsf{pub})$.

$\mathsf{Gather}(\Omega, \mathsf{pk}(\Omega), \mathbf{X}, (M_{\mathsf{V}})_{\mathsf{V} \in \mathbf{X}}) \rightarrow (M)$**:** Output $M = \mathsf{Recon}_{n,\Omega}(\mathbf{X}, (s_{\mathsf{V}})_{\mathsf{V} \in \mathbf{X}}, \mathsf{pub})$.

Theorem 4.6. *Suppose that the underlying DAGE scheme is* IND-$(\mathsf{x},\mathsf{y}|\ \mathcal{G},\Delta)$-$\mathsf{Z}$*-secure for* $\mathsf{x}\in\{\mathsf{a},\mathsf{s}\}$, $\mathsf{y}\in\{\mathsf{f},\mathsf{n}\}$, $\mathsf{Z}\in\{\mathsf{CPA},\mathsf{CCA}\}$. *Then the above multiple-node DAGE scheme is* IND-$(\mathsf{x},\mathsf{y}|\ \mathcal{G},\Delta)$-$\mathsf{MZ}$*-secure.*

Note that the proof follows almost directly from [DK05], albeit adapted to DAGE appropriately.

4.2.4 Definition for Class of ID-based Graphs

From any LAD $G=(V,H)$, we are interested in constructing its powerful extension where each node in V can be assigned arbitrary string variable so that the new LAD will consist of nodes where each is specified by its "position" which is exactly the label of node in G and, in addition, its "variable" which is assigned by this extension. More precisely, a node in the new LAD will be of the form $\left(v,(\mathsf{id}_w)_{w\preccurlyeq v}\right)$, *i.e.,* it consists of the position V and the ID variables for all of its ancestors. The formal definition is given below.

Definition 4.7. Let $G=(V,H)$ be a LAD. Its ID-based graph extension $\mathcal{G}_{\mathsf{Id}\mathsf{x}(G)}=(\mathcal{V},\mathcal{H})$ defined from the base LAD G is given by

$$\mathcal{V}=\left\{\ \left(v,(\mathsf{id}_w)_{w\preccurlyeq v}\right)\ |\ v\in V;\ \forall w\preccurlyeq v,\ \text{if } w\in V^0 \text{ then } \mathsf{id}_w=\varepsilon \text{ else } \mathsf{id}_w\in\mathcal{I}\right\}$$

$$\mathcal{H}=\left\{\ \left\{\left(v_1,(\mathsf{id}_w)_{w\preccurlyeq v_1}\right),\ldots,\left(v_j,(\mathsf{id}_w)_{w\preccurlyeq v_j}\right)\right\}\xrightarrow{\Theta\otimes_{\mathsf{Id}\mathsf{x}}(\mathsf{id}_w)_{w\preccurlyeq v^*}}\left(v^*,(\mathsf{id}_w)_{w\preccurlyeq v^*}\right)\right.$$
$$\left.\ \Big|\ \left(\{v_1,\ldots,v_j\}\xrightarrow{\Theta}v^*\right)\in H;\ \forall w\preccurlyeq v^*,\ \text{if } w\in V^0 \text{ then } \mathsf{id}_w=\varepsilon \text{ else } \mathsf{id}_w\in\mathcal{I}\right\}.$$

where $\Theta\otimes_{\mathsf{Id}\mathsf{x}}(\mathsf{id}_w)_{w\preccurlyeq v^*}=\left\{\left\{\left(v_{i_1},(\mathsf{id}_w)_{w\preccurlyeq v_{i_1}}\right),\ldots,\left(v_{i_k},(\mathsf{id}_w)_{w\preccurlyeq v_{i_k}}\right)\right\}\ |\ \{v_{i_1},\ldots,v_{i_k}\}\in\Theta\right\}$.

All of our DAGE constructions are built for ID-based graphs for generality. Constructions for normal non-ID-based graphs are specific cases where the ID variables are fixed in an appropriate way.

4.3 Graph Syntactic Consequence

Typically, the implication among primitives can be shown by first constructing one primitive from black-box usage of another primitive then proving that the security of the base primitive is reduced to the security of the resulting construction (in either black-box fashion or not). Such a reduction proof must be reconstructed for every pair of primitives and is usually complex in the context of computational complexity. It is thus preferable to have a criterion which is relatively easier to justify for any pair of primitives but yet implies the full implication, as described above. We obtain such a criterion for primitives that can be casted as DAGE as the notion we call graph syntactic consequence. Somewhat surprisingly, this notion can be described purely in the context of propositional logic. Therefore, such proofs of relations among primitives can be made logically formal, and hence can be automated verified. This section will elaborate the definition and its main theorem with the proof.

For a set S, let $\mathcal{AS}^{(S)}$ denote the collection of all privileged access structures over S. Let $\mathcal{V}_{\mathcal{G}}^0$ be the set of source nodes in $\mathcal{G}$.

Definition 4.8. Let $(\hat{\Sigma}, \hat{\Delta})$ (resp., (Σ, Δ)) be the positive derivability and a relaxed negative underivability of $\hat{\mathcal{G}}$ (resp., $\mathcal{G}$). We say that a tuple $(\hat{\mathcal{G}}, \hat{\Sigma}, \hat{\Delta})$ *graph syntactically implies* $(\mathcal{G}, \Sigma, \Delta)$ and denote it by $(\hat{\mathcal{G}}, \hat{\Sigma}, \hat{\Delta}) \Vdash_{\mathsf{syn}} (\mathcal{G}, \Sigma, \Delta)$ if there exists injective mappings

$$\sigma : \mathcal{V}_{\mathcal{G}} \to \mathcal{AS}^{(\mathcal{V}_{\hat{\mathcal{G}}})},$$
$$\delta : \mathcal{V}_{\mathcal{G}} \to \mathcal{AS}^{(\mathcal{V}_{\hat{\mathcal{G}}})}$$

with two simple prior requirements:

- for all $\mathsf{V} \in \mathcal{V}_{\mathcal{G}}$, we have $\vdash [\![\sigma(\mathsf{V})]\!] \Rightarrow [\![\delta(\mathsf{V})]\!]$,
- for all $\mathsf{V} \in \mathcal{V}^0_{\mathcal{G}}$, we have $\sigma(\mathsf{V}) \in \mathcal{AS}^{(\mathcal{V}^0_{\hat{\mathcal{G}}})}$ and its minimal representation $\sigma(\mathsf{V})^{(\mathsf{min})}$ is the collection of one set (*i.e.,* $\sigma(\mathsf{V})$ is an AND-structure); furthermore, write $\sigma(\mathsf{V})^{(\mathsf{min})} = \{\hat{\mathbf{X}}\}$, $\sigma(\mathsf{W})^{(\mathsf{min})} = \{\hat{\mathbf{Y}}\}$, it is required that $\hat{\mathbf{X}} \cap \hat{\mathbf{Y}} = \emptyset$ for different $\mathsf{V}, \mathsf{W} \in \mathcal{V}^0_{\mathcal{G}}$;

such that the two following main properties hold:

1. $\hat{\Sigma} \vdash \Sigma|_{(\sigma)}$,
2. $\hat{\Delta} \vdash \Delta|_{(\sigma,\delta)}$.

where we define two *substitution* procedures as follows. Let $[\mathbf{V}]|_\sigma = \bigwedge_{\mathsf{V} \in \mathbf{V}} [\![\sigma(\mathsf{V})]\!]$ and let

$$\Sigma|_{(\sigma)} = \left\{ \quad [\mathbf{V}]|_\sigma \Rightarrow [\![\sigma(\mathsf{V}^*)]\!] \quad \middle| \quad \Sigma \vdash \quad [\mathbf{V}] \Rightarrow \langle\!\langle \mathsf{V}^* \rangle\!\rangle \quad \right\},$$
$$\Delta|_{(\sigma,\delta)} = \left\{ \neg([\mathbf{V}]|_\sigma \Rightarrow [\![\delta(\mathsf{V}^*)]\!]) \;\middle|\; \Delta \vdash \neg([\mathbf{V}] \Rightarrow \langle\!\langle \mathsf{V}^* \rangle\!\rangle) \right\}.$$

Furthermore we also abuse the notation by denoting $(\hat{\mathcal{G}}, \hat{\Sigma}, \hat{\Sigma}^{\mathsf{c}}) \Vdash_{\mathsf{syn}} (\mathcal{G}, \Sigma, \Sigma^{\mathsf{c}})$ by just $\hat{\mathcal{G}} \Vdash_{\mathsf{syn}} \mathcal{G}$ as its shorthand. (Recall that indeed $\mathcal{G}$ completely defines $\Sigma_{\mathcal{G}}$ and hence also $\Sigma^{\mathsf{c}}_{\mathcal{G}}$).

Theorem 4.9 (The First Primitive Implication Theorem or The Soundness Theorem for Graph Syntactic Consequence)**.** *Suppose that* $(\hat{\mathcal{G}}, \hat{\Sigma}, \hat{\Delta}) \Vdash_{\mathsf{syn}} (\mathcal{G}, \Sigma, \Delta)$. *Then we have a black-box construction from* $\mathsf{IND}\text{-}(\mathsf{x}, \mathsf{n}|\ \hat{\mathcal{G}}, \hat{\Delta})\text{-}\mathsf{Z}$*-secure* $\hat{\mathcal{G}}$*-DAGE to* $\mathsf{IND}\text{-}(\mathsf{x}, \mathsf{n}|\ \mathcal{G}, \Delta)\text{-}\mathsf{Z}$*-secure* $\mathcal{G}$*-DAGE for* $\mathsf{x} \in \{\mathsf{a}, \mathsf{s}\}$, $\mathsf{Z} \in \{\mathsf{CPA}, \mathsf{CCA}\}$.

The above theorem indeed is constructive in the sense that we can describe the construction explicitly. To state such a construction, we first give two lemmas as follows.

Lemma 4.10. *Suppose that* $(\hat{\mathcal{G}}, \hat{\Sigma}, \hat{\Delta}) \Vdash_{\mathsf{syn}} (\mathcal{G}, \Sigma, \Delta)$ *via mappings* σ, δ *(as in Definition 4.8). Then for all* $\mathbf{V} \subseteq \{\mathsf{V} \in \mathcal{V}_{\mathcal{G}} \mid \mathsf{V} \prec_{\mathsf{c}} \mathsf{V}^*\}$ *such that* $\vdash [\mathbf{V}] \Rightarrow [\![\Theta_{\mathsf{V}^*}]\!]$, *let* $\mathbf{V} = \{\mathsf{V}_1, \ldots, \mathsf{V}_t\}$, *we have that for all* $\left(\hat{\mathbf{W}}_1, \ldots, \hat{\mathbf{W}}_t \right) \in \sigma(\mathsf{V}_1) \times \cdots \times \sigma(\mathsf{V}_t)$, *there exists* $\hat{\mathbf{Y}} \in \sigma(\mathsf{V}^*)^{(\mathsf{min})}$ *such that*

$$\hat{\Sigma} \vdash \bigwedge_{i=1}^{t} [\hat{\mathbf{W}}_i] \Rightarrow [\hat{\mathbf{Y}}].$$

Lemma 4.11. *Suppose that there exists a* $\mathcal{G}$*-DAGE scheme* $\mathcal{E}$. *Then the following polynomial-time randomized algorithms can be constructed in a black-box manner from* $\mathcal{E}$.

XExtract**:** *Takes as inputs* $\mathbf{Z} = \{\mathsf{Z}_1, \ldots, \mathsf{Z}_k\}, \mathsf{Y}, (\mathsf{Z}_j, \mathsf{sk}_{\mathsf{Z}_j})$ *such that* $\Sigma_{\mathcal{G}} \vdash \langle\!\langle \mathsf{Z}_1 \rangle\!\rangle \wedge \cdots \wedge \langle\!\langle \mathsf{Z}_k \rangle\!\rangle \Rightarrow \langle\!\langle \mathsf{Y} \rangle\!\rangle$ *and* $1 \leq j \leq k$. *It outputs* $\mathsf{tk}_{\mathbf{Z},(\mathsf{Z}_j \rightarrowtail \mathsf{Y})}$.

$\mathsf{XCombine}$**:** *Takes as inputs* $\mathbf{Z}, \mathsf{Y}, (\mathsf{tk}_{\mathbf{Z},(\mathsf{Z}_j \rightarrowtail \mathsf{Y})})_{\mathsf{Z}_j \in \mathbf{Z}}$*. It outputs* sk_{Y}*.*

Now we can state the generic construction. Suppose that $(\hat{\mathcal{G}}, \hat{\Sigma}, \hat{\Delta}) \Vdash_{\mathsf{syn}} (\mathcal{G}, \Sigma, \Delta)$ via the maps (σ, δ) and that we have a construction of $\hat{\mathcal{G}}$-DAGE, say the concrete scheme $\hat{\mathcal{E}} = (\hat{\mathsf{GlobSetup}}, \hat{\mathsf{Setup}}, \hat{\mathsf{Extract}}, \hat{\mathsf{Combine}}, \hat{\mathsf{Encrypt}}, \hat{\mathsf{Decrypt}})$, which is extended to multiple-node DAGE with algorithms $\hat{\mathsf{Mencrypt}}, \hat{\mathsf{Mdecrypt}}, \hat{\mathsf{Gather}})$. Then we construct $\mathcal{G}$-DAGE which we denote by $\mathcal{G}$-$\mathsf{GenDAGE}(\hat{\mathcal{E}}, \hat{\mathcal{G}}, \sigma, \delta)$ as follows.

Construction 4-2. Generic Conversion from $\hat{\mathcal{G}}$-DAGE to $\mathcal{G}$-DAGE: $\mathcal{G}$-GenDAGE$(\hat{\mathcal{E}}, \hat{\mathcal{G}}, \sigma, \delta)$

$\mathsf{GlobSetup}(1^\lambda) \rightarrow (\mathsf{gpk})$**:** Run $\hat{\mathsf{GlobSetup}}(1^\lambda) \rightarrow (\hat{\mathsf{gpk}})$. Let $\mathsf{gpk} = \hat{\mathsf{gpk}}$ and outputs it.

$\mathsf{Setup}(1^\lambda, \mathsf{gpk}, \mathsf{V}) \rightarrow (\mathsf{pk}_{\mathsf{V}}, \mathsf{sk}_{\mathsf{V}})$**:** Write $\sigma(\mathsf{V})^{(\mathsf{min})} = \{\mathbf{X}\}$. For all $\hat{\mathsf{W}} \in \mathbf{X}$, run $\hat{\mathsf{Setup}}(1^\lambda, \hat{\mathsf{gpk}}, \hat{\mathsf{W}}) \rightarrow (\mathsf{pk}_{\hat{\mathsf{W}}}, \mathsf{sk}_{\hat{\mathsf{W}}})$. Let $\mathsf{pk}_{\mathsf{V}} = (\mathsf{pk}_{\hat{\mathsf{W}}})_{\hat{\mathsf{W}} \in \mathbf{X}}$ and $\mathsf{sk}_{\mathsf{V}} = (\mathsf{sk}_{\hat{\mathsf{W}}})_{\hat{\mathsf{W}} \in \mathbf{X}}$.

In the following, for each $\mathsf{V} \in \mathcal{V}_{\mathcal{G}}$ our construction will ensure that

$$\mathsf{sk}_{\mathsf{V}} = \left(\mathsf{sk}_{\hat{\mathsf{W}}} \right)_{\hat{\mathsf{W}} \in \hat{\mathbf{W}}} \text{ for some } \hat{\mathbf{W}} \in \sigma(\mathsf{V})^{(\mathsf{min})}. \tag{4.1}$$

We prove Eq.(4.1) by induction over the partial ordering $\preccurlyeq$. The base case, *i.e.,* at source node V, is trivial due to the output of Setup. Fixing V^* we assume that Eq.(4.1) is true for $\mathsf{V} \prec_{\mathsf{c}} \mathsf{V}^*$ and prove that it holds for V^*.

$\mathsf{Extract}(\mathsf{V}, \mathsf{V}^*, \mathsf{pk}(\mathsf{V}^*), \mathsf{sk}_{\mathsf{V}}) \rightarrow (\mathsf{sk}_{(\mathsf{V} \rightarrowtail \mathsf{V}^*)})$**:** First we denote some terms as follows.

- Parse $\mathsf{sk}_{\mathsf{V}} = (\mathsf{sk}_{\hat{\mathsf{W}}})_{\hat{\mathsf{W}} \in \hat{\mathbf{W}}}$.
- Parse $\{\mathbf{V} \in \Theta_{\mathsf{V}^*}^{(\mathsf{min})} \mid \mathsf{V} \in \mathbf{V}\}$ as $\{\mathbf{V}_1, \ldots, \mathbf{V}_m\}$.
- For $i = 1, \ldots, m$, parse $\mathbf{V}_i = \{\mathsf{V}, \mathsf{V}_{i,1}, \ldots, \mathsf{V}_{i,r_i}\}$.
- For $i = 1, \ldots, m$ and $j = 1, \ldots, r_i$, parse $\sigma(\mathsf{V}_{i,j})^{(\mathsf{min})} = \{\hat{\mathbf{W}}_{i,j,1}, \ldots, \hat{\mathbf{W}}_{i,j,s_{i,j}}\}$.

From Lemma 4.10, we have that for all $i = 1, \ldots, m$ and $\boldsymbol{d} = (d_1, \ldots, d_j) \in \mathcal{D}_i = \{1, \ldots, s_{i,1}\} \times \cdots \times \{1, \ldots, s_{i,r_i}\}$, there exists $\hat{\mathbf{Y}}_{i,\boldsymbol{d}} \in \sigma(\mathsf{V}^*)^{(\mathsf{min})}$ such that

$$\hat{\Sigma} \vdash [\hat{\mathbf{W}}] \wedge \bigwedge_{j=1}^{r_i} [\hat{\mathbf{W}}_{i,j,d_j}] \Rightarrow [\hat{\mathbf{Y}}_{i,\boldsymbol{d}}]. \tag{4.2}$$

Let $\hat{\mathbf{Z}}_{i,\boldsymbol{d}} = \hat{\mathbf{W}} \cup \bigcup_{j=1}^{r_i} \hat{\mathbf{W}}_{i,j,d_j}$. Also parse $\hat{\mathbf{Y}}_{i,\boldsymbol{d}} = \{\hat{\mathsf{Y}}_{i,\boldsymbol{d},1}, \ldots, \hat{\mathsf{Y}}_{i,\boldsymbol{d},t_{i,\boldsymbol{d}}}\}$. Then for all $\hat{\mathsf{W}} \in \hat{\mathbf{W}}$, $i = 1, \ldots, m$, $\boldsymbol{d} \in \mathcal{D}_i$, and $k = 1, \ldots, t_{i,\boldsymbol{d}}$, it runs

$$\hat{\mathsf{X}}\mathsf{Extract}\left(\hat{\mathbf{Z}}_{i,\boldsymbol{d}}, \hat{\mathsf{Y}}_{i,\boldsymbol{d},k}, (\hat{\mathsf{W}}, \mathsf{sk}_{\hat{\mathsf{W}}}) \right) \rightarrow \mathsf{tk}_{\hat{\mathbf{Z}}_{i,\boldsymbol{d}},(\hat{\mathsf{W}} \rightarrowtail \hat{\mathsf{Y}}_{i,\boldsymbol{d},k})}.$$

Note that such an algorithm is from Lemma 4.11 and can be run due to Eq.(4.2). Finally it outputs

$$\mathsf{sk}_{(\mathsf{V} \rightarrowtail \mathsf{V}^*)} = \left(\mathsf{tk}_{\hat{\mathbf{Z}}_{i,\boldsymbol{d}},(\hat{\mathsf{W}} \rightarrowtail \hat{\mathsf{Y}}_{i,\boldsymbol{d},k})} \right)_{\left[\begin{smallmatrix} \hat{\mathsf{W}} \in \hat{\mathbf{W}} \\ i=1,\ldots,m \\ \boldsymbol{d} \in \mathcal{D}_i \\ k=1,\ldots,t_{i,\boldsymbol{d}} \end{smallmatrix} \right]}.$$

$\mathsf{Combine}(\mathsf{V}^*, \mathsf{pk}(\mathsf{V}^*), S_{\mathsf{V}^*,\mathbf{V}'}) \to (\mathsf{sk}_{\mathsf{V}^*})$**:** Note that $\mathbf{V}' \in \Theta_{\mathsf{V}^*}$; WLOG we can assume that $\mathbf{V}' \in \Theta^{(\mathsf{min})}_{\mathsf{V}^*}$. Parse $\mathbf{V}' = \{\mathsf{V}'_1, \ldots, \mathsf{V}'_r\}$. Recall that $\mathsf{sk}_{\mathsf{V}'_j} = (\mathsf{sk}_{\hat{\mathsf{W}}})_{\hat{\mathsf{W}} \in \hat{\mathbf{W}}}$ for some $\hat{\mathbf{W}} \in \sigma(\mathsf{V}'_j)^{(\mathsf{min})}$. We denote such $\hat{\mathbf{W}}$ by $\hat{\mathbf{W}}_{\mathsf{V}'_j}$. From Lemma 4.10, we have that there exists $\hat{\mathbf{Y}}' = \{\hat{\mathsf{Y}}'_1, \ldots, \hat{\mathsf{Y}}'_t\} \in \sigma(\mathsf{V}^*)^{(\mathsf{min})}$ such that $\hat{\Sigma} \vdash \bigwedge_{j=1}^{r} [\hat{\mathbf{W}}_{\mathsf{V}'_j}] \Rightarrow [\hat{\mathbf{Y}}']$. Let $\hat{\mathbf{Z}}' = \bigcup_{j=1}^{r} \hat{\mathbf{W}}_{\mathsf{V}'_j}$. Parse $S_{\mathsf{V}^*,\mathbf{V}'} = (\mathsf{sk}_{(\mathsf{V}'_j \rightarrowtail \mathsf{V}^*)})_{j=1,\ldots,r}$. For $k = 1, \ldots, t$, do the following:

- From each $\mathsf{sk}_{(\mathsf{V}'_j \rightarrowtail \mathsf{V}^*)}$, we can extract the part $(\mathsf{tk}_{\hat{\mathbf{Z}}',(\hat{\mathsf{W}} \rightarrowtail \hat{\mathsf{Y}}_k)})_{\hat{\mathsf{W}} \in \hat{\mathbf{W}}_{\mathsf{V}'_j}}$. Collecting these parts for $j = 1, \ldots, r$, we obtain $(\mathsf{tk}_{\hat{\mathbf{Z}}',(\hat{\mathsf{W}} \rightarrowtail \hat{\mathsf{Y}}'_k)})_{\hat{\mathsf{W}} \in \hat{\mathbf{Z}}'}$.
- Run $\hat{\mathsf{X}}\mathsf{Combine}(\ \hat{\mathbf{Z}}', \hat{\mathsf{Y}}'_k, (\mathsf{tk}_{\hat{\mathbf{Z}}',(\hat{\mathsf{W}} \rightarrowtail \hat{\mathsf{Y}}'_k)})_{\hat{\mathsf{W}} \in \hat{\mathbf{Z}}'}\ ) \to \mathsf{sk}_{\hat{\mathsf{Y}}'_k}$.

Finally it outputs $\mathsf{sk}_{\mathsf{V}^*} = (\mathsf{sk}_{\hat{\mathsf{Y}}'_k})_{k=1,\ldots,t}$. This satisfies Eq.(4.1) at $\hat{\mathsf{V}}^*$ and thus completes the induction proof.

$\mathsf{Encrypt}(\mathsf{V}, \mathsf{pk}(\mathsf{V}), M) \to (C)$**:** Run the multiple encryption $\hat{\mathsf{M}}\mathsf{Encrypt}(\ \delta(\mathsf{V}), \mathsf{pk}(\delta(\mathsf{V})), M\ ) \to (C)$ and output the ciphertext C.

$\mathsf{Decrypt}(\mathsf{V}, \mathsf{pk}(\mathsf{V}), \mathsf{sk}_{\mathsf{V}}, C) \to (M)$**:** Parse $\mathsf{sk}_{\mathsf{V}} = (\mathsf{sk}_{\hat{\mathsf{W}}})_{\hat{\mathsf{W}} \in \hat{\mathbf{W}}}$, where $\hat{\mathbf{W}} \in \sigma(\mathsf{V})^{(\mathsf{min})}$. Due to property $\vdash [\![\sigma(\mathsf{V})]\!] \Rightarrow [\![\delta(\mathsf{V})]\!]$, we have that there exists $\hat{\mathbf{X}} \in \delta(\mathsf{V})^{(\mathsf{min})}$ such that $\hat{\mathbf{X}} \subseteq \hat{\mathbf{W}}$. For each $\hat{\mathsf{W}} \in \hat{\mathbf{X}}$, run

$$\hat{\mathsf{M}}\mathsf{Decrypt}(\ \delta(\mathsf{V}), \mathsf{pk}(\delta(\mathsf{V})), \hat{\mathsf{W}}, \mathsf{sk}_{\hat{\mathsf{W}}}, C\ ) \to (M_{\hat{\mathsf{W}}}).$$

Then it runs $\hat{\mathsf{G}}\mathsf{ather}(\ \delta(\mathsf{V}), \mathsf{pk}(\delta(\mathsf{V})), \hat{\mathbf{X}}, (M_{\hat{\mathsf{W}}})_{\hat{\mathsf{W}} \in \hat{\mathbf{X}}}\ ) \to (M)$ and outputs M.

Lemma 4.12. *The construction of $\mathcal{G}$-DAGE above is* IND-$(\mathsf{x}, \mathsf{n}|\ \mathcal{G}, \Delta)$-$\mathsf{Z}$*-secure, for* $\mathsf{x} \in \{\mathsf{a}, \mathsf{s}\}$, $\mathsf{Z} \in \{\mathsf{CPA}, \mathsf{CCA}\}$*, assuming that the underlying $\hat{\mathcal{G}}$-DAGE is* IND-$(\mathsf{x}, \mathsf{n}|\ \hat{\mathcal{G}}, \hat{\Delta})$-$\mathsf{Z}$*-secure.*

Proof (of Lemma 4.12). Suppose there exists an adversary, $\mathcal{A}$, that has advantage ϵ in attacking the $\mathcal{G}$-DAGE scheme. We build an algorithm $\mathcal{B}$ that successfully attack the $\hat{\mathcal{G}}$-DAGE scheme. Algorithm $\mathcal{B}$ proceeds by interacting with its challenger $\mathcal{C}$ and simulating the challenger for the view of $\mathcal{A}$ as follows.

Setup. Algorithm $\mathcal{B}$, upon input $\hat{\mathsf{gpk}}$ from $\mathcal{C}$, gives this $\hat{\mathsf{gpk}}(= \mathsf{gpk})$ to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ adaptively issues queries where each is one of the following. WLOG, the same query will be asked only once.

- Public key query (V) : '$\mathsf{pk}_{\mathsf{V}} =?$'. $\mathcal{B}$ responds by first querying to $\mathcal{C}$ public key queries $(\hat{\mathsf{W}})$: '$\mathsf{pk}_{\hat{\mathsf{W}}} =?$' for all $\hat{\mathsf{W}} \in \mathbf{X}$, where $\sigma(\mathsf{V})^{(\mathsf{min})} = \{\mathbf{X}\}$. $\mathcal{B}$ then gives $\mathsf{pk}_{\mathsf{V}} = (\mathsf{pk}_{\hat{\mathsf{W}}})_{\hat{\mathsf{W}} \in \mathbf{X}}$ to $\mathcal{A}$.

- Secret key query (V) : '$\mathsf{sk}_{\mathsf{V}} =?$'. In responding, $\mathcal{B}$ first randomly chooses $\hat{\mathbf{W}} \in \sigma(\mathsf{V})^{(\mathsf{min})}$ and then queries to $\mathcal{C}$ secret key queries $(\hat{\mathsf{W}})$: '$\mathsf{sk}_{\hat{\mathsf{W}}} =?$' for all $\hat{\mathsf{W}} \in \hat{\mathbf{W}}$. $\mathcal{B}$ finally returns $\mathsf{sk}_{\mathsf{V}} = (\mathsf{sk}_{\hat{\mathsf{W}}})_{\hat{\mathsf{W}} \in \hat{\mathbf{W}}}$ to $\mathcal{A}$.

- Decryption query (V, C) : '$\mathsf{Decrypt}(\mathsf{V}, \mathsf{pk}(\mathsf{V}), \mathsf{sk}_{\mathsf{V}}, C) =?$'. In responding, $\mathcal{B}$ first randomly chooses $\hat{\mathbf{X}} \in \delta(\mathsf{V})^{(\mathsf{min})}$ and queries to $\mathcal{C}$ a weak multiple decryption query $(\delta(\mathsf{V}), \hat{\mathbf{X}}, C)$:

$$\text{`}\hat{\mathsf{G}}\mathsf{ather}\Big(\delta(\mathsf{V}), \mathsf{pk}(\delta(\mathsf{V})), \hat{\mathbf{X}}, \big(\hat{\mathsf{M}}\mathsf{Decrypt}(\delta(\mathsf{V}), \mathsf{pk}(\delta(\mathsf{V})), \hat{\mathsf{W}}, \mathsf{sk}_{\hat{\mathsf{W}}}, C) \big)_{\hat{\mathsf{W}} \in \hat{\mathbf{X}}} \Big) =?\text{'},$$

  and returns the result to $\mathcal{A}$.

Recall that during this phase, $\mathcal{B}$ maintains the list of queries that have been asked by $\mathcal{A}$ for each oracle type: $\mathsf{L}_{\mathsf{pub}}, \mathsf{L}_{\mathsf{sec}}, \mathsf{L}_{\mathsf{dec}}$ respectively. On the other hand, $\hat{\mathsf{L}}_{\mathsf{pub}}, \hat{\mathsf{L}}_{\mathsf{sec}}, \hat{\mathsf{L}}_{\mathsf{dec}}$ are the lists of queries by $\mathcal{B}$ to its challenger $\mathcal{C}$. From the simulation above, it is direct that

$$\vdash [\hat{\mathsf{L}}_{\mathsf{sec}}] \Rightarrow [\mathsf{L}_{\mathsf{sec}}]|_{\sigma}, \tag{4.3}$$

where we recall the notation $[\mathsf{L}_{\mathsf{sec}}]|_{\sigma} = \bigwedge_{\mathsf{V} \in \mathsf{L}_{\mathsf{sec}}} [\![\sigma(\mathsf{V})]\!]$.

Challenge. Once $\mathcal{A}$ decides that Phase 1 is over, it outputs two messages $M_0, M_1 \in \mathcal{M}$ of equal length and a target node $\mathsf{V}^{\star}$ which it intends to attack, which satisfies the restriction:

$$\Delta \vdash \neg\big([\mathsf{L}_{\mathsf{sec}}] \Rightarrow \langle\!\langle \mathsf{V}^{\star} \rangle\!\rangle \big). \tag{4.4}$$

Recall that since we consider the Normal type of notion, we have Der becomes $\mathsf{L}_{\mathsf{sec}}$. In responding, $\mathcal{B}$ outputs to $\mathcal{C}$ the challenge $(M_0, M_1, \delta(\mathsf{V}^{\star}))$, obtains $C^{\star}$, and forward it to $\mathcal{A}$. We claim that this is a legitimate challenge, *i.e.,* it satisfies

$$\hat{\Delta} \vdash \neg\big([\hat{\mathsf{L}}_{\mathsf{sec}}] \Rightarrow [\![\delta(\mathsf{V}^{\star})]\!] \big). \tag{4.5}$$

We prove the claim as follows. The second property of graph syntactic consequence, $\hat{\Delta} \vdash \Delta|_{(\sigma,\delta)}$, and Eq.(4.4) together ensure that challenge, *i.e.,* it satisfies

$$\hat{\Delta} \vdash \neg\big([\mathsf{L}_{\mathsf{sec}}]|_{\sigma} \Rightarrow [\![\delta(\mathsf{V}^{\star})]\!] \big). \tag{4.6}$$

From Eq.(4.3), one can deduce by a simple propositional logic deduction that

$$\vdash \big([\hat{\mathsf{L}}_{\mathsf{sec}}] \Rightarrow [\![\delta(\mathsf{V}^{\star})]\!] \big) \Rightarrow \big([\mathsf{L}_{\mathsf{sec}}]|_{\sigma} \Rightarrow [\![\delta(\mathsf{V}^{\star})]\!] \big),$$

and hence

$$\vdash \neg\big([\mathsf{L}_{\mathsf{sec}}]|_{\sigma} \Rightarrow [\![\delta(\mathsf{V}^{\star})]\!] \big) \Rightarrow \neg\big([\hat{\mathsf{L}}_{\mathsf{sec}}] \Rightarrow [\![\delta(\mathsf{V}^{\star})]\!] \big).$$

Combining this with Eq.(4.6), we have that Eq.(4.5) holds, as claimed.

Phase 2. $\mathcal{A}$ issues additional queries where each is one of the following. $\mathcal{B}$ maintains the three lists as usual.

- Public key query (V) : '$\mathsf{pk}_{\mathsf{V}} =?$'. $\mathcal{B}$ responds as usual.
- Secret key query (V) : '$\mathsf{sk}_{\mathsf{V}} =?$'. Let $\mathsf{L}_{\mathsf{sec}}$ be the list before recording this query. Such a query satisfies the restriction:

$$\Delta \vdash \neg\big([\mathsf{L}_{\mathsf{sec}}] \wedge \langle\!\langle \mathsf{V} \rangle\!\rangle \Rightarrow \langle\!\langle \mathsf{V}^{\star} \rangle\!\rangle \big).$$

  As usual, $\mathcal{B}$ randomly chooses $\hat{\mathbf{W}} \in \sigma(\mathsf{V})^{(\mathsf{min})}$ and then queries to $\mathcal{C}$ secret key queries $(\hat{\mathsf{W}})$: '$\mathsf{sk}_{\hat{\mathsf{W}}} =?$' for all $\hat{\mathsf{W}} \in \hat{\mathbf{W}}$. At each query, $\hat{\mathsf{L}}_{\mathsf{sec}}$ is incrementally accumulated so

that the restriction for the last query in $\hat{\mathbf{W}}$ (and hence the overall restriction) will be

$$\hat{\Delta} \vdash \neg(\ [\hat{\mathsf{L}}_{\mathsf{sec}}] \wedge [\hat{\mathbf{W}}] \Rightarrow [\![\delta(\mathsf{V}^\star)]\!]\ ). \tag{4.7}$$

By using exactly the same approach as that of Eq.(4.5), we can prove that Eq.(4.7) holds.

- Decryption query (V, C) : '$\mathsf{Decrypt}(\mathsf{V}, \mathsf{pk}(\mathsf{V}), \mathsf{sk}_{\mathsf{V}}, C) =?$', with the property $(\mathsf{V}, C) \neq (\mathsf{V}^\star, C^\star)$. As in phase 1, $\mathcal{B}$ responds by querying to $\mathcal{C}$ a weak multiple decryption query $(\sigma(\mathsf{V}), \hat{\mathbf{X}}, C)$ for some $\hat{\mathbf{X}} \in \sigma(\mathsf{V})^{(\mathsf{min})}$. The restriction is that $(\sigma(\mathsf{V}), C) \neq (\sigma(\mathsf{V}^\star), C^\star)$ but this is direct consequence from the above property.

Guess Finally $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$. $\mathcal{B}$ just forwards b' to $\mathcal{C}$ as its guess.

It is not hard to see that the simulation of oracles for $\mathcal{A}$ is perfect. Hence, $\mathcal{B}$ has the same advantage as $\mathcal{A}$, which is ϵ, in attacking $\hat{\mathcal{G}}$-DAGE scheme. This completes the proof. □

4.4 Generic Constructions

4.4.1 Fully-secure Arbitrary Graph DAGE from HIBE

In this section, we give a fully-secure DAGE for any LAD from HIBE in a black-box manner. For any LAD, we deduce a forest graph (so that HIBE is sufficient to implement it) that graph syntactically implies it. The construction then follows from Construction 4-2 and the normal security level, *i.e.,* IND-$(\mathsf{x}, \mathsf{n}|\ \mathcal{G}, \Delta)$-$\mathsf{Z}$, follows from Lemma 4.12. The main point in this section is that when using such a forest graph, the deduced construction does not only achieve the normal security level, but also the full security level, *i.e.,* IND-$(\mathsf{x}, \mathsf{f}|\ \mathcal{G}, \Delta)$-$\mathsf{Z}$, where the partial key exposure is also allowed.

For generality, we will consider ID-based LADs, where any LADs can be casted by letting some ID variable fixed appropriately. We now recall the definition of ID-based LADs (cf. Definition 4.7). Let $G = (V, H)$ be a LAD. Its ID-based extension $\mathcal{G}_{\mathsf{Id}\mathsf{x}(G)} = (\mathcal{V}, \mathcal{H})$ defined from the base LAD G is given by

$$\mathcal{V} = \left\{ \left(v, (\mathsf{id}_w)_{w \preccurlyeq v}\right) \mid v \in V;\ \forall w \preccurlyeq v,\ \text{if } w \in V^0 \text{ then } \mathsf{id}_w = \varepsilon \text{ else } \mathsf{id}_w \in \mathcal{I} \right\}$$

$$\mathcal{H} = \left\{ \left\{ \left(v_1, (\mathsf{id}_w)_{w \preccurlyeq v_1}\right), \ldots, \left(v_j, (\mathsf{id}_w)_{w \preccurlyeq v_j}\right) \right\} \xrightarrow{\Theta \otimes_{\mathsf{Id}\mathsf{x}} (\mathsf{id}_w)_{w \preccurlyeq v^*}} \left(v^*, (\mathsf{id}_w)_{w \preccurlyeq v^*}\right) \right.$$
$$\left. \Big| \left(\{v_1, \ldots, v_j\} \xrightarrow{\Theta} v^*\right) \in H;\ \forall w \preccurlyeq v^*,\ \text{if } w \in V^0 \text{ then } \mathsf{id}_w = \varepsilon \text{ else } \mathsf{id}_w \in \mathcal{I} \right\}.$$

where $\Theta \otimes_{\mathsf{Id}\mathsf{x}} (\mathsf{id}_w)_{w \preccurlyeq v^*} = \left\{ \left\{ \left(v_{i_1}, (\mathsf{id}_w)_{w \preccurlyeq v_{i_1}}\right), \ldots, \left(v_{i_k}, (\mathsf{id}_w)_{w \preccurlyeq v_{i_k}}\right) \right\} \mid \{v_{i_1}, \ldots, v_{i_k}\} \in \Theta \right\}$.

We are interested in constructing DAGE for the LAD $\mathcal{G}_{\mathsf{Id}\mathsf{x}(G)}$ from HIBE, *i.e.,* DAGE of a tree graph. Fixing $\mathcal{G}_{\mathsf{Id}\mathsf{x}(G)}$, the description of such a forest (or tree) graph that yields

$\mathcal{G}_{\mathsf{Idx}(G)}$-DAGE is as follows. We denote it by $\mathcal{T}(\mathcal{G}_{\mathsf{Idx}(G)}) = (\mathcal{V}', \mathcal{H}')$.

$$\mathcal{V}' = \left\{ \left(v_0, (v_1, \mathsf{id}_{v_1}), \ldots, (v_t, \mathsf{id}_{v_t})\right) \mid v_0 \prec_{\mathsf{c}} v_1 \prec_{\mathsf{c}} \cdots \prec_{\mathsf{c}} v_t;\ v_0 \in V^0;\ \mathsf{id}_{v_1}, \ldots \mathsf{id}_{v_t} \in \mathcal{I} \right\}$$

$$\mathcal{H}' = \left\{ \left\{\left(v_0, (v_1, \mathsf{id}_{v_1}), \ldots, (v_t, \mathsf{id}_{v_t})\right)\right\} \to \left(v_0, (v_1, \mathsf{id}_{v_1}), \ldots, (v_{t+1}, \mathsf{id}_{v_{t+1}})\right) \right.$$
$$\left. \Big| \ v_0 \prec_{\mathsf{c}} v_1 \prec_{\mathsf{c}} \cdots \prec_{\mathsf{c}} v_{t+1};\ v_0 \in V^0;\ \mathsf{id}_{v_1}, \ldots \mathsf{id}_{v_{t+1}} \in \mathcal{I} \right\}.$$

Theorem 4.13. *$\mathcal{T}(\mathcal{G}_{\mathsf{Idx}(G)}) \Vdash_{\mathsf{syn}} \mathcal{G}_{\mathsf{Idx}(G)}$ via the maps $\sigma_{\mathsf{fr}} \equiv \delta$ which is defined recursively as follows.*

$$\begin{array}{rrcl}
\sigma_{\mathsf{fr}}: & \mathcal{V} & \to & \mathcal{AS}^{(\mathcal{V}')} \\
& (v) & \mapsto & \left\{\{v\}\right\} \qquad \text{for } v \in V^0, \\
& \left(v, (\mathsf{id}_w)_{w \preccurlyeq v}\right) & \mapsto & \left\{ \left\{ \sigma_{\mathsf{fr}}\left(\left(v_{i_1}, (\mathsf{id}_w)_{w \preccurlyeq v_{i_1}}\right)\right) \circ (v, \mathsf{id}_v), \ldots, \sigma_{\mathsf{fr}}\left(\left(v_{i_k}, (\mathsf{id}_w)_{w \preccurlyeq v_{i_k}}\right)\right) \circ (v, \mathsf{id}_v) \right\} \right. \\
& & & \left. \Big| \ \{v_{i_1}, \ldots, v_{i_k}\} \in \Theta_v \right\} \qquad \text{for } v \notin V^0.
\end{array}$$

Here, recall that $\circ$ denotes the concatenation of strings.

Proof. We prove that $\Sigma_{\mathcal{T}(\mathcal{G}_{\mathsf{Idx}(G)})} \vdash \Sigma_{\mathcal{G}_{\mathsf{Idx}(G)}}$ as follows. It is sufficient to prove that for all $v \in V$, for all $(\mathsf{id}_w)_{w \preccurlyeq v}$, for all $\{v_{i_1}, \ldots, v_{i_k}\} \in \Theta_v$, it must hold that

$$\Sigma_{\mathcal{T}(\mathcal{G}_{\mathsf{Idx}(G)})} \vdash \left[\left\{\left(v_{i_1}, (\mathsf{id}_w)_{w \preccurlyeq v_{i_1}}\right), \ldots, \left(v_{i_k}, (\mathsf{id}_w)_{w \preccurlyeq v_{i_k}}\right)\right\}\right]\Big|_{\sigma_{\mathsf{fr}}} \Rightarrow \sigma_{\mathsf{fr}}\left(\left(v, (\mathsf{id}_w)_{w \preccurlyeq v}\right)\right).$$

But this is straightforward from our construction of σ_{fr}. □

From this theorem and Theorem 4.9, we can conclude that if the underlying $\mathcal{T}(\mathcal{G}_{\mathsf{Idx}(G)})$-DAGE scheme $\mathcal{E}$ (which is an HIBE) is IND-$(\mathsf{x}, \mathsf{n}|\ \mathcal{G}, \Delta)$-$\mathsf{Z}$-secure, then the construction $\mathcal{G}_{\mathsf{Idx}(G)}$-$\mathsf{GenDAGE}(\mathcal{E}, \mathcal{T}(\mathcal{G}_{\mathsf{Idx}(G)}), \sigma_{\mathsf{fr}}, \sigma_{\mathsf{fr}})$ (cf. Construction 4-2) yields an IND-$(\mathsf{x}, \mathsf{n}|\ \mathcal{G}, \Delta)$-$\mathsf{Z}$-secure construction for $\mathcal{G}_{\mathsf{Idx}(G)}$-DAGE.

The main point of this section is that such a construction indeed provides full security, *i.e.,* we have the following theorem.

Theorem 4.14 (The Second Primitive Implication Theorem)**.** *Suppose that the underlying $\mathcal{T}(\mathcal{G}_{\mathsf{Idx}(G)})$-DAGE scheme $\mathcal{E}$ is IND-$(\mathsf{x}, \mathsf{f}|\ \mathcal{G}, \Delta)$-$\mathsf{Z}$-secure. Then the construction described above, or more formally $\mathcal{G}_{\mathsf{Idx}(G)}$-$\mathsf{GenDAGE}(\mathcal{E}, \mathcal{T}(\mathcal{G}_{\mathsf{Idx}(G)}), \sigma_{\mathsf{fr}}, \sigma_{\mathsf{fr}})$ (cf. Construction 4-2), yields an IND-$(\mathsf{x}, \mathsf{f}|\ \mathcal{G}, \Delta)$-$\mathsf{Z}$-secure construction for $\mathcal{G}_{\mathsf{Idx}(G)}$-DAGE.*

4.4.2 Weak DAGE from PKE and Cover-Admissable Families

In this section, we briefly describe how to construct DAGE from PKE. We give an observation that the existence of combinatorial structures which depends on the structure and the notions of the target DAGE is sufficient. We call such a structure cover-admissable family.

Definition 4.15 (Cover-Admissible Family)**.** For a LAD $\mathcal{G}$ and its derivability structure pair (Σ, Δ), we say that a set system $(N, \mathcal{S})$, where $\mathcal{S} = \{S_1, ..., S_\ell\} \in 2^N$, is a $(\mathcal{G}, \Sigma, \Delta)$-cover-admissible family there are maps

$$\tilde{\sigma} : \mathcal{V}_\mathcal{G} \to \{1, ..., \ell\}$$
$$\tilde{\delta} : \mathcal{V}_\mathcal{G} \to \{1, ..., \ell\}$$

such that

1. For all $\mathbf{V} \in 2^{\mathcal{V}_\mathcal{G}}, \mathsf{V}^* \in \mathcal{V}_\mathcal{G}$ such that $\Sigma \vdash [\mathbf{V}] \Rightarrow \langle\langle \mathsf{V}^* \rangle\rangle$ we have

$$S_{\tilde{\sigma}(\mathsf{V}^*)} \subseteq \bigcup_{\mathsf{V} \in \mathbf{V}} S_{\tilde{\sigma}(\mathsf{V})}.$$

2. For all $\mathbf{V} \in 2^{\mathcal{V}_\mathcal{G}}, \mathsf{V}^* \in \mathcal{V}_\mathcal{G}$ such that $\Delta \vdash \neg([\mathbf{V}] \Rightarrow \langle\langle \mathsf{V}^* \rangle\rangle)$ we have

$$S_{\tilde{\delta}(\mathsf{V}^*)} \not\subseteq \bigcup_{\mathsf{V} \in \mathbf{V}} S_{\tilde{\sigma}(\mathsf{V})}.$$

3. For all $\mathsf{V} \in \mathcal{V}_\mathcal{G}$, we have $S_{\tilde{\delta}(\mathsf{V})} \subseteq S_{\tilde{\gamma}(\mathsf{V})}$.

Theorem 4.16. *Suppose that $(N, \mathcal{S})$ is a $(\mathcal{G}, \Sigma, \Delta)$-cover-admissible family. Let $\hat{\mathcal{G}}_{\mathsf{PKE}} = (N, \emptyset)$ be a LAD. Then we have $(\hat{\mathcal{G}}_{\mathsf{PKE}}, \Sigma_{\hat{\mathcal{G}}_{\mathsf{PKE}}}, (\Sigma_{\hat{\mathcal{G}}_{\mathsf{PKE}}})^{\mathsf{c}}) \Vdash_{\mathsf{syn}} (\mathcal{G}, \Sigma, \Delta)$ via the maps*

$$\begin{array}{rccc} \sigma: & \mathcal{V}_\mathcal{G} & \to & \mathcal{AS}^N \\ & \mathsf{V} & \mapsto & \{S_{\tilde{\sigma}(\mathsf{V})}\} \\ \delta: & \mathcal{V}_\mathcal{G} & \to & \mathcal{AS}^N \\ & \mathsf{V} & \mapsto & \{S_{\tilde{\delta}(\mathsf{V})}\} \end{array}$$

The proof is straightforward. From this and Theorem 4.9, we can conclude that the construction of IND-$(\mathsf{a}, \mathsf{f}|\ \mathcal{G}, \Delta)$-Z-secure $\mathcal{G}$-DAGE is immediate from IND-CCA-secure PKE, if one can construct a (Σ, Δ)-cover-admissible family where $|S_1|, ..., |S_\ell|$ are $O(\mathsf{poly}(\lambda))$. As a special case, the generic construction of threshold collusion-resistant Key-insulated PKE of [DK02] (which is further generalized to 2-level HIBE with threshold collusion-resistant in [HHSI05]) is the one where the underlying cover-admissible family is rendered to a standard cover-free family.

4.5 Efficient OR Graph DAGE Construction

In this section, we construct an efficient DAGE construction for monotone OR-graph from bilinear pairing. Although one can construct it from HIBE via the generic conversion, the resulting scheme does not enjoy practical efficiency. In particular, since the generic conversion utilizes the generic multiple encryption, the expansion of ciphertext is linear to the size of support of access structure for target recipients. In this section, we give a construction with constant-size ciphertext based on bilinear pairing. The security proof is done in the standard model. For generality, as usual we consider ID-based graphs.

Let $G = (V, H)$ be a LAD that all access structures at hyper-edges are of OR-type, *i.e.,* a monotone OR graph. (Note that thus one can also view G as a DAG). For simplicity, we

consider the case that there is one source node. Its ID-based extension $\mathcal{G}_{\mathsf{Idx}(G)} = (\mathcal{V}, \mathcal{H})$ defined from the base DAG G is given by

$$\mathcal{V} = \left\{ \left(v, (\mathsf{id}_w)_{w \preccurlyeq v}\right) \mid v \in V;\ \forall w \preccurlyeq v,\ \text{if } w \in V^0 \text{ then } \mathsf{id}_w = \varepsilon \text{ else } \mathsf{id}_w \in \mathcal{I} \right\}$$

$$\mathcal{H} = \left\{ \left\{ \left(v_1, (\mathsf{id}_w)_{w \preccurlyeq v_1}\right), \ldots, \left(v_j, (\mathsf{id}_w)_{w \preccurlyeq v_j}\right) \right\} \xrightarrow{\vee} \left(v^*, (\mathsf{id}_w)_{w \preccurlyeq v^*}\right) \right.$$
$$\left. \Big| \left(\{v_1, \ldots, v_j\} \xrightarrow{\vee} v^*\right) \in H;\ \forall w \preccurlyeq v^*,\ \text{if } w \in V^0 \text{ then } \mathsf{id}_w = \varepsilon \text{ else } \mathsf{id}_w \in \mathcal{I} \right\}.$$

Construction 4-3. OR Graph DAGE with Constant-Size Ciphertext

Setup(): Let $\mathbb{G}$ be a bilinear group of prime order p. The algorithm first picks a random generator $g \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p$. Let $g_1 = g^\alpha$. Next, pick randomly $g_2, y \in \mathbb{G}$ and for each $v \in V$ pick randomly $h_v \in \mathbb{G}$. Let $J : V \times \mathcal{I} \to \mathbb{Z}_p$ be a collision-resistant hash function. The public key and the master key are given by

$$\mathsf{pk} = \left(g, g_1, g_2, y, \left(h_v\right)_{v \in V_G}, J \right), \quad \mathsf{msk} = g_2^\alpha.$$

Extract(): A private key of vertex $\mathsf{V} = \left(v, \{(w, \mathsf{id}_w)\}_{w \preccurlyeq v}\right)$ will be of the following form $\mathsf{sk}_\mathsf{V} = \left(d, z, \left\{(w, d_w)\right\}_{\left[\substack{w \not\preccurlyeq v \\ w \in V_G}\right]}\right)$, where

$$d = g_2^\alpha \cdot \left(y \prod_{w \preccurlyeq v} h_w^{J(w, \mathsf{id}_w)} \right)^s, \quad z = g^s, \quad d_w = h_w^s.$$

To generate $\mathsf{sk}_{\mathsf{V}^*}$ for $\mathsf{V}^* = \left(v^*, \{(w, \mathsf{id}_w)\}_{w \preccurlyeq v^*}\right)$ from sk_V, first pick a random elements $\delta \in \mathbb{Z}_p$. It then lets

$$d^* = d \cdot \left(\prod_{\substack{w \preccurlyeq v^* \\ w \not\preccurlyeq v}} d_w^{J(w, \mathsf{id}_w)} \right) \cdot \left(y \prod_{w \preccurlyeq v^*} h_w^{J(w, \mathsf{id}_w)} \right)^\delta, \quad z^* = z \cdot g^\delta, \quad d_w^* = d_w \cdot h_w^\delta.$$

It outputs $\mathsf{sk}_{\mathsf{V}^*} = \left(d^*, z^*, \left\{(w, d_w)\right\}_{\left[\substack{w \not\preccurlyeq v^* \\ w \in V_G}\right]}\right)$.

Encrypt(): To encrypt a message $M \in \mathbb{G}_T$, pick a random $r \in \mathbb{Z}_p$ and output

$$C = \left(e(g_1, g_2)^r \cdot M,\ g^r,\ \left(y \prod_{w \preccurlyeq v^*} h_w^{J(w, \mathsf{id}_w)} \right)^r \right).$$

Decrypt(): Consider a vertex V. To decrypt a given ciphertext $C = (C_1, C_2, C_3)$ using the

$$\text{private key } \mathsf{sk}_{\mathsf{V}} = \left(d, z, \left\{(w, d_w)\right\}_{\substack{w \not\preccurlyeq v \\ w \in V_G}}\right), \text{ output}$$

$$C_1 \cdot \frac{e(C_3, z)}{e(C_2, d)} = M$$

Theorem 4.17. *Suppose that the Decision $|V_G|$-BDHE assumption holds in $\mathbb{G}$. Then the above scheme is IND-$(\mathsf{s}, \mathsf{f}|\ \mathcal{G}_{\mathsf{Idx}(G)}, (\Sigma_{\mathcal{G}_{\mathsf{Idx}(G)}})^{\mathsf{c}})$-CPA-secure for any OR-graph G.*

Proof. Suppose there exists an adversary, $\mathcal{A}$, that has advantage ϵ in attacking the above scheme. We build an algorithm $\mathcal{B}$ that solves the Decision n-BDHE problem in $\mathbb{G}$, where we denote $n = |V_G|$. $\mathcal{B}$ is given as input a random n-BDHE challenge $(g, h, \vec{y}_{g,\alpha,n}, Z)$, where $\vec{y}_{g,\alpha,n} = (y_1, \ldots, y_n, y_{n+2}, \ldots, y_{2n})$ and Z is either $e(y_{n+1}, h)$ or a random element in $\mathbb{G}_1$, where we denote $y_j = g^{(\alpha^j)}$. Algorithm $\mathcal{B}$ proceeds as follows.

Initialization. The selective node game begins with $\mathcal{A}$ first outputting a target node $\mathsf{V}^* = \left(v^*, \{(w, \mathsf{id}^\star_w)\}_{w \preccurlyeq v^*}\right)$ that it intends to attack.

Setup. Algorithm $\mathcal{B}$ first defines a random bijective map $f : V_G \to \{1, \ldots, n\}$. To generate pk, algorithm $\mathcal{B}$ picks a random $\gamma \in \mathbb{Z}_p$ and sets $g_1 = y_1 = g^\alpha$ and $g_2 = y_n^{-1} \cdot g^\gamma = g^{\gamma - (\alpha^n)}$. Next, $\mathcal{B}$ randomly chooses $\beta_1, \ldots, \beta_n, \sigma \in \mathbb{Z}_p$ then defines

$$h_{f(v)} = y_{n+1-f(v)} \cdot g^{\beta_{f(v)}} \in \mathbb{G}$$

for all $v \in V_G$ and sets

$$y = g^\sigma \cdot \prod_{w \preccurlyeq v^*} y_{n+1-f(w)}^{-J(w, \mathsf{id}^\star_w)}.$$

It gives $\mathcal{A}$ the $\mathsf{pk} = \left(g, g_1, g_2, y, \left(h_v\right)_{v \in \mathcal{V}}, J\right)$. Since $g, \alpha, \gamma, \sigma$ and β_j's are chosen randomly and independently, pk has an identical distribution to that in the actual construction.

Phase 1. $\mathcal{A}$ issues up to q_{sec} private key queries. Consider a query for the private key corresponding to node $\hat{\mathsf{V}} = \left(\hat{v}, \{(w, \hat{\mathsf{id}}_w)\}_{w \preccurlyeq \hat{v}}\right)$, of which we can categorize to two cases.

1. There exists $v \preccurlyeq \hat{v}$ such that $v \preccurlyeq v^\star$ and $\hat{\mathsf{id}}_v \neq \mathsf{id}^\star_v$.
2. $v^\star \prec \hat{v}$ and for all $w \preccurlyeq v^\star$, $\hat{\mathsf{id}}_w = \mathsf{id}^\star_w$.

Case 1. For the first case, we assume WLOG that v is the smallest one, *i.e.,* there exists no $v' \prec v$ with the same property. Note that such smallest v is not necessarily unique; we just pick one. Therefore, $\hat{\mathsf{id}}_w = \mathsf{id}^\star_w$ for all $w \prec v$. $\mathcal{B}$ computes a private key for $\mathsf{V} = \left(v, \{(w, \hat{\mathsf{id}}_w)\}_{w \preccurlyeq v}\right)$ from which it then constructs a private key for the requested node $\hat{\mathsf{V}} = \left(\hat{v}, \{(w, \hat{\mathsf{id}}_w)\}_{w \preccurlyeq \hat{v}}\right)$. $\mathcal{B}$ picks random elements $s \in \mathbb{Z}_p$. We pose $\tilde{s} = s + \alpha^{f(v)}/(J(v, \hat{\mathsf{id}}_v) - J(v, \mathsf{id}^\star_v))$. Note that $\tilde{s}$ is unknown to $\mathcal{B}$. Next, $\mathcal{B}$ generates the private key $\mathsf{sk}_{\mathsf{V}} = \left(d, z, \left\{(w, d_w)\right\}_{\substack{w \not\preccurlyeq v \\ w \in V_G}}\right)$ where

$$d = g_2^\alpha \cdot \left(y \prod_{w \preccurlyeq v} h_w^{J(w, \hat{\mathsf{id}}_w)}\right)^{\tilde{s}}, \quad z = g^{\tilde{s}}, \quad d_w = h_w^{\tilde{s}}. \tag{4.8}$$

which is a valid random private key for node V by definition. We show that $\mathcal{B}$ can compute all elements of this private key given the values that it knows. First assume that $v \prec v^\star$. To generate d we observe that

$$d = y_1^{\gamma} \cdot y_{n+1}^{-1} \cdot \Bigg(g^{\sigma+\sum_{w \preccurlyeq v} \beta_{f(w)}} \cdot \underbrace{\prod_{w \prec v} y_{n+1-f(w)}^{J(w,\hat{\mathsf{id}}_w)-J(w,\mathsf{id}_w^\star)}}_{=1} \cdot y_{n+1-f(v)}^{J(v,\hat{\mathsf{id}}_v)-J(v,\mathsf{id}_v^\star)} \cdot \prod_{\substack{w \preccurlyeq v^\star \\ w \not\preccurlyeq v}} y_{n+1-f(w)}^{-J(w,\mathsf{id}_w^\star)} \Bigg)^{\tilde{s}}$$

$$= \underbrace{y_{n+1}^{-1} \cdot y_{n+1-f(v)}^{(J(v,\hat{\mathsf{id}}_v)-J(v,\mathsf{id}_v^\star))\tilde{s}}}_{T_1} \cdot \underbrace{g^{(\sigma+\sum_{w \preccurlyeq v} \beta_{f(w)})\tilde{s}}}_{T_2} \cdot \underbrace{\prod_{\substack{w \preccurlyeq v^\star \\ w \not\preccurlyeq v}} y_{n+1-f(w)}^{-J(w,\mathsf{id}_w^\star)\tilde{s}}}_{T_3}.$$

The term T_1 can be computed by $\mathcal{B}$ since

$$T_1 = y_{n+1}^{-1} \cdot y_{n+1-f(v)}^{(J(v,\hat{\mathsf{id}}_v)-J(v,\mathsf{id}_v^\star))(s+\frac{\alpha^{f(v)}}{(J(v,\hat{\mathsf{id}}_v)-J(v,\mathsf{id}_v^\star))})}$$
$$= y_{n+1}^{-1} \cdot y_{n+1-f(v)}^{(J(v,\hat{\mathsf{id}}_v)-J(v,\mathsf{id}_v^\star))s} \cdot y_{n+1-f(v)}^{\alpha^{f(v)}} = y_{n+1-f(v)}^{(J(v,\hat{\mathsf{id}}_v)-J(v,\mathsf{id}_v^\star))s}.$$

where the unknown term y_{n+1} is canceled out. The term T_2 can be computed by using $y_{f(v)}$, which is not y_{n+1} by definition of $f(\cdot)$. To see that each term in the product T_3 is computable, we observe that

$$y_{n+1-f(w)}^{\tilde{s}} = y_{n+1-f(w)}^{s} \cdot y_{n+1-f(w)+f(v)}^{1/(J(v,\hat{\mathsf{id}}_v)-J(v,\mathsf{id}_v^\star))}.$$

It is clear that $y_{n+1-f(w)}$ can be computed (since $f(w) \neq 0$). For $w \not\preccurlyeq v$, due to the bijection of f we have that $n+1-f(w)+f(v) \neq n+1$ and that $1 \leq n+1-f(w)+f(v) \leq n$ or $n+2 \leq n+1-f(w)+f(v) \leq 2n$, hence $y_{n+1-f(w)+f(v)}$ can be computed.

It is left to consider the case $v = v^\star$. In this case, d is exactly the same as above except that the last product term, T_3, does not appear. The analysis of computability by $\mathcal{B}$ thus follows from the same argument.

The component a_1 can be generated since $a_1 = g^{\tilde{s}} = g^s \cdot y_k^{1/(\mathrm{I}_k - \mathrm{I}_k^\star)}$. For $j = k+1, \ldots, z$, b_j can be computed as $b_j = h_j^{\tilde{s}} = h_j^s (y_{n+1-j+k} y_k)^{1/(\mathrm{I}_k - \mathrm{I}_k^\star)}$.

Case 2. In this case, let v' be the node such that $v^\star \prec_{\mathsf{c}} v'$ and $v' \preccurlyeq \hat{v}$. $\mathcal{B}$ responds to the query by first computing a private key for node $\mathsf{V}' = \big(v', \{(w, \hat{\mathsf{id}}_w)\}_{w \preccurlyeq v'}\big)$ from which it then construct a private key for the request node $\hat{\mathsf{V}} = \big(\hat{v}, \{(w, \hat{\mathsf{id}}_w)\}_{w \preccurlyeq \hat{v}}\big)$. $\mathcal{B}$ picks random elements $s \in \mathbb{Z}_p$. We pose $\tilde{s} = s + \alpha^{f(v')}/(J(v', \mathsf{id}_{v'}))$. Note that $\tilde{s}$ is unknown to $\mathcal{B}$. Next, $\mathcal{B}$ generates the private key in exactly the same form as (4.8). From a similar observation as above, one can show that $\mathcal{B}$ is able to compute this key.

Challenge. When $\mathcal{A}$ decides that Phase 1 is over, it outputs two messages $M_0, M_1 \in \mathbb{G}_1$ on which it wishes to be challenged. The algorithm $\mathcal{B}$ generates the challenge by first randomly chooses a bit $b \in \{0,1\}$ and then computing

$$C^\star = \Big(Z^{-1} \cdot e(y_1, h)^{\gamma} \cdot M_b,\ h,\ h^{\sigma+\sum_{w \preccurlyeq v^\star} \beta_{f(w)} J(w,\mathsf{id}_w)} \Big).$$

We claim that when $Z = e(y_{n+1}, h)$ (that is, the input to $\mathcal{B}$ is a n-BDHE tuple) then $C^\star$ is a valid challenge to $\mathcal{A}$ as in a real attack game. To see this, write $h = g^t$ for some

(unknown) $t \in \mathbb{Z}_p$. Then, we have that

$$Z^{-1} \cdot e(y_1, h)^{\gamma} = \left(e(y_{n+1}^{-1}, g) \cdot e(y_1^{\gamma}, g)\right)^t = e(y_1, y_n^{-1} g^{\gamma})^t = e(g_1, g_2)^t$$

$$h^{\sigma + \sum_{w \preccurlyeq v^\star} \beta_{f(w)} J(w, \mathsf{id}_w)} = \left(\left(g^{\sigma} \cdot \prod_{w \preccurlyeq v^*} y_{n+1-f(w)}^{-J(w, \mathsf{id}_w^\star)} \right) \cdot \prod_{w \preccurlyeq v^\star} \left(y_{n+1-f(w)} \cdot g^{\beta_{f(w)}} \right)^{J(w, \mathsf{id}_w^\star)} \right)^t$$

$$= \left(y \cdot \prod_{w \preccurlyeq v^\star} h_w^{J(w, \mathsf{id}_w^\star)} \right)^t .$$

Thus, by definition, $C^\star$ is a valid encryption of the key M_b.

On the other hand, when Z is random in $\mathbb{G}_1$ (that is, the input to $\mathcal{B}$ is a random tuple) then $C^\star$ is independent of b in the adversary's view.

Phase 2. $\mathcal{A}$ continues to ask queries not issued in Phase 1. $\mathcal{B}$ responds as before.

Guess. Finally, $\mathcal{A}$ outputs $b' \in \{0, 1\}$ for guessing b. If $b = b'$ then $\mathcal{B}$ outputs 1 (meaning $Z = e(y_{n+1}, h)$). Otherwise, it outputs 0 (meaning Z is random in $\mathbb{G}_1$).

We see that if $(g, h, \vec{y}_{g,\alpha,n}, Z)$ is sampled from $\mathcal{R}_{BDHE}$ then $\Pr[\mathcal{B}(g, h, \vec{y}_{g,\alpha,n}, Z) = 0] = \frac{1}{2}$. On the other hand, if $(g, h, \vec{y}_{g,\alpha,n}, Z)$ is sampled from $\mathcal{P}_{BDHE}$ then $|\Pr[\mathcal{B}(g, h, \vec{y}_{g,\alpha,n}, Z) = 0] - \frac{1}{2}| \geq \epsilon$. It follows that $\mathcal{B}$ has advantage at least ϵ in solving n-BDHE problem in $\mathbb{G}$. This concludes the proof of Theorem 4.17. □

4.6 Efficient OR Bounded-Complete Graph DAGE Construction

In this section, we consider what we call a complete DAG. It is an analog of complete graph for directed graph. The important of this kind of DAG is that it enables flexibility of hierarchy since complete graph connects all nodes. When coming to the case of DAG, which cyclic paths are not allowed, we have to refine what we mean by complete DAG. Intuitively, we assign each node with rank, which starts at 0 for the source nodes. Only nodes of lower ranks can be edged directed into those of upper ranks. Each node will be specified completely by the structure of all lower-rank nodes edged into it with the ID for each node. More formally we define the complete DAG as follows. First define recursively

$$\mathcal{V}_0 = \{\text{'R'}\},$$
$$\mathcal{V}_i = \{(\mathsf{V}_1, \ldots, \mathsf{V}_j, \mathsf{id}) \mid \mathsf{V}_1, \ldots, \mathsf{V}_j \in \mathcal{V}_0 \cup \cdots \cup \mathcal{V}_{i-1};\ j \geq 1;\ \ \mathsf{id} \in \mathcal{I}\}.$$

The graph $\mathcal{G}_{\mathsf{Comp}} = (\mathcal{V}, \mathcal{H})$ is defined by

$$\mathcal{V} = \mathcal{V}_0 \ \cup\ \mathcal{V}_1 \ \cup\ \mathcal{V}_2 \ \cup\ \cdots$$
$$\mathcal{H} = \{\ \{\mathsf{V}_1, \ldots, \mathsf{V}_j\} \rightarrow (\mathsf{V}_1, \ldots, \mathsf{V}_j, \mathsf{id}) \mid (\mathsf{V}_1, \ldots, \mathsf{V}_j, \mathsf{id}) \in \mathcal{V}\}.$$

If there is a directed path from W to V, we say that W is a predecessor of V, denoted $\mathsf{W} \preccurlyeq \mathsf{V}$.

Due to exponential number of nodes in each rank of this DAG, it seems that to construct a DAGE scheme for this DAG with monotone structure such as OR-graph is impossible at first glance. Construction 4-3 cannot implement this since the public key size will end up with exponential number. In this section, we present a OR-graph construction for a

reasonable relaxed version of the complete DAG where we have a-priori bounded number of predecessors of each node. We call such a graph the bounded-complete DAG. Let t be such a bound. The graph $\mathcal{G}_{\mathsf{Comp}(t)} = (\mathcal{V}', \mathcal{H}')$ is defined by

$$\mathcal{V}' = \left\{ \mathsf{V} \in \mathcal{V} \;\middle|\; |\{\mathsf{W} \in \mathcal{V} \mid \mathsf{W} \preccurlyeq \mathsf{V}\}| \leq t \right\}$$
$$\mathcal{H}' = \left\{ \{\mathsf{V}_1, \ldots, \mathsf{V}_j\} \to (\mathsf{V}_1, \ldots, \mathsf{V}_j, \mathsf{id}) \;\middle|\; (\mathsf{V}_1, \ldots, \mathsf{V}_j, \mathsf{id}) \in \mathcal{V}' \right\}.$$

In what follows, we let $\|\mathsf{V}\| = |\{\mathsf{W} \in \mathcal{V} \mid \mathsf{W} \preccurlyeq \mathsf{V}\}|$.

Let the OR-graph LAD for the DAG $\mathcal{G}_{\mathsf{Comp}(t)}$ be denoted by $\mathcal{G}_{\mathsf{Comp\text{-}OR}(t)}$. We construct a $\mathcal{G}_{\mathsf{Comp\text{-}OR}(t)}$-DAGE as follows.

Construction 4-4. OR Bounded-Complete Graph DAGE

Setup($\mathcal{G}$): Let $\mathbb{G}$ be a bilinear group of prime order p. The algorithm first picks a random generator $g \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p$. Let $g_1 = g^\alpha$. Next, pick random elements $g_2, h_0, \ldots, h_t \in \mathbb{G}$. Let $H : \mathcal{V} \to \mathbb{Z}_p^*$ be a collision-resistant hash function. The public key and the master key are given by

$$\mathsf{pk} = (g, g_1, g_2, h_0, \ldots, h_t, H), \quad \mathsf{msk} = g_2^\alpha.$$

Define $F : \mathbb{Z}_p \to \mathbb{G}$ to be the function $F(x) = \prod_{j=0}^{t} h_j^{(x^j)}$. (Recall that t is the maximum number of predecessors allowed for one node).

Extract(): A private key of vertex V will be of the form $\mathsf{sk}_\mathsf{V} = (d, (\mathsf{W}, d_\mathsf{W})_{\mathsf{W} \preccurlyeq \mathsf{V}})$, where

$$d = g_2^\alpha \cdot \prod_{\mathsf{W} \preccurlyeq \mathsf{V}} F(H(\mathsf{W}))^{s_\mathsf{W}} \quad \text{and} \quad d_\mathsf{W} = g^{s_\mathsf{W}}.$$

To generate $\mathsf{sk}_{\mathsf{V}^*}$ from $\mathsf{sk}_\mathsf{V} = (d, (\mathsf{W}, d_\mathsf{W})_{\mathsf{W} \preccurlyeq \mathsf{V}})$, first pick random elements $s_\mathsf{W} \in \mathbb{Z}_p$ for W such that $\mathsf{W} \preccurlyeq \mathsf{V}^*, \mathsf{W} \not\preccurlyeq \mathsf{V}$. It then lets

$$d^* = d \cdot \prod_{\substack{\mathsf{W} \preccurlyeq \mathsf{V}^* \\ \mathsf{W} \not\preccurlyeq \mathsf{V}}} F(H(\mathsf{W}))^{s_\mathsf{W}} \quad \text{and} \quad d^*_\mathsf{W} = \begin{cases} d_\mathsf{W} & \text{if } \mathsf{W} \preccurlyeq \mathsf{V}, \\ g^{s_\mathsf{W}} & \text{if } \mathsf{W} \preccurlyeq \mathsf{V}^*, \mathsf{W} \not\preccurlyeq \mathsf{V}. \end{cases}$$

It outputs $\mathsf{sk}_{\mathsf{V}^*} = (d^*, (\mathsf{W}, d^*_\mathsf{W})_{\mathsf{W} \preccurlyeq \mathsf{V}^*})$.

Encrypt(): To encrypt a message $M \in \mathbb{G}_T$, pick a random $r \in \mathbb{Z}_p$ and output

$$C = \left(e(g_1, g_2)^r \cdot M,\ g^r,\ \left(F(H(\mathsf{W}))^r\right)_{\mathsf{W} \preccurlyeq \mathsf{V}} \right).$$

Decrypt(): Consider a vertex V. To decrypt a given ciphertext $C = (A, B, (C_\mathsf{W})_{\mathsf{W} \preccurlyeq \mathsf{V}})$ using the private key $\mathsf{sk}_\mathsf{V} = (d, (\mathsf{W}, d_\mathsf{W})_{\mathsf{W} \preccurlyeq \mathsf{V}})$, output

$$A \cdot \frac{\prod_{\mathsf{W} \preccurlyeq \mathsf{V}} e(C_\mathsf{W}, d_\mathsf{W})}{e(B, d)} = M$$

Theorem 4.18. *Suppose that the Decision BDH assumption holds in $\mathbb{G}$. Then the above scheme is IND-$(\mathsf{s},\mathsf{f}|\ \mathcal{G}_{\mathsf{Comp\text{-}OR}(t)},(\Sigma_{\mathcal{G}_{\mathsf{Comp\text{-}OR}(t)}})^{\mathsf{c}})$-CPA-secure.*

Proof. Suppose there exists an adversary, $\mathcal{A}$, that has advantage ϵ in attacking the DAGE scheme. We build an algorithm $\mathcal{B}$ that solves the Decision BDH problem in $\mathbb{G}$. Algorithm $\mathcal{B}$ is given as input a random BDH challenge (g, g^a, g^b, g^c, Z), where Z is either $e(g,g)^{abc}$ or a random element in $\mathbb{G}_T$. Set $g_1 = g^a, g_2 = g^b, g_3 = g^c$. Algorithm $\mathcal{B}$ proceeds as follows.

Initialization. The game begins with $\mathcal{A}$ first outputting a node $\mathsf{V}^\star \in \mathcal{V}'$.

Setup. To generate the public key, algorithm $\mathcal{B}$ first defines a (t)-degree polynomial in $\mathbb{Z}_p[x]$ as $f(x) = a_t x^t + a_{t-1}x^{t-1} + \cdots + a_1 x + a_0$ for random elements $a_i \in \mathbb{Z}_p$. Next it defines

$$q(x) = \prod_{\mathsf{W} \preccurlyeq \mathsf{V}^\star} \big(x - H(\mathsf{W})\big) = b_t x^t + b_{t-1}x^{t-1} + \cdots + b_1 x + b_0,$$

where we note that b_i's term can be computed completely from $\mathsf{V}^\star$ and note that for $i \geq \|\mathsf{V}^\star\|$, $b_i = 0$. In this way and due to the collision-resistance of H, we can ensure that $q(x) = 0$ if and only if $x = H(W)$ for some $\mathsf{W} \preccurlyeq \mathsf{V}^\star$. It then lets $h_j = g_2^{b_j} \cdot g^{a_j}$ for $j = 0, \ldots, t$. We thus have

$$F(x) = \prod_{j=0}^{t} h_j^{(x^j)} = g_2^{q(x)} \cdot g^{f(x)}.$$

It then outputs $\mathsf{pk} = (g, g_1, g_2, h_0, \ldots, h_t, H)$. Since $g, a, b, a_0, \ldots, a_t$ are chosen randomly and independently, we have that pk has an identical distribution to that that in the actual construction.

Phase 1. $\mathcal{A}$ issues up to q_{P} private key queries. Consider a query for the private key corresponding to node V. From the restriction, it must be that $\mathsf{V} \not\preccurlyeq \mathsf{V}^\star$. Hence there must be $\mathsf{V}', \mathsf{V}''$ such that $\mathsf{V}' \prec_{\mathsf{c}} \mathsf{V}'' \preccurlyeq \mathsf{V}$ and $\mathsf{V}' \preccurlyeq \mathsf{V}^\star$ but $\mathsf{V}'' \not\preccurlyeq \mathsf{V}^\star$. Note that this is not necessarily unique. $\mathcal{B}$ responds to the query by first computing a private key for node V'' from which it then constructs a private key for request node V. Fix $\mathsf{Y} = H(\mathsf{V}'')$. Algorithm $\mathcal{B}$ picks random element $s_{\mathsf{w}} \in \mathbb{Z}_p$ for each $\mathsf{w} \preccurlyeq \mathsf{V}''$ and sets

$$d = g_1^{-f(\mathsf{Y})/q(\mathsf{Y})} \cdot \prod_{\mathsf{W} \preccurlyeq \mathsf{V}''} F(H(\mathsf{W}))^{s_{\mathsf{W}}}, \quad \big(\ d_{\mathsf{W}}\ \big)_{\mathsf{W} \prec \mathsf{V}''} = \big(\ g^{s_{\mathsf{W}}}\ \big)_{\mathsf{W} \prec \mathsf{V}''}, \quad d_{\mathsf{V}''} = g^{s_{\mathsf{w}}} \cdot g_1^{-1/q(\mathsf{Y})}.$$

Since $\mathsf{V}'' \not\preccurlyeq \mathsf{V}^\star$ thus $q(\mathsf{V}'') \neq 0$, hence the above terms can be computed. We claim that this is a valid random private key for V''. We pose $\tilde{s}_{\mathsf{V}''} = s_{\mathsf{V}''} - a/q(\mathsf{Y})$. Then observe that

$$g_1^{-f(\mathsf{Y})/q(\mathsf{Y})} \cdot F(\mathsf{Y})^{s_{\mathsf{V}''}} = g_1^{-f(\mathsf{Y})/q(\mathsf{Y})} \cdot (g_2^{q(\mathsf{Y})} g^{f(\mathsf{Y})})^{s_{\mathsf{V}''}} = g_2^a \cdot (g_2^{q(\mathsf{Y})} g^{f(\mathsf{Y})})^{s_{\mathsf{V}''} - a/q(\mathsf{Y})} = g_2^a \cdot F(\mathsf{Y})^{\tilde{s}_{\mathsf{V}''}}.$$

It follows that the private key $\mathsf{sk}_{\mathsf{V}''} = (d, (\mathsf{W}, d_{\mathsf{W}})_{\mathsf{W} \preccurlyeq \mathsf{V}''})$ defined above satisfies

$$d = g_2^a \cdot \Big(\prod_{\mathsf{W} \prec \mathsf{V}''} F(H(\mathsf{W}))^{s_{\mathsf{W}}} \Big) \cdot F(H(\mathsf{V}''))^{\tilde{s}_{\mathsf{V}''}}, \quad \big(\ d_{\mathsf{W}}\ \big)_{\mathsf{W} \prec \mathsf{V}''} = \big(\ g^{s_{\mathsf{W}}}\ \big)_{\mathsf{W} \prec \mathsf{V}''}, \quad d_{\mathsf{V}''} = g^{\tilde{s}_{\mathsf{w}}},$$

where $(s_{\mathsf{W}})_{\mathsf{W} \prec \mathsf{V}''}, \tilde{s}_{\mathsf{V}''}$ are uniform in $\mathbb{Z}_p$. This matches the definition of a private key for V''. Algorithm $\mathcal{B}$ then derives a private key for the requested node V properly and gives $\mathcal{A}$ the result.

Challenge. When $\mathcal{A}$ decides that Phase 1 is over, it outputs two messages $M_0, M_1 \in \mathbb{G}_T$. To generate the challenge, $\mathcal{B}$ picks a random bit $b \in \{0,1\}$ and responds with ciphertext:

$$C = \Big(Z \cdot M_b,\ g_3,\ \big(g_3^{f(H(\mathsf{W}))} \big)_{\mathsf{W} \preccurlyeq \mathsf{V}^\star} \Big).$$

For all $\mathsf{W} \preccurlyeq \mathsf{V}^\star$, since $q(H(\mathsf{W})) = 0$ we have that $g_3^{f(H(\mathsf{W}))} = F(H(\mathsf{W}))^c$. Therefore, if $Z = e(g,g)^{abc} = e(g_1, g_2)^c$ then C is a valid encryption of M_b for node $\mathsf{V}^\star$. On the other hand, when Z is uniform and independent in $\mathbb{G}_T$ then C is independent of b for the adversary's view.

Phase 2. $\mathcal{A}$ continues to ask queries not issued in Phase 1. $\mathcal{B}$ responds as before.

Guess. Finally, $\mathcal{A}$ outputs $b' \in \{0,1\}$ for guessing b. If $b = b'$ then $\mathcal{B}$ outputs 1 (meaning $Z = e(g,g)^{abc}$). Otherwise, it outputs 0 (meaning Z is random in $\mathbb{G}_T$).

We see that if (g, g_1, g_2, g_3, Z) is sampled from $\mathcal{R}_{BDH}$ then $\Pr[\mathcal{B}(g, g_1, g_2, g_3, Z) = 0] = \frac{1}{2}$. On the other hand, if (g, g_1, g_2, g_3, Z) is sampled from $\mathcal{P}_{BDH}$ then $|\Pr[\mathcal{B}(g, g_1, g_2, g_3, Z) = 0] - \frac{1}{2}| \geq \epsilon$. It follows that $\mathcal{B}$ has advantage at least ϵ in solving the DBDH problem in $\mathbb{G}$. This concludes the proof. □

4.7 Efficient AND Graph DAGE Construction

In this section, we construct an efficient DAGE construction for monotone AND-graph from bilinear pairing. Similar to Construction 4-3 for the case of OR-graph, our AND-graph scheme also enjoys constant-size ciphertext.

Let $G = (V, H)$ be a LAD that all access structures at hyper-edges are of AND-type, *i.e.,* a monotone AND graph. (Note that thus one can also view G as a DAG). Its ID-based extension $\mathcal{G}_{\mathsf{Idx}(G)} = (\mathcal{V}, \mathcal{H})$ defined from the base DAG G is given by

$$\mathcal{V} = \left\{ \Big(\mathsf{v}, (\mathsf{id}_\mathsf{w})_{\mathsf{w} \preccurlyeq \mathsf{v}} \Big) \mid \mathsf{v} \in V;\ \forall \mathsf{w} \preccurlyeq \mathsf{v},\ \text{if } \mathsf{w} \in V^0 \text{ then } \mathsf{id}_\mathsf{w} = \varepsilon \text{ else } \mathsf{id}_\mathsf{w} \in \mathcal{I} \right\}$$

$$\mathcal{H} = \Bigg\{ \Big\{ \Big(\mathsf{v}_1, (\mathsf{id}_\mathsf{w})_{\mathsf{w} \preccurlyeq \mathsf{v}_1} \Big), \ldots, \Big(\mathsf{v}_j, (\mathsf{id}_\mathsf{w})_{\mathsf{w} \preccurlyeq \mathsf{v}_j} \Big) \Big\} \xrightarrow{\wedge} \Big(\mathsf{v}^*, (\mathsf{id}_\mathsf{w})_{\mathsf{w} \preccurlyeq \mathsf{v}^*} \Big) \Bigg| \Big(\{\mathsf{v}_1, \ldots, \mathsf{v}_j\} \xrightarrow{\wedge} \mathsf{v}^* \Big) \in H;\ \forall \mathsf{w} \preccurlyeq \mathsf{v}^*,\ \text{if } \mathsf{w} \in V^0 \text{ then } \mathsf{id}_\mathsf{w} = \varepsilon \text{ else } \mathsf{id}_\mathsf{w} \in \mathcal{I} \Bigg\}.$$

The construction will depend heavily on the structure of the base DAG G. Let $\mathcal{P}$ be the set of all directed paths from some source node to some sink node in G. For each $\mathsf{v} \in V$, let P_v be the set of all paths in $\mathcal{P}$ that pass v. Define a function $t : \mathcal{P} \to \mathbb{Z}^+$ so that the following properties hold.

1. For all *non-source* nodes $\mathsf{v} \in V$, for all two different paths $S, S' \in P_\mathsf{v}$, we have $t(S) \neq t(S')$.
2. For all two *source* nodes $\mathsf{v}, \mathsf{v}' \in V$, we have $t(P_\mathsf{v}) \cap t(P_{\mathsf{v}'}) = \emptyset$.

Here we let $t(P_\mathsf{v})$ be $\{t(S) \mid S \in P_\mathsf{v}\}$. In the following construction, the size of range $|t(\mathcal{P})|$ will determine the public key size; therefore, t is constructed in such a way that $|t(\mathcal{P})|$ is minimized. Wlog, for simplicity, we can choose $t(\mathcal{P}) = \{1, 2, \ldots, |t(\mathcal{P})|\}$. An example of assignment for $t(\cdot)$ is shown in Figure 4-1.

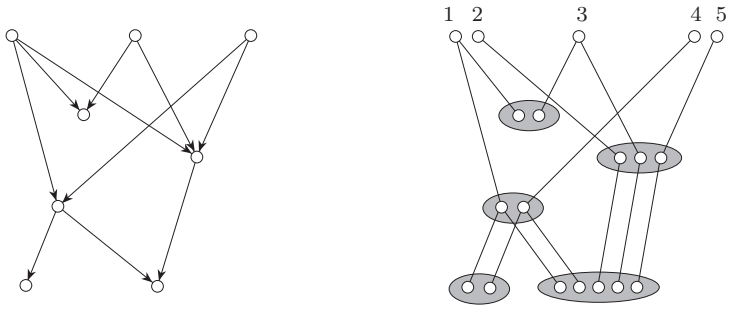


Figure 4-1: A path function $t(\cdot)$ for an accumulated DAG G

Accumulated DAG. For simplicity, we first consider a special case of DAG. An *accumulated DAG* G is defined as a DAG such that there is no tuple of nodes (v_1, v_2, v_3, v_4) such that $v_1 \prec v_2 \prec v_4$ and $v_1 \prec v_3 \prec v_4$. Consequently, one can always define $t(\cdot)$ so that for all $v \in V_G$, $t(P_v) = \bigcup_{w \prec_c v} t(P_w)$. This property enables a kind of aggregation for secret keys.

Construction 4-5. Accumulated AND Graph DAGE with Constant-Size Ciphertext

$\mathsf{GlobSetup}()$**:** Let $\mathbb{G}$ be a bilinear group of prime order p. The algorithm picks a random generator $g \in \mathbb{G}$. Pick randomly $\hat{g}, y \in \mathbb{G}$ and for each $v \in V$ pick randomly $h_v \in \mathbb{G}$. Let $H : V \times \mathcal{I} \to \mathbb{Z}_p$ be a collision-resistant hash function. The global public key is given by

$$\mathsf{gpk} = \Big(\; g, \; \hat{g}, \; y, \; \big(h_v\big)_{v \in V_G}, \; H \; \Big)$$

$\mathsf{Setup}()$**:** The setup algorithm at source node v is done as follows. It first picks random $\alpha_i \in \mathbb{Z}_p$ for each $i \in t(P_v)$. Let $g_i = g^{\alpha_i}$ and $\hat{g}_i = \hat{g}^{\alpha_i}$. The public key and the master key are given by

$$\mathsf{pk}_v = \Big(\; \mathsf{gpk}, \; \big(g_i\big)_{i \in t(P_v)} \; \Big), \quad \mathsf{msk}_v = \big(\hat{g}_i\big)_{i \in t(P_v)}.$$

$\mathsf{Extract}()$**:** A private key of vertex $\mathsf{V} = \big(v, \{(w, \mathsf{id}_w)\}_{w \preccurlyeq v}\big)$ will be of following the form $\mathsf{sk}_{\mathsf{V}} = \Big(d, z, \big\{(w, d_w)\big\}_{\substack{w \not\preccurlyeq v \\ w \in V_G}}\Big)$, where

$$d = \left(\prod_{i \in t(P_v)} \hat{g}_i \right) \cdot \left(y \prod_{w \preccurlyeq v} h_w^{H(w, \mathsf{id}_w)} \right)^s, \quad z = g^s, \quad d_w = h_w^s.$$

To generate $\mathsf{sk}_{(\mathsf{V} \rightarrowtail \mathsf{V}^*)}$ for $\mathsf{V}^* = \big(v^*, \{(w, \mathsf{id}_w)\}_{w \preccurlyeq v^*}\big)$ from sk_{V} where $\mathsf{V} \prec_{\mathsf{c}} \mathsf{V}^*$, first pick

a random elements $\delta \in \mathbb{Z}_p$. It then lets

$$d_{(\mathsf{V}\rightarrowtail\mathsf{V}^*)} = d \cdot \left(\prod_{\substack{w \preccurlyeq v^* \\ w \not\preccurlyeq v}} d_w^{H(w,\mathsf{id}_w)} \right) \cdot \left(y \prod_{w \preccurlyeq v^*} h_w^{H(w,\mathsf{id}_w)} \right)^{\delta},$$
$$z_{(\mathsf{V}\rightarrowtail\mathsf{V}^*)} = z \cdot g^{\delta},$$
$$d_{(\mathsf{V}\rightarrowtail\mathsf{V}^*),w} = d_w \cdot h_w^{\delta}.$$

It outputs $\mathsf{sk}_{(\mathsf{V}\rightarrowtail\mathsf{V}^*)} = \left(d_{(\mathsf{V}\rightarrowtail\mathsf{V}^*)}, z_{(\mathsf{V}\rightarrowtail\mathsf{V}^*)}, \left\{ (w, d_{(\mathsf{V}\rightarrowtail\mathsf{V}^*),w}) \right\}_{\left[\substack{w \not\preccurlyeq v^* \\ w \in V_G}\right]} \right)$.

Combine(): From $\left(\mathsf{sk}_{(\mathsf{V}\rightarrowtail\mathsf{V}^*)} \right)_{\mathsf{V}:\ \mathsf{V} \prec_c \mathsf{V}^*}$, it computes

$$d^* = \prod_{\mathsf{V}:\ \mathsf{V} \prec_c \mathsf{V}^*} d_{(\mathsf{V}\rightarrowtail\mathsf{V}^*)}, \quad z^* = \prod_{\mathsf{V}:\ \mathsf{V} \prec_c \mathsf{V}^*} z_{(\mathsf{V}\rightarrowtail\mathsf{V}^*)}, \quad d_w^* = \prod_{\mathsf{V}:\ \mathsf{V} \prec_c \mathsf{V}^*} d_{(\mathsf{V}\rightarrowtail\mathsf{V}^*),w},$$

for $w \not\preccurlyeq v^*$ (for the last equation). It outputs $\mathsf{sk}_{\mathsf{V}^*} = \left(d^*, z^*, \left\{ (w, d_w^*) \right\}_{\left[\substack{w \not\preccurlyeq v^* \\ w \in V_G}\right]} \right)$.

Encrypt(): To encrypt a message $M \in \mathbb{G}_T$, pick a random $r \in \mathbb{Z}_p$ and output

$$C = \left(e\Big(\prod_{i \in t(P_{v^*})} g_i, \hat{g} \Big)^r \cdot M,\ g^r,\ \left(y \prod_{w \preccurlyeq v^*} h_w^{H(w,\mathsf{id}_w)} \right)^r \right).$$

Decrypt(): Consider a vertex V. To decrypt a given ciphertext $C = (C_1, C_2, C_3)$ using the private key $\mathsf{sk}_{\mathsf{V}} = \left(d, z, \left\{ (w, d_w) \right\}_{\left[\substack{w \not\preccurlyeq v \\ w \in V_G}\right]} \right)$, output

$$C_1 \cdot \frac{e(C_3, z)}{e(C_2, d)} = M$$

General DAG. Now we briefly describe how to adapt the above construction to the case of general DAG. This time, there might be tuple of nodes (v_1, v_2, v_3, v_4) such that $v_1 \prec v_2 \prec v_4$ and $v_1 \prec v_3 \prec v_4$. An illustration is given in Figure 4-2.

The only differences from the construction for accumulated DAG are the secret keys part: the element d will not "aggregate" key components taken from all indexes of $t(P_v)$ into one element, but is separated to many elements depended on its predecessors.

Let $T(v) = \{t(P_v) \cap t(P_w) \mid v \prec w\}$. Let $T_0(v)$ be the set of minimal elements of $T(v)$. The secret key of node $\mathsf{V} = \left(v, \{(w, \mathsf{id}_w)\}_{w \preccurlyeq v}\right)$ will be of following the form:

$$\mathsf{sk}_{\mathsf{V}} = \left(d_{(S)}, z_{(S)}, \left\{ (w, d_{(S),w}) \right\}_{\left[\substack{w \not\preccurlyeq v \\ w \in V_G}\right]} \right)_{S \in T_0(v)},$$

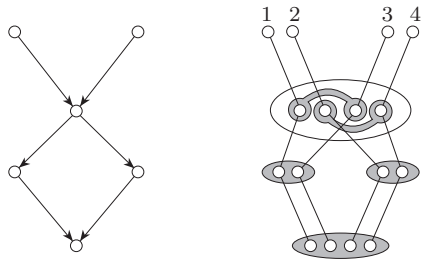


Figure 4-2: A path function $t(\cdot)$ for a general DAG G

where

$$d_{(S)} = \left(\prod_{i\in S} \hat{g}_i\right) \cdot \left(y \prod_{w \preccurlyeq v} h_w^{H(w,\mathsf{id}_w)}\right)^{s_{(S)}}, \quad z_{(S)} = g^{s_{(S)}}, \quad d_{(S),w} = h_w^{s_{(S)}}.$$

The algorithms GlobSetup,Setup,Encrypt are exactly the same, while Extract,Combine,Decrypt can be generalized from the accumulated DAG case straightforwardly. We omit the detail here.

Theorem 4.19. *Suppose that the Decision $|\mathcal{V}|$-BDHE assumption holds in $\mathbb{G}$. Then the above scheme is IND-$(\mathsf{s},\mathsf{f}|\ \mathcal{G}_{\mathsf{Idx}(G)}, (\Sigma_{\mathcal{G}_{\mathsf{Idx}(G)}})^{\mathsf{c}})$-CPA-secure for any AND-graph G.*

The proof can be generalized from that of Theorem 4.17.

4.8 Prototype Functionalities

In this section, we give some prototype applications of DAGE.

General Methodology. To define a primitive, we define a class of LADs for the primitive, which we will call definitional LADs since they give the definition of syntax and security notion. This step must be intuitive enough so that it captures the purpose of the primitive in question in a natural way. In other words, such a primitive should be described solely by the graph, with as few as possible "meta-syntax", which is a description about the functions of key derivations at various hyper-edges in the LAD when the scheme is used in the real application.

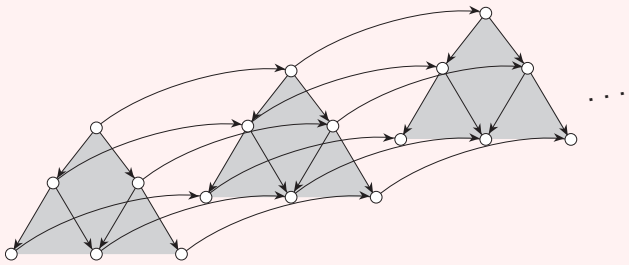
To construct a scheme for that primitive, we find another LAD that graph syntactically implies the definitional LAD and possesses some good properties (*e.g.,* , achieving better efficiency or using more simple primitives) than directly working on the definitional LAD. Such a LAD will be called a constructional LAD. Then the DAGE scheme for this constructional LAD is constructed. By Theorem 4.9, we convert this to DAGE scheme of the definitional LAD, and we are done.

Notation in Graphs. In this section, we will draw many graphs to illustrate schemes. We let the dot line represent that its adjacent nodes are exactly the identical node. (We write some dot lines for some ease of visuality). The ellipse over more than one nodes represent multiple encryption on those nodes (with some access structures which should be clear from the context).

4.8.1 Forward-secure Functionality

In a forward-secure encryption scheme, a user maintains a fixed public key, but it evolves its private key forward in such a way that, if its private key for time period t is compromised, the security for previously encrypted message (*i.e.,* for time period $t-1$ or earlier) is still guaranteed. Note that however, all future encrypted messages will be vulnerable since the attacker can update the compromised key by itself. Canetti, Halevi, Katz [CHK03] constructed the first non-trivial forward-secure public-key encryption based on HIBE scheme of Gentry and Silverberg [GS02]. Yao et al. [YFDL04] consequently formalized and constructed forward-secure HIBE. In this work, we give the definitional LAD for forward-secure version of any DAGE. This thus gives the formalization and construction of forward-secure DAGE via our unified definition and constructions. Let $\mathcal{G} = (\mathcal{V}, \mathcal{H})$ be a LAD, the definitional LAD of its forward-secure version, $\mathsf{F}_{\mathsf{Def\text{-}FS}}(\mathcal{G})$, is shown in Graph 1. We denote by $\circ$ the concatenation of strings separated by a comma. When X is a set, we let $t \circ X = \{t \circ x \mid x \in X\}$.

Graph 1. Definition of Forward-secure DAGE: $\mathsf{F}_{\mathsf{Def\text{-}FS}}(\mathcal{G}) = (\mathcal{V}', \mathcal{H}')$



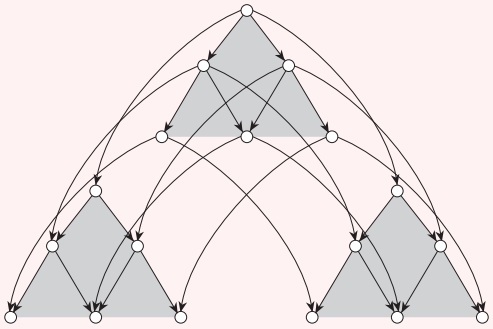
$$\mathcal{V}' = \left\{ (t, \mathsf{V}) \;\middle|\; t \in \mathbb{Z}_{\geq 0};\ \mathsf{V} \in \mathcal{V} \right\}$$

$$\mathcal{H}' = \left\{ \left\{(t, \mathsf{V}^*)\right\} \cup (t+1) \circ \mathbf{V} \xrightarrow{\left\{(t,\mathsf{V}^*)\right\} \ \cup\ (t+1)\circ\Theta_{\mathsf{V}^*}} (t+1) \circ \mathsf{V}^* \;\middle|\; \left(\mathbf{V} \xrightarrow{\Theta_{\mathsf{V}^*}} \mathsf{V}^*\right) \in \mathcal{H} \right\}$$

The description is quite intuitive: it states that "A node V updates its own private key from time t to $t+1$ *or* some group in the qualified access structure (as specified by Θ_{V} in $\mathcal{G}$) just derives the key for V instantly". This is exactly the property of forward-secure version of LAD $\mathcal{G}$. Note that one may specify a-priori fixed T to be the maximum time period.

Although one can directly implement $\mathsf{F}_{\mathsf{Def\text{-}FS}}(\mathcal{G})$-DAGE by the unified constructions. A problem concerning efficiency is that the length of the longest directed path to node (t, V) is proportional to t, which could make private key size as large as $O(t)$. An efficient constructional LAD which graph syntactically implies $\mathsf{F}_{\mathsf{Def\text{-}FS}}(\mathcal{G})$ can be constructed by generalizing the time-tree approach from [CHK03]. Our constructional LAD, $\mathsf{F}_{\mathsf{Con\text{-}FS}}(\mathcal{G})$, is given as follows.

Graph 2. A Construction of Forward-secure DAGE: $\mathsf{F}_{\mathsf{Con\text{-}FS}}(\mathcal{G}) = (\mathcal{V}'', \mathcal{H}'')$



$$\mathcal{V}'' = \Big\{ (w, \mathsf{V}) \;\Big|\; w \in \{0,1\}^{\leq m};\ \mathsf{V} \in \mathcal{V} \Big\}$$

$$\mathcal{H}'' = \Big\{ \big\{(w, \mathsf{V}^*)\big\} \cup (w||b) \circ \mathbf{V} \xrightarrow{\big\{\{(w,\mathsf{V}^*)\}\big\} \,\cup\, (w||b)\circ\Theta_{\mathsf{V}^*}} (w||b) \circ \mathsf{V}^* \;\Big|\; \big(\mathbf{V} \xrightarrow{\Theta_{\mathsf{V}^*}} \mathsf{V}^*\big) \in \mathcal{H},\ b \in \{0,1\} \Big\}$$

To state the maps that enable to prove the graph syntactic consequence, we first give some terminology. Let T be the maximum time period. We image a complete balance binary tree of depth $\ell = \log_2(T+1) - 1$. Let each node be assigned with a string in $\{0,1\}^{\leq \ell}$. We assign the root with the empty string ε. The left and right child of w is assigned $w||0$ and $w||1$ respectively. Following the notation in [CHK03], we let w^t to be the t-th node in a pre-order traversal of the binary tree.[2] We let $\mathsf{Hang}(w)$ be the set of the right sibling of nodes on the path from the root to w in the binary tree.

Theorem 4.20. $\mathsf{F}_{\mathsf{Con\text{-}FS}}(\mathcal{G}) \Vdash_{\mathsf{syn}} \mathsf{F}_{\mathsf{Def\text{-}FS}}(\mathcal{G})$ *via the following maps* (σ, δ).

$$\begin{array}{rrcl} \sigma: & \mathcal{V}' & \to & \mathcal{AS}^{(\mathcal{V}'')} \\ & (t, \mathsf{V}) & \mapsto & \{\{(w, \mathsf{V}) \mid w \in \mathsf{Hang}(w^t)\}\} \\ \delta: & \mathcal{V}' & \to & \mathcal{AS}^{(\mathcal{V}'')} \\ & (t, \mathsf{V}) & \mapsto & \{\{(w^t, \mathsf{V})\}\} \end{array}$$

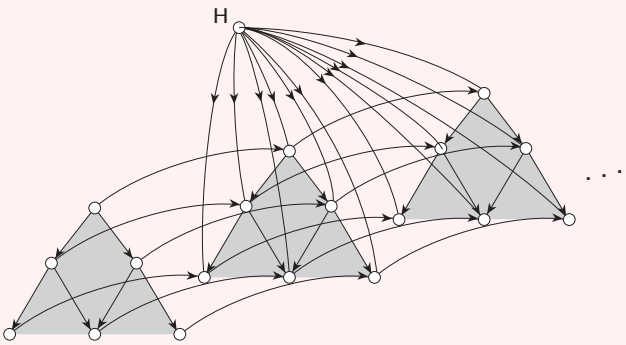
4.8.2 Key-insulated, Intrusion-resilient Functionality

Key-insulated DAGE. A key-insulated encryption scheme is a further-strengthened version of forward-secure encryption where updating key must be done via the collaboration of two modules, one is called a *helper server* and the other one is the user itself, instead of only the user in the forward-secure setting. Key-insulated public-key encryption was introduced in [DK02]. We generalize to DAGE.

The definitional LAD of key-insulated version of LAD $\mathcal{G} = (\mathcal{V}, \mathcal{H})$, which we will denote by $\mathsf{F}_{\mathsf{Def\text{-}KI}}(\mathcal{G})$, is similar to $\mathsf{F}_{\mathsf{Def\text{-}FS}}(\mathcal{G})$ with only the helper node 'H' included. It is defined as follows.

[2] The pre-order traversal is started from the root, $w^1 = \varepsilon$. From a node w it goes to $w||0$ if w is not a leaf otherwise it goes to $v||1$ such that $v||0$ is the largest string with 0 at the end that is a prefix of w.

Graph 3. Definition of Key-insulated DAGE: $\mathsf{F}_{\mathsf{Def\text{-}KI}}(\mathcal{G}) = (\mathcal{V}', \mathcal{H}')$



$$\mathcal{V}' = \{\text{`H'}\} \cup \Big\{(t, \mathsf{V}) \;\Big|\; t \in \mathbb{Z}_{\geq 0};\ \mathsf{V} \in \mathcal{V}\Big\}$$

$$\mathcal{H}' = \Big\{\big\{\text{`H'}, (t, \mathsf{V}^*)\big\} \cup (t+1) \circ \mathbf{V} \xrightarrow{\left\{\{\text{`H'},(t,\mathsf{V}^*)\}\right\} \cup (t+1)\circ\Theta_{\mathsf{V}^*}} (t+1) \circ \mathsf{V}^* \;\Big|\; \big(\mathbf{V} \xrightarrow{\Theta_{\mathsf{V}^*}} \mathsf{V}^*\big) \in \mathcal{H}\Big\}$$

The definition above can be described intuitively as private keys can be derived hierarchically or 'H' and V must collaborate to update the key of V.

Many variants of key-insulated encryption can be considered. The first categorization is due to random updatability, which is the ability that the helper and a node (t, V) can produce the key for node (t', V) for any t'.

- Scheme with random update function. In this case, we redefine the hyper-edge set to

$$\tilde{\mathcal{H}}' = \Big\{\big\{\text{`H'}, (t, \mathsf{V}^*) \mid t \neq t^*\big\} \cup (t^*) \circ \mathbf{V} \xrightarrow{\left\{\{\text{`H'},(t,\mathsf{V}^*)\} \mid t\neq t^*\right\} \cup (t^*)\circ\Theta_{\mathsf{V}^*}} (t^*) \circ \mathsf{V}^* \;\Big|\; \big(\mathbf{V} \xrightarrow{\Theta_{\mathsf{V}^*}} \mathsf{V}^*\big) \in \mathcal{H};\ t^* \in \mathbb{Z}_{\geq 0}\Big\}.$$

  Then the definitional LAD for this case is $\mathsf{F}_{\mathsf{Def\text{-}RandKI}}(\mathcal{G}) = (\mathcal{V}', \tilde{\mathcal{H}}')$.

- Scheme with don't care condition for random update. The definition is exactly the same as the standard one, $\mathsf{F}_{\mathsf{Def\text{-}KI}}(\mathcal{G})$, but with a weaker notion $\mathsf{IND}\text{-}(\mathsf{a}, \mathsf{n}|\ \mathsf{F}_{\mathsf{Def\text{-}KI}}(\mathcal{G}), \Delta)\text{-}\mathsf{CCA}$ where $\Delta = (\Sigma_{\mathsf{F}_{\mathsf{Def\text{-}RandKI}}(\mathcal{G})})^{\mathsf{c}} \subset (\Sigma_{\mathsf{F}_{\mathsf{Def\text{-}KI}}(\mathcal{G})})^{\mathsf{c}}$. Note that this variant is implied by the previous one but not the reverse.

- Scheme secure with forward-only update. This is exactly the standard one with the full structure of the negative underivability.

Another categorization is due to the security against malicious helper.

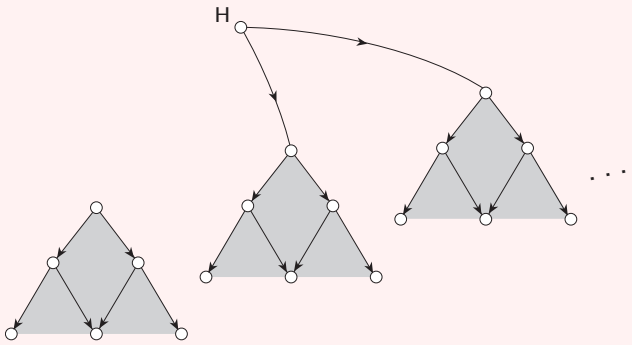
- The *strongly* key-insulated notion. This is exactly the notion $\mathsf{IND}\text{-}(\mathsf{a}, \mathsf{n}|\ \mathsf{F}_{\mathsf{Def\text{-}KI}}(\mathcal{G}), \Sigma^{\mathsf{c}}_{\mathsf{F}_{\mathsf{Def\text{-}KI}}(\mathcal{G})})\text{-}\mathsf{CCA}$.

- The *weakly* key-insulated notion. This is exactly the notion $\mathsf{IND}\text{-}(\mathsf{a}, \mathsf{n}|\ \mathsf{F}_{\mathsf{Def\text{-}KI}}(\mathcal{G}), \Delta)\text{-}\mathsf{CCA}$ where $\Delta = (\Sigma_{\mathsf{F}_{\mathsf{Def\text{-}KI}}(\mathcal{G})} \cup A)^{\mathsf{c}} \subseteq \Sigma_{\mathsf{F}_{\mathsf{Def\text{-}KI}}(\mathcal{G})}{}^{\mathsf{c}}$ where $A = \{(\langle\!\langle \text{`H'} \rangle\!\rangle \Rightarrow \langle\!\langle (t, \mathsf{V}) \rangle\!\rangle) \mid t \in \mathbb{Z}_{\geq 1};\ \mathsf{V} \in \mathcal{V}\}$, *i.e.,* the security against the exposure of helper's key is not guaranteed.

Up to now, we consider only the normal security level, *i.e.,* the exposure of partial keys is not taken into account. The notion for key-insulated encryption with security against partial key exposure is $\mathsf{IND}\text{-}(\mathsf{a},\mathsf{f}|\ \mathsf{F}_{\mathsf{Def\text{-}KI}}(\mathcal{G}), \Sigma^{\mathsf{c}}_{\mathsf{F}_{\mathsf{Def\text{-}KI}}(\mathcal{G})})\text{-}\mathsf{CCA}$.

We now give two example of combinations from the above variants. The first one is the weakest one among them, hence a simple structure of HIBE is sufficient to construct as shown below.

Graph 4. A Construction of Weakly Key-insulated DAGE with Random Update: $\mathsf{F}_{\mathsf{Con\text{-}wRandKI}}(\mathcal{G}) = (\mathcal{V}'', \mathcal{H}'')$

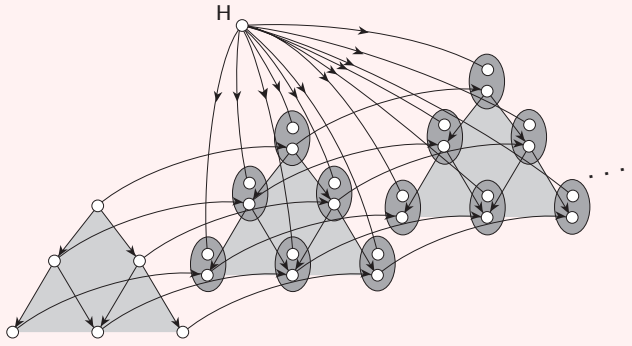


$$\mathcal{V}'' = \{\text{`H'}\} \cup \left\{(t, \mathsf{V}) \;\middle|\; t \in \mathbb{Z}_{\geq 0};\ \mathsf{V} \in \mathcal{V}\right\}$$

$$\mathcal{H}'' = \left\{t \circ \mathbf{V} \xrightarrow{t \circ \Theta_{\mathsf{V}^*}} (t, \mathsf{V}^*) \middle|\ \left(\mathbf{V} \xrightarrow{\Theta_{\mathsf{V}^*}} \mathsf{V}^*\right) \in \mathcal{H};\ t \in \mathbb{Z}_{\geq 0}\right\} \cup \left\{\text{`H'} \to (t, \mathsf{V}) \;\middle|\; \mathsf{V} \in \mathcal{V}^0;\ t \in \mathbb{Z}_{\geq 1}\right\}$$

The second example is a constructional LAD of strongly key-insulated DAGE with forward-only update. It can be constructed from a forward-secure $\mathcal{G}$-DAGE and an IBE as shown below.

Graph 5. A Construction of Strongly Key-insulated DAGE with Forward-only Update: $\mathsf{F}_{\mathsf{Con\text{-}wRandKI}}(\mathcal{G}) = (\mathcal{V}'', \mathcal{H}'')$



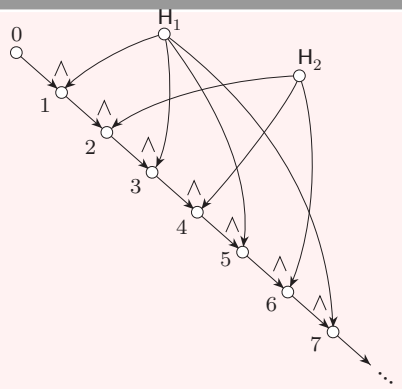
$$\mathcal{V}'' = \mathcal{V}_{\mathsf{F}_{\mathsf{Def\text{-}FS}}(\mathcal{G})} \cup \{\text{`H'}\} \cup \left\{\overline{(t, \mathsf{V})} \;\middle|\; t \in \mathbb{Z}_{\geq 1};\ \mathsf{V} \in \mathcal{V}\right\}$$

$$\mathcal{H}'' = \mathcal{H}_{\mathsf{F}_{\mathsf{Def\text{-}FS}}(\mathcal{G})} \cup \left\{\text{`H'} \to \overline{(t, \mathsf{V})} \;\middle|\; \mathsf{V} \in \mathcal{V};\ t \in \mathbb{Z}_{\geq 1}\right\}$$

To change the forward-only update option to random update, one can simply change the underlying forward-secure $\mathcal{G}$-DAGE to the normal $\mathcal{G}$-DAGE instead.

Parallel Key-insulated PKE. Parallel key-insulated encryption [HHI06] is an extension of key-insulated PKE with two helpers who exactly alternately involve in the updating turn by turn. We can indeed consider general parallel key-insulated DAGE, but for simplicity we consider the case of PKE.

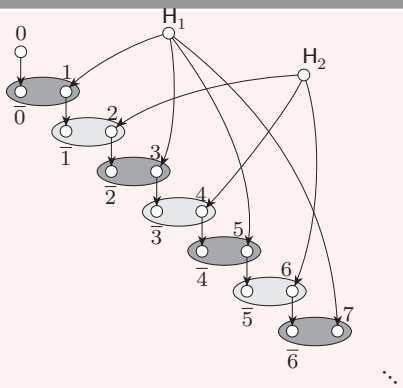
Graph 6. Definition of Parallel Key-insulated PKE: $\mathcal{G}_{\mathsf{Def\text{-}2KI}} = (\mathcal{V}, \mathcal{H})$



$$\mathcal{V} = \{\text{`H}_1\text{'}, \text{`H}_2\text{'}\} \cup \{t \mid t \in \mathbb{Z}_{\geq 0}\}$$
$$\mathcal{H} = \Big\{\{\text{`H}_1\text{'}, t-1\} \xrightarrow{\wedge} t \;\Big|\; t \in \mathbb{Z}_{\geq 1};\ t \bmod 2 = 1\Big\} \cup \Big\{\{\text{`H}_2\text{'}, t-1\} \xrightarrow{\wedge} t \;\Big|\; t \in \mathbb{Z}_{\geq 2};\ t \bmod 2 = 0\Big\}$$

A constructional LAD is shown below. It turns out that 2-level HIBE is sufficient to construct parallel key-insulated PKE. This is the first known generic construction.

Graph 7. A construction of Parallel Key-insulated PKE: $\mathcal{G}_{\mathsf{Con\text{-}2KI}} = (\mathcal{V}', \mathcal{H}')$



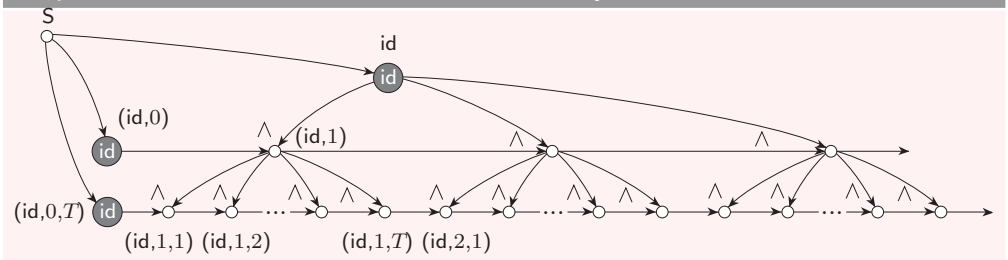
$$\mathcal{V}' = \{\text{`H}_1\text{'}, \text{`H}_2\text{'}\} \cup \{t,\ \bar{t} \mid t \in \mathbb{Z}_{\geq 0}\}$$
$$\mathcal{H}' = \Big\{\{\text{`H}_1\text{'}\} \to t \;\Big|\; t \in \mathbb{Z}_{\geq 1};\ t \bmod 2 = 1\Big\} \cup \Big\{\{\text{`H}_2\text{'}\} \to t \;\Big|\; t \in \mathbb{Z}_{\geq 2};\ t \bmod 2 = 0\Big\} \cup \{t \to \bar{t} \mid t \in \mathbb{Z}_{\geq 0}\}$$

Theorem 4.21. $\mathcal{G}_{\mathsf{Con\text{-}2KI}} \Vdash_{\mathsf{syn}} \mathcal{G}_{\mathsf{Def\text{-}2KI}}$ *via the map* $\sigma \equiv \delta$ *where*

$$\begin{array}{rccll} \sigma: & \mathcal{V} & \to & \mathcal{AS}^{(\mathcal{V}')} & \\ & \text{`}\mathsf{H}_i\text{'} & \mapsto & \{\{\text{`}\mathsf{H}_i\text{'}\}\} & \text{for } i = 1,2 \\ & t & \mapsto & \begin{cases} \{\{0\}\} & \text{for } t = 0,1 \\ \{\{\overline{t-1}, t\}\} & \text{for } t \geq 2 \end{cases} & \end{array}$$

ID-based Hierarchical Key-insulated PKE. ID-based Hierarchical Key-insulated PKE scheme [HHSI05] extends the normal key-insulated PKE by (1) enabling another upper level modules to update the private key of lower-level helper servers and (2) changing one-user setting to ID-based setting (helper is also ID-specific). This can be viewed as the application of ID-based functionality over hierarchical key-insulation functionality. For simplicity, we consider only 2 levels of helper, but this can be generalized straightforwardly. The definitional graph is as follows.

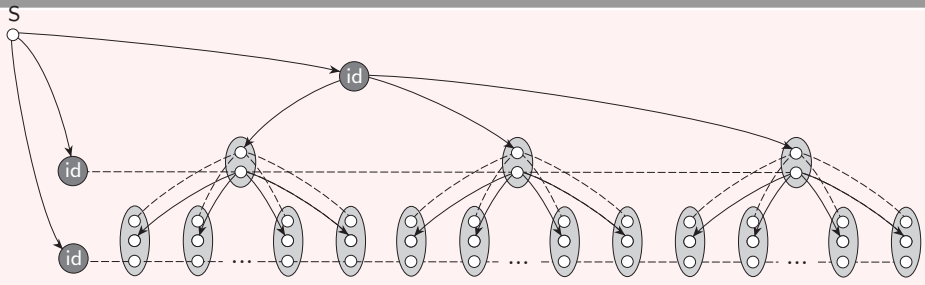
Graph 8. Definition of ID-based Hierarchical Key-insulated PKE



$$\begin{aligned} \mathcal{V} =& \{\text{`S'}\} \cup \{\mathsf{id} \mid \mathsf{id} \in \mathcal{I}\} \cup \{(\mathsf{id}, t_1) \mid \mathsf{id} \in \mathcal{I};\ \ t_1 \in \mathbb{Z}_{\geq 0}\} \cup \\ & \{(\mathsf{id}, t_1, t_2) \mid \mathsf{id} \in \mathcal{I};\ \ t_1 \in \mathbb{Z}_{\geq 0};\ t_2 = 0, \ldots, T\} \\ \mathcal{H} =& \Big\{ \big(\{\text{`S'}\} \to \mathsf{id}\big),\ \big(\{\text{`S'}\} \to (\mathsf{id}, 0)\big),\ \big(\{\text{`S'}\} \to (\mathsf{id}, 0, T)\big) \mid \mathsf{id} \in \mathcal{I} \Big\} \cup \\ & \Big\{ \{\mathsf{id}, (\mathsf{id}, t_1 - 1)\} \xrightarrow{\wedge} (\mathsf{id}, t_1) \mid \mathsf{id} \in \mathcal{I};\ t_1 \in \mathbb{Z}_{\geq 1} \Big\} \cup \\ & \Big\{ \{(\mathsf{id}, t_1), (\mathsf{id}, t_1, t_2 - 1)\} \xrightarrow{\wedge} (\mathsf{id}, t_1, t_2) \mid \mathsf{id} \in \mathcal{I};\ t_1 \in \mathbb{Z}_{\geq 0};\ t_2 = 1, \ldots, T \Big\} \cup \\ & \Big\{ \{(\mathsf{id}, t_1 - 1), (\mathsf{id}, t_1 - 1, T)\} \xrightarrow{\wedge} (\mathsf{id}, t_1, 0) \mid \mathsf{id} \in \mathcal{I};\ t_1 \in \mathbb{Z}_{\geq 1} \Big\} \end{aligned}$$

A constructional LAD for strongly key-insulated scheme is given below. In contrast with the decomposition to HIBE in [HHSI05] where their result shows that $n+1$-level HIBE is sufficient for scheme with n levels of helpers, ours construction needs only 2-level HIBE for scheme with any levels of helpers.

Graph 9. A Construction of ID-based Hierarchical Key-insulated PKE



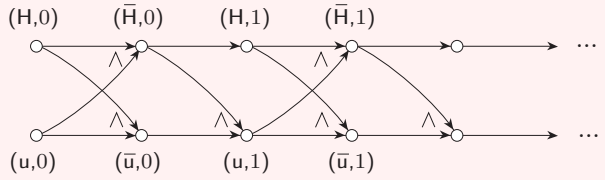
$$\mathcal{V} = \{\text{`S'}\} \cup \{(i, \mathsf{id}) \mid i = 0, 1, 2;\ \mathsf{id} \in \mathcal{I}\} \cup \{(i, \mathsf{id}, t_{i+1}) \mid i = 0, 1;\ \mathsf{id} \in \mathcal{I};\ \ t_1 \in \mathbb{Z}_{\geq 0};\ t_2 = 0, \ldots, T\}$$

$$\mathcal{H} = \Big\{ \big(\{\text{`S'}\} \to (i, \mathsf{id})\big) \ \Big|\ i = 0, 1, 2;\ \mathsf{id} \in \mathcal{I} \Big\} \cup \Big\{ \{(i, \mathsf{id})\} \to (i, \mathsf{id}, t_{i+1}) \ \Big|\ i = 0, 1;\ \mathsf{id} \in \mathcal{I};\ t_1 \in \mathbb{Z}_{\geq 0};\ t_2 = 1, \ldots, T \Big\}$$

Intrusion-resilient PKE. Intrusion-resilient PKE [DFK+03, DFK+04] combines the functionality of forward-security and key-insulation. It extends key-insulated PKE by enabling helper keys also to be updateable. The advantage of this scheme is that its security is guaranteed as long as the adversary does not compromise both modules (helper and user) in the same time period; it retains the security except only at the time where the user keys are compromised. Key-insulated PKE does not provide this since exposure of both modules destroys the security of future time period completely. When both modules are compromised at the same period, it still retains the minimal security of forward security, where the past time period is still protected.

The original paper [DFK+03] presented the formalization where within each time period both modules can proactively refresh their keys. For simplicity, we present the definitional LAD with only one-time refresh (at the cross).

Graph 10. Definition of Our Variant of Intrusion-resilient PKE



$$\mathcal{V} = \{(\text{`H'}, t), (\text{`}\overline{\mathsf{H}}\text{'}, t), (\text{`u'}, t), (\text{`}\overline{\mathsf{u}}\text{'}, t) \mid t \in \mathbb{Z}_{\geq 0}\}$$

$$\mathcal{H} = \Big\{ \{(\text{`}\overline{\mathsf{H}}\text{'}, t-1))\} \to (\text{`H'}, t) \ \Big|\ t \in \mathbb{Z}_{\geq 1} \Big\} \cup \Big\{ \{(\text{`H'}, t-1), (\text{`u'}, t-1)\} \xrightarrow{\wedge} (\text{`}\overline{\mathsf{u}}\text{'}, t) \ \Big|\ t \in \mathbb{Z}_{\geq 1} \Big\} \cup \Big\{ \{(\text{`H'}, t-1), (\text{`u'}, t-1)\} \xrightarrow{\wedge} (\text{`}\overline{\mathsf{H}}\text{'}, t) \ \Big|\ t \in \mathbb{Z}_{\geq 1} \Big\} \cup \Big\{ \{(\text{`}\overline{\mathsf{H}}\text{'}, t-1), (\text{`}\overline{\mathsf{u}}\text{'}, t-1)\} \xrightarrow{\wedge} (\text{`u'}, t) \ \Big|\ t \in \mathbb{Z}_{\geq 1} \Big\}$$

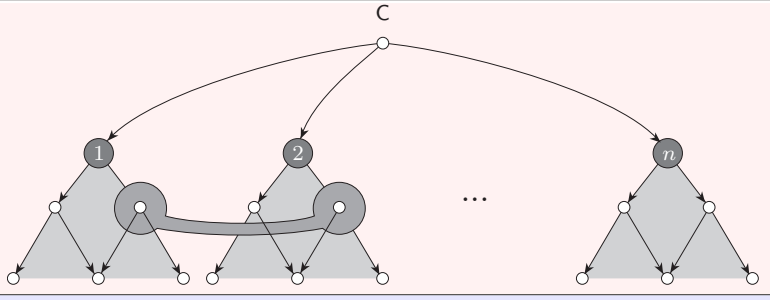
A meta-syntax is that to encrypt to time t use node $(\text{`u'}, t)$. We remark that our

formulation is not exactly equivalent to that of [DFK+03, DFK+04]. However, it captures the same goal described above (in particular, the security is guaranteed unless simultaneous compromises occur).

4.8.3 Broadcast Functionality

Public-key broadcast encryption scheme is broadcast encryption (cf. 3) in which the encrypt key is public. The definitional LAD is $\mathcal{G}_{\mathsf{BE}} = (\mathcal{V}, \mathcal{H})$ where $\mathcal{V} = \{1, \ldots, n\}$, $\mathcal{H} = \emptyset$, with multiple-node collection $\mathcal{P} = \{\{\{i_1\}, ..., \{i_k\}\} \mid \{i_1, ..., i_k\} \subseteq \mathcal{V}\}$. The definitional LAD for broadcast version of any DAGE $\mathcal{G} = (\mathcal{V}, \mathcal{H})$ is described below. It can be viewed as an extension to ID-based DAGE.

Graph 11. Definition of Broadcast DAGE: $\mathsf{F}_{\mathsf{Def\text{-}BE}}(\mathcal{G}) = (\mathcal{V}', \mathcal{H}')$



$$\mathcal{V}' = \{\text{`C'}\} \cup \{(i, \mathsf{V}) \mid i = 1, \ldots, n;\ \mathsf{V} \in \mathcal{V}\}$$
$$\mathcal{H}' = \{\{\text{`C'}\} \to (i, \mathsf{V}) \mid i = 1, \ldots, n;\ \mathsf{V} \in \mathcal{V}^0\} \cup \left\{ i \circ \mathbf{V} \xrightarrow{i \circ \Theta_{\mathsf{V}^*}} (i, \mathsf{V}^*) \;\middle|\; \mathbf{V} \xrightarrow{\Theta_{\mathsf{V}^*}} \mathsf{V}^* \in \mathcal{H} \right\}$$
$$\mathcal{P} = \left\{ \{\{(i, \mathsf{V})\} \mid i \in S\} \mid S \subseteq \{1, \ldots, n\};\ \mathsf{V} \in \mathcal{V} \right\}$$

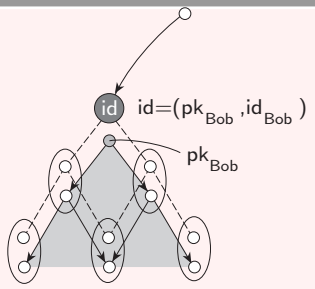
A class of constructional DAGs that we consider is corresponding to subset-cover broadcast encryption.

4.8.4 Certificate-based Functionality

In Certificate-based PKE [Gen03], a user Bob creates his own public/private key pair and also obtains a private key corresponding to his public key and his name, *i.e.,* $(\text{Bob}, \mathsf{pk}_{\text{Bob}})$, in an IBE scheme. A sender will doubly encrypt with Bob's public key and Bob's identity in the IBE scheme. Only Bob, who possess these two keys, can decrypt. The private key from IBE scheme can be viewed of a certificate, thus its name. This scheme preserves a good property from IBE as the sender needs not verify the public key, while at the same time it eliminates the problem of key escrow in the IBE scheme.

As in the case of ID-based functionality, we present certificate-based version of any DAGE $\mathcal{G} = (\mathcal{V}, \mathcal{H})$ as follows. Such a definition essentially employs an IBE scheme $\mathcal{G}_{\mathsf{IBE}} = (\mathcal{V}_{\mathsf{IBE}}, \mathcal{H}_{\mathsf{IBE}})$. When DAGE itself is certificate-based scheme, we get hierarchical certificate-based scheme recursively. The hierarchical certificate-based PKE of [Gen03] is also consistent with our formulation.

Graph 12. Definition of Certificate-based DAGE: $\mathsf{F}_{\mathsf{Def\text{-}CBE}}(\mathcal{G}) = (\mathcal{V}', \mathcal{H}')$



$$\mathcal{V}' = \mathcal{V} \cup \mathcal{V}_{\mathsf{IBE}}$$
$$\mathcal{H}' = \mathcal{H} \cup \mathcal{H}_{\mathsf{IBE}}$$

We add a meta-syntax that to encrypt to $\mathsf{V} \in \mathcal{V}$ in bob's scheme, use multiple encryption for structure $\{\{(\mathsf{pk}_{\mathsf{Bob}}, \mathsf{id}_{\mathsf{Bob}}), \mathsf{pk}_{\mathsf{Bob}}\}\}$.

4.9 Concluding Remarks

We have presented an encryption primitive called directed acyclic graph encryption (DAGE). It works as a unified framework for various "public-key encryption with high functionalities" in the aspects of a unified definition and a unified security notion. We then presented first and second primitive implication theorems. The first theorem (Theorem 4.9) is more general since it gives the criterion for primitive implications for any pair of DAGEs but the security level considered is only normal, *i.e.,* the $\mathsf{IND}\text{-}(\mathsf{x}, \mathsf{n}|\ \mathcal{G}, \Delta)\text{-}\mathsf{Z}$ notion, while the second one (Theorem 4.14) states that any DAGEs can be induced from its related HIBE in the full security notion, *i.e.,* the $\mathsf{IND}\text{-}(\mathsf{x}, \mathsf{f}|\ \mathcal{G}, \Delta)\text{-}\mathsf{Z}$ notion. We then gave three efficient constructions from bilinear maps; two of which achieves constant-size ciphertext, while the other can deal with the most complex graph (bounded-complete type). Finally we gave prototypes for functionalities; one consequence from this is that we can obtain DAGEs of any combinations of functionalities automatically.

We give a remark that very recently proposed encryption primitives called *attribute-based encryption* [GPSW06], which is an extension of *fuzzy ID-based encryption* [SW05], does not fit into our framework in the present form. However, it is possible to extend the framework to a more complex one that will include also this primitive. The merit of doing so is that one can possibly obtain some combinations such as forward-secure attribute-based encryption, while the demerit is that the resulting framework might be too complex. We postpone this as a future work.

Chapter 5

Practical Forward-Secure and Searchable Broadcast Encryption

5.1 Introduction

We first review some facts on broadcast encryption previously described in Chapter 3. Broadcast encryption (BE) scheme [FN93] allows a broadcaster to encrypt a message for some designated subset of all users in the system. Any user in S can decrypt the message by using his own private key while the coalition of all users outside S should not be able to do so even if they collude. The set S can be arbitrary.

Public-key broadcast encryption is the one in which the broadcaster key is a public key. Such a scheme is typically harder to construct than a private-key one. We let n denote the number of all users.

The only scheme that achieves asymptotically optimal sizes, $O(1)$, for both broadcast ciphertexts and user private keys so far in the literature was recently proposed by Boneh, Gentry, and Waters [BGW05]. Their scheme, which is a public-key scheme, however, achieves such a performance with the price of $O(n)$-size public key. (To tradeoff the public key size, they also proposed a generalized scheme that achieves the public key of size $O(\sqrt{n})$, while ciphertext size is increased to $O(\sqrt{n})$). The previously best schemes [NNL01, HS02, GST04] can only achieve a broadcast ciphertext of size $O(r)$ with each user's private key being of size $O(\log n)$, where $r = n - |S|$, the number of revoked users. Although these schemes are improved in [AI05a] by reducing the private key size to $O(1)$, the ciphertext is still of size $O(r)$.[1] These NNL derivatives are originally private-key schemes. Dodis and Fazio [DF02] gives a framework to extend these schemes to public-key versions using Hierarchical Identity-Based Encryption (HIBE) [HL02, GS02]. Instantiating this framework with a recent efficient HIBE scheme by Boneh, Boyen, and Goh [BBG05] gives a public-key version of NNL-based schemes without any loss of performance.

Forward-Secure Broadcast Encryption. Unfortunately, a normal broadcast encryption scheme offers no security protection for any user whatsoever once his private key is compromised. As an extension to the normal variant in order to cope with the vulnerability against key exposure, the notion of forward security [Gün89, And97] in the context of

[1] Note that, as mentioned in Chapter 3, one advantage of these NNL-based schemes is that, in contrast to the BGW scheme, all the other efficiency parameters, beside ciphertext sizes and private key sizes, are also of sub-linear (in n) size.

public-key broadcast encryption was first studied by Yao et al. [YFDL04]. A forward-secure public-key broadcast encryption (FS-BE) allows each user to update his private key periodically while keeping the public key unchanged. Such a scheme guarantees that even if an adversary learns the private key of some user at time period τ, message encrypted during all time periods prior to τ remain secret.

In [YFDL04], Yao et al. proposed a FS-BE scheme achieving ciphertexts of size $O(r \log T \log n)$ while each user's private key is of size $O(\log^3 n \log T)$, where T is the maximum allowed time period. Indeed, they proposed a forward-secure HIBE scheme and then applied it to the NNL scheme in essentially the same manner as done by [DF02], as mentioned above. Later, Boneh et al. [BBG05] proposed (at least two) more efficient forward-secure HIBE schemes, which when applying to the NNL scheme gives a FS-BE scheme with ciphertexts of size $O(r)$ and user private keys of size $O(\log^3 n \log T)$ and another FS-BE scheme with ciphertexts of size $O(r \log T)$ and user private keys of size $O((\log^2 n)(\log n + \log T))$. These schemes are the best FS-BE schemes so far in the literature.

5.1.1 Our Contributions.

Towards constructing a more efficient FS-BE scheme, we introduce a new primitive called *Hierarchical Identity-Coupling Broadcast Encryption* (HICBE), which can be considered as a generalization either of BE that further includes hierarchical-identity dimension together with key derivation functionality or of HIBE that further includes a user dimension together with broadcast functionality. Besides forward security, HICBE can be used to construct BE with other extended properties such as keyword-searchability, which is another feature that we study as a side result in this chapter (see below).

FS-BE with Short Ciphertexts and Private Keys. Using HICBE as a building block, we propose at least three new FS-BE schemes. One of our best two schemes achieves ciphertexts of size $O(1)$ and user private keys of size $O(\log^2 T)$. The other best scheme achieves ciphertexts of size $O(\log T)$ and user private keys of size $O(\log T)$. These outperform the previous schemes in terms of both overheads. In particular, they are independent of the parameters in the user dimension, namely n and r; moreover, the first scheme achieves the constant-size ciphertext. These performances of our schemes are comparable to those of the currently best single-user forward-secure public-key encryption scheme (cf. [BBG05]). The public keys for both schemes are of size $O(n + \log T)$. Analogously to [BGW05], we can show that this amount can be traded off to $O(\sqrt{n} + \log T)$ with ciphertext size being increased to $O(\sqrt{n})$ and $O(\sqrt{n} + \log T)$ respectively in both schemes. Security of our systems is based on the Decision Bilinear Diffie-Hellman Exponent assumption (BDHE), which is previously used in [BGW05, BBG05]. We prove the security in the standard model (i.e., without random oracle).

Searchable Broadcast Encryption. Public-key BE can be applied naturally to encrypted file systems, which enable file sharing among privileged users over a public server, as already suggested in [BGW05]. A file can be created by anyone using the public key and the privileged subset can be arbitrarily specified by the creator of the file. Due to a possible large amount of databases, a user Alice might want to retrieve only those files that contain a particular keyword of interest (among all the files in which Alice is specified as a privileged user), but without giving the server the ability to decrypt the databases. *Public-key Broadcast Encryption with Keyword Search* (BEKS) allows to do exactly this.

It enables Alice to give the server a capability (or a trapdoor) to test whether a particular keyword, w, is contained in any (and only) file that includes Alice as a privileged user. This is done in such a way that (1) the server is unable to learn anything else about that file, besides the information about containment of w, and (2) all the other users outside the privileged set cannot learn anything, in particular, cannot generate such a trapdoor, even if they collude. BEKS is a new generalization of public key encryption with keyword search (PEKS) [BDOP04] that we introduce here. We then relate that an anonymous ICBE (1-level HICBE) is sufficient to construct BEKS, analogously to the relation between anonymous IBE and PEKS [ABC+05].

A trivial BEKS achieving ciphertexts of size $O(n)$ can be constructed from the concatenation of PEKS-encryption of the same keyword to each privileged user. Our scheme achieves ciphertexts of size $O(r \log n)$, trapdoors of size $O(\log^3 n)$, and private keys of size $O(\log^4 n)$. Before coming up with this result, we constructively hint that even using the same technique as our FS-BE schemes (where a *non-anonymous* HICBE is sufficient), it might not be easy to construct a BEKS scheme with both ciphertext and private key of sizes independent of n.

5.1.2 Organization of the Chapter.

In Section 5.2, we introduce our HICBE primitive and briefly describe its security notions. In Section 5.3, we show how to construct HICBE schemes based on the BE scheme of Boneh, Gentry, and Waters [BGW05] and the two HIBE constructions of Boneh and Boyen [BB04a] and Boneh, Boyen, and Goh [BBG05]. In Section 5.4.1, we briefly describe the notion of FS-BE schemes, followed by a generic construction from HICBE and a direct construction. We compare our schemes to the other previous FS-BE schemes in Section 5.4.5. In Section 5.5, we introduce the BEKS primitive and its relation to Anonymous ICBE, then give an anonymous HICBE construction based on the recent HIBE scheme of Boyen and Waters [BW06].

5.2 Hierarchical Identity-Coupling Broadcast Encryption

In this section, we define a new primitive called Hierarchical Identity-Coupling Broadcast Encryption (HICBE). It will be used as a building block to construct FS-BE systems, BEKS systems and some generalizations in generic ways.

5.2.1 Syntax of HICBE

HICBE. A HICBE system consists of n users, each with index $i \in \{1, \ldots, n\}$. In usage, a user index will be coupled with some additional arbitrary identity tuple $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_z)$, for any I_j in some predefined identity space and any $z = 1, \ldots, L$ where L is a predetermined maximum depth of tuples. The user i coupling with ID, which we will refer as a *node* (i, ID), will possess its own private key $d_{i,\mathsf{ID}}$. If $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_z)$, then for $j = 1, \ldots, z$, let $\mathsf{ID}_{|j} = (\mathrm{I}_1, \ldots, \mathrm{I}_j)$, and let $\mathsf{ID}_{|0}$ be the empty string ε. A HICBE system enables a derivation from $d_{i,\mathsf{ID}_{|z-1}}$ to $d_{i,\mathsf{ID}}$. In particular, $d_{i,(\mathrm{I}_1)}$ can be derived from d_i, the *root* private keys of i. A HICBE system enables one to encrypt a message to a set of nodes $\{(i, \mathsf{ID}) | i \in S\}$ for arbitrary $S \subseteq \{1, \ldots, n\}$. In this case, we say that it is encrypted to *multi-node* (S, ID). If $i \in S$, the user i coupling with ID (who possesses $d_{i,\mathsf{ID}}$) can decrypt this ciphertext. Formally, a HICBE system is made up of five randomized algorithms as in the following.

For simplicity, we define HICBE as a key encapsulation mechanism (KEM). When $L = 1$, we simply call it a ICBE.

$\mathsf{Setup}(n, L)$**:** Takes as input the number of all users n and the maximum depth L of the identity hierarchy. It outputs a public key pk and a master key msk.

$\mathsf{PrivKeyGen}(i, \mathsf{pk}, \mathsf{msk})$**:** Takes as input a user index i, the public key pk, and the master key msk. It outputs a root private key d_i of user i.

$\mathsf{Derive}(i, \mathsf{ID}, d_{i,\mathsf{ID}_{|z-1}})$**:** Takes as input a user i, an identity ID of depth z, and the private key $d_{i,\mathsf{ID}_{|z-1}}$ of user i coupling with the parent identity $\mathsf{ID}_{|z-1}$. It outputs $d_{i,\mathsf{ID}}$. Here $d_{i,\mathsf{ID}_{|0}} = d_i$.

$\mathsf{Encrypt}(\mathsf{pk}, S, \mathsf{ID})$**:** Takes as input the public key pk, a subset $S \subseteq \{1, \ldots, n\}$, and an identity tuple ID. It outputs a pair (hdr, K) where hdr is called the header and $K \in \mathcal{K}$ is a message encryption key. We will refer to hdr as the broadcast ciphertext.

$\mathsf{Decrypt}(\mathsf{pk}, S, i, d_{i,\mathsf{ID}}, \mathsf{hdr})$**:** Takes as input the pk, a subset S, a user i, the private key $d_{i,\mathsf{ID}}$ of user i coupling with ID, and the header hdr. If $i \in S$ it outputs $K \in \mathcal{K}$ else outputs a special symbol '?'.

The system is correct if for all $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_z) \in \mathcal{I}^{\leq L}$, where $\mathcal{I}$ is the ID space, all $S \subseteq \{1, \ldots, n\}$ and all $i \in S$, if $(\mathsf{pk}, \mathsf{msk}) \xleftarrow{R} \mathsf{Setup}(n, L)$, $d_i \xleftarrow{R} \mathsf{PrivKeyGen}(i, \mathsf{pk}, \mathsf{msk})$, for $j = 1, \ldots, z$, $d_{i,\mathsf{ID}_{|j}} \xleftarrow{R} \mathsf{Derive}(i, \mathsf{ID}_{|j}, d_{i,\mathsf{ID}_{|j-1}})$, and $(\mathsf{hdr}, K) \xleftarrow{R} \mathsf{Encrypt}(\mathsf{pk}, S, \mathsf{ID})$ then $\mathsf{Decrypt}(\mathsf{pk}, S, i, d_{i,\mathsf{ID}}, \mathsf{hdr}) = K$.

5.2.2 Security Notions for HICBE

In this section, we describe the security notions of HICBE omitted from Section 5.2.

Confidentiality. We define semantic security of a HICBE system by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Both $\mathcal{C}$ and $\mathcal{A}$ are given n, L as input.

Setup. The challenger $\mathcal{C}$ runs $\mathsf{Setup}(n, L)$ to obtain a public key pk and the master key msk. It then gives the public key pk to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ adaptively issues queries $q_1, \ldots, q_\mu$ where query q_k is one of the following:

- Private key query $\langle i, \mathsf{ID} \rangle$. $\mathcal{C}$ responds by running algorithm $\mathsf{PrivKeyGen}$ and Derive to derive the private key $d_{i,\mathsf{ID}}$, corresponding to the node (i, ID) and sends $d_{i,\mathsf{ID}}$ to $\mathcal{A}$.
- Decryption query $\langle S, \mathsf{ID}, i, \mathsf{hdr} \rangle$ where $i \in S$. $\mathcal{C}$ responds by running algorithm $\mathsf{PrivKeyGen}$ and Derive to derive the private key $d_{i,\mathsf{ID}}$, corresponding to the node (i, ID). It then gives to $\mathcal{A}$ the output from $\mathsf{Decrypt}(\mathsf{pk}, S, i, d_{i,\mathsf{ID}}, \mathsf{hdr})$.

Challenge. Once $\mathcal{A}$ decides that Phase 1 is over, it outputs $(S^\star, \mathsf{ID}^\star)$ which is the multi-node it wants to attack, where $S^\star \subseteq \{1, \ldots, n\}$. The only restriction is that $\mathcal{A}$ did not previously issue a private key query for $\langle i, \mathsf{ID} \rangle$ such that $i \in S^\star$ and that either $\mathsf{ID} = \mathsf{ID}^\star$ or ID is a prefix of $\mathsf{ID}^\star$. $\mathcal{C}$ then compute $(\mathsf{hdr}^\star, K) \xleftarrow{R} \mathsf{Encrypt}(\mathsf{pk}, S^\star, \mathsf{ID}^\star)$ where $K \in \mathcal{K}$. Next $\mathcal{C}$ picks a random $b \in \{0, 1\}$. It sets $K_b = K$ and picks a random K_{1-b} in $\mathcal{K}$. It then gives $(\mathsf{hdr}^\star, K_0, K_1)$ to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ issues additional queries $q_{\mu+1}, \ldots, q_\nu$ where query q_k is one of the following:

- Private key query $\langle i, \mathsf{ID}\rangle$ such that if $i \in S^\star$ then $\mathsf{ID} \neq \mathsf{ID}^\star$ and ID must not be a prefix of $\mathsf{ID}^\star$, else if $i \notin S^\star$ then ID can be arbitrary.

- Decryption query $\langle S, \mathsf{ID}, i, \mathsf{hdr}\rangle$ where $i \in S$ and $S \subseteq S^\star$.[2] The only constraint is that $\mathsf{hdr} \neq \mathsf{hdr}^\star$ if either $\mathsf{ID} = \mathsf{ID}^\star$ or ID is a prefix of $\mathsf{ID}^\star$.

In both cases, $\mathcal{C}$ responds as in Phase 1. These queries may be adaptive.

Guess Finally $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$ for b and wins the game if $b = b'$.

We refer to such an adversary $\mathcal{A}$ as an IND-aID-aSet-CCA adversary and the above game as the IND-aID-aSet-CCA game.

Weaker notions of security can be defined by modifying the above game so that it is required that the adversary must commit ahead of time to the target subset $S^\star$ or the target identity $\mathsf{ID}^\star$ or both. These notions are analogous to the notion of selective-identity secure HIBE, given by Canetti, Halevi, and Katz [CHK03, CHK04]. We have 4 possible combinations: the game IND-xID-ySet-CCA where $(\mathsf{x},\mathsf{y}) \in \{(\mathsf{a},\mathsf{a}),(\mathsf{a},\mathsf{s}),(\mathsf{s},\mathsf{a}),(\mathsf{s},\mathsf{s})\}$. If $(\mathsf{x},\mathsf{y}) = (\mathsf{s},*)$ then it is exactly the same as IND-aID-aSet-CCA except that $\mathcal{A}$ must disclose to $\mathcal{C}$ the target identity $\mathsf{ID}^\star$ before the Setup phase. Analogously, if $(\mathsf{x},\mathsf{y}) = (*,\mathsf{s})$, $\mathcal{A}$ must disclose the target subset $S^\star$ before the Setup phase. For only the case of (s,s), it is further required that the restrictions on private key queries from phase 2 also hold in phase 1. Intuitively, s means selective security and a means adaptive security.

We define the advantage of the adversary $\mathcal{A}$ in attacking the HICBE scheme $\mathcal{E}$ in the game IND-xID-ySet-CCA as $\mathsf{AdvHICBE}_{\mathsf{xy}}(\mathcal{E},\mathcal{A}) = |\Pr[b = b'] - \frac{1}{2}|$, where the probability is over the random bits used by $\mathcal{C}$ and $\mathcal{A}$ in that game.

Definition 5.1. We say that a HICBE system $\mathcal{E}$ is $(t, q_\mathsf{P}, q_\mathsf{D}, \epsilon)$-IND-xID-ySet-CCA-secure if for any t-time IND-xID-ySet-CCA adversary $\mathcal{A}$ that makes at most q_P chosen private key queries and at most q_D chosen decryption queries, we have that $\mathsf{AdvHICBE}_{\mathsf{xy}}(\mathcal{E},\mathcal{A}) < \epsilon$.

Definition 5.2. We say that a HICBE system $\mathcal{E}$ is $(t, q_\mathsf{P}, \epsilon)$-IND-xID-ySet-CPA-secure if $\mathcal{E}$ is $(t, q_\mathsf{P}, 0, \epsilon)$-IND-xID-ySet-CCA-secure.

Proposition 5.3. *A HICBE system for n users which is $(t, q_\mathsf{P}, q_\mathsf{D}, \epsilon)$-IND-xID-sSet-CCA-secure is also $(t, q_\mathsf{P}, q_\mathsf{D}, \epsilon/2^n)$-IND-xID-aSet-CCA-secure. This also holds for the case of CPA.*

Anonymity. Recipient anonymity is the property that the adversary be unable to distinguish the ciphertext intended for a chosen identity from another one intended for a random identity. We capture via what we call ANO-xID-ySet-CCA[Δ] game where $\Delta \subseteq \{0,\ldots,L\}$ indicates a set of levels that satisfy anonymity, with 0 corresponds to the anonymity of the set S. Such a notion is generalized from [ABC+05]. This is defined exactly the same as IND-xID-ySet-CCA except the challenge phase: instead of creating challenge ciphertext as usual, $\mathcal{C}$ picks a random bit $b \in \{0,1\}$ and computes $(\mathsf{hdr}^\star, K) \xleftarrow{R} \mathsf{Encrypt}(\mathsf{pk}, S_b^\star, \mathsf{ID}_b^\star)$ where $(S_0^\star, \mathsf{ID}_0^\star) = (S^\star, \mathsf{ID}^\star)$ (the requested set and identity from $\mathcal{A}$) and $S_1^\star$ is a random set in $2^{\{1,\ldots,n\}}$ if $0 \in \Delta$ otherwise $S_1^\star = S^\star$ and $\mathsf{ID}_1^\star$ is an identity tuple of the same length as $\mathsf{ID}^\star$, denoted z, that has a random identity at each level in $\{1,\ldots,z\} \cap \Delta$.

[2]Here, it is without loss of generality, and simpler, that we just restrict $S \subseteq S^\star$ since for S such that $S \not\subseteq S^\star$, one can make a private key query for some $i \in S \setminus S^\star$ and perform the decryption oneself.

5.2.3 Conversion for Chosen-Ciphertext Security

A result of Canetti, Halevi, and Katz [CHK04] gives an efficient way to construct a selective-identity, chosen-ciphertext secure L-level HIBE from a selective-identity, chosen-plaintext secure $L+1$-level HIBE. We can generalize this technique for applying to HICBE in a straightforward way. We also note that a more efficient method by Boneh and Katz [BK05] can also be generalized to the case of HICBE straightforwardly.

Given a HICBE scheme $\mathsf{HICBE} = (\mathsf{Setup}, \mathsf{PrivKeyGen}, \mathsf{Derive}, \mathsf{Encrypt}, \mathsf{Decrypt})$ for $(L+1)$-level of identities which is IND-sID-ySet-CPA-secure, we construct another HICBE scheme $\mathsf{HICBE}^{\diamond} = (\mathsf{Setup}^{\diamond}, \mathsf{PrivKeyGen}^{\diamond}, \mathsf{Derive}^{\diamond}, \mathsf{Encrypt}^{\diamond}, \mathsf{Decrypt}^{\diamond})$ for L-level of identities secure in the IND-sID-ySet-CCA sense. In the construction, we use a signature scheme $\mathsf{Sig} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ in which the verification key output by Gen is in $\mathcal{I}$, the identity space of the HICBE scheme. We require that this scheme be secure in the sense of strong unforgeability. The construction of $\mathsf{HICBE}^{\diamond}$ proceeds as follows: The first three algorithms of $\mathsf{HICBE}^{\diamond}$ are exactly the same as those of HICBE. The rest two are as follow.

$\mathsf{Encrypt}^{\diamond}(\mathsf{pk}, S, \mathsf{ID})$**:** The algorithm first runs Gen to obtain verification key V_{SIG} and signing key K_{SIG}. It computes $(\mathsf{hdr}, K) \leftarrow \mathsf{Encrypt}(\mathsf{pk}, S, \mathsf{ID}||V_{\mathsf{SIG}})$. Let $\mathsf{hdr}^{\diamond} = (\mathsf{hdr}, \mathsf{Sign}_{K_{\mathsf{SIG}}}(\mathsf{hdr}), V_{\mathsf{SIG}})$. It then outputs $(\mathsf{hdr}^{\diamond}, K)$.

$\mathsf{Decrypt}^{\diamond}(\mathsf{pk}, S, i, d_{i,\mathsf{ID}}, \mathsf{hdr}^{\diamond})$**:** It parses $\mathsf{hdr}^{\diamond} = (\mathsf{hdr}, \sigma, V_{\mathsf{SIG}})$ then checks whether $\mathsf{Vrfy}_{V_{\mathsf{SIG}}}(\mathsf{hdr}, \sigma) = 1$. If not, it outputs $\perp$. Otherwise, it outputs the result from $\mathsf{Decrypt}(\mathsf{pk}, S, i, d_{i,\mathsf{ID}}, \mathsf{hdr})$.

Before stating the theorem, we recall that Sig is (t, ϵ, q_S) strongly existentially unforgeable if no t-time adversary who makes at most q_S signature queries can produce some new (message,signature) pair with probability at least ϵ.

Theorem 5.4. *Suppose the HICBE scheme for n users and maximum depth $L+1$ is $(t, q_{\mathsf{P}}, \epsilon_1)$-IND-sID-ySet-CPA-secure for some $\mathsf{y} \in \{\mathsf{s}, \mathsf{a}\}$ and the signature scheme is $(t, \epsilon_2, 1)$ strongly existentially unforgeable. Then HICBE$^{\diamond}$ for n users and maximum depth L is $(t, q_{\mathsf{P}}, q_{\mathsf{D}}, \epsilon_1 + \epsilon_2)$-IND-sID-ySet-CCA-secure for arbitrary n, L, t, q_{P}, and q_{D}.*

5.3 HICBE Constructions

In this section, we will give our first two HICBE constructions. We begin by offering some intuition into the design. A HICBE system must have both broadcast and hierarchical-identity-based derivation properties. To achieve this we will combine some techniques from the BGW broadcast encryption [BGW05] with the BB and BBG HIBE systems by Boneh-Boyen [BB04a] and Boneh-Boyen-Goh [BBG05] respectively. We will first describe that a HICBE system, however, cannot simply be obtained from combining a broadcast encryption and a HIBE scheme in a naive manner.

An Efficient but Insecure Approach. Suppose we created BE and HIBE schemes separately and let the key for node (i, ID) be composed of the private key for user i from the BE scheme and the key for ID from the HIBE scheme. To encrypt to (S, ID), just encrypt using any CCA-secure AND-double encryption [DK05] of BE-ciphertext for S and HIBE-ciphertext for ID. In order to decrypt, a user $i \in S$ coupling with ID will need to be able to decrypt under both systems. However, two malicious users, Alice who possesses key for node $(j \notin S, \mathsf{ID})$ and Bob who possesses key for node $(k \in S, \mathsf{ID}')$ where ID is a prefix of,

but not equal to ID', can collude by using Alice's HIBE-key and Bob's BE-key to decrypt the ciphertext.

Secure but Inefficient Approaches. We create a HIBE and let the first level to be used as user indexes and lower levels be used as identity tuples. To encrypt to (S, ID), just use OR-multiple encryption to each identity $i||\mathsf{ID}$ of the HIBE for all $i \in S$. This yields ciphertext of size linear in $|S|$, which is quite inefficient. One may use a wild-card HIBE [ACD+06], which allows to encrypt to many identities (in the same level) in one ciphertext. But this allows only to encrypt to *all* users. A more sophisticated approach is to "expand" this flat tree of users to the tree of key derivation patterns of NNL broadcast encryption [NNL01, DF02] so as to trade off ciphertext size with private key size. This "double hierarchies" (one for user-broadcast and one for identity) approach was already appeared in [YFDL04, BBG05] in the context of forward-secure HIBE and BE. However, the best ciphertext size we can achieve from NNL-like schemes is still only being $O(r)$.

Indeed, these secure schemes share a common principle behind so as to resist the collusion attack occurred in the first naive method. Such a principle is to construct private keys in such a way that they must be "simultaneously" used for both broadcast and hierarchical identity portions of a HICBE system. In particular, the secure naive approach above embeds both portions into one big HIBE, while the YFDL-like approaches make the two HIBE systems essentially "intertwined". Note that the two HIBE systems used in [YFDL04] are two copies of Gentry-Silverberg HIBE [GS02], while in [BBG05], the BB and BBG systems are used.

Our Design Intuition. In this chapter, we are able to make secure integrations between the BGW broadcast encryption and the BB or BBG HIBE systems. We achieve this by observing first that in the BB or BBG HIBE systems, each private key consists of a main key and some cancelation keys. The main key is the product of a "core" part and a "perturbation" part. The main point in our design is to replace the core part of the HIBE-key with the private key of the BGW scheme. This will force the key to be simultaneously used in both the BE and HIBE systems, thus we can hope that the two systems will become intertwined! Intuitively speaking, due to the implicit "orthogonality" of the BGW scheme and the BB or BBG scheme, other portions of the two sub-systems will not interfere functionality and security of each other. This reason, together with the similarity among their underlying assumptions, allows us to prove the security. There are some points to make clear on the difference of our integration technique to the previous ones. The integrations of two HIBE systems, GS and GS in [YFDL04] and BB and BBG in [BBG05], were able to be done due to the fact that (1) both sub-systems are the same primitive (HIBE), and (2) the core parts of keys in both sub-systems are algebraically exactly the same; therefore, intuitively, collecting perturbation parts (multiplicatively) from both HIBE sub-systems to the main key will simply force the intertwining. In contrast, our integration takes place over different primitives (BE and HIBE) and the core part of the key in the BB or BBG system is different from that of the BGW scheme, which itself also essentially provides different one-element key for each user. Therefore, there could be a possibility that the two systems were not compatible in the first place.

As a result, however, after proving all the functionality and security issues (which we will do from now), it turns out that our intuition was correct. In particular, it means that we are able to show that the BGW scheme and the BB or BBG scheme can be combined with

the minimal modification as possible. Rather than a drawback (for lacking new elements), we view this contribution as an advantage; indeed, the resulting schemes are simple and inherit those good efficiency performances from the original sub-systems directly.

5.3.1 Our First HICBE Construction Based on BGW and BB

We first show how to combine the basic BGW scheme with the HIBE scheme by Boneh and Boyen [BB04a]. We assume that the identity space $\mathcal{I}$ is $\mathbb{Z}_p$. Thus, if ID is of depth z then $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_z) \in {\mathbb{Z}_p}^z$. As in [BB04a], we can later extend the construction to arbitrary in $\{0,1\}^*$ by first hashing each I_j using a collision resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p$. We follow almost the same terminology from [BGW05, BB04a]. This scheme, denoted BasicHICBE1, works as follows.

Construction 5-1. Our First HICBE: BasicHICBE1

Setup(n, L)**:** Let $\mathbb{G}$ be a bilinear group of prime order p. The algorithm first picks a random generator $g \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p$. It computes $g_i = g^{(\alpha^i)} \in \mathbb{G}$ for $i = 1, 2, \ldots, n, n+2, \ldots, 2n$. Next, it picks a random $\gamma \in \mathbb{Z}_p$ and sets $v = g^\gamma \in \mathbb{G}$. It then picks random elements $h_1, \ldots, h_L \in \mathbb{G}$. The public key is:

$$\mathsf{pk} = (g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, v, h_1, \ldots, h_L) \in \mathbb{G}^{2n+L+1}.$$

The master key is $\mathsf{msk} = \gamma$. For $j = 1, \ldots, L$, we define $F_j : \mathbb{Z}_p \to \mathbb{G}$ to be the function: $F_j(x) = g_1^x h_j$. The algorithm outputs the public key pk and the master key msk.

PrivKeyGen$(i, \mathsf{pk}, \mathsf{msk})$**:** Set a root private key for user i as $d_i = (g_i)^\gamma \in \mathbb{G}$. Indeed $d_i = v^{(\alpha^i)}$.

Derive$(i, \mathsf{ID}, d_{i,\mathsf{ID}_{|z-1}})$**:** To generate the private key for node (i, ID) where $i \in \{1, \ldots, n\}$ and $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_z) \in {\mathbb{Z}_p}^z$ of depth $z \leq L$, pick random elements $s_1, \ldots, s_z \in \mathbb{Z}_p$ and output

$$d_{i,\mathsf{ID}} = \left((g_i)^\gamma \cdot \prod_{j=1}^{z} F_j(\mathrm{I}_j)^{s_j} \ , \ g^{s_1} \ , \ \ldots \ , \ g^{s_z} \right) \in \mathbb{G}^{z+1}.$$

Note that the private key for node (i, ID) can be generated just given a private key for node $(i, \mathsf{ID}_{|z-1})$ where $\mathsf{ID}_{|z-1} = (\mathrm{I}_1, \ldots \mathrm{I}_{z-1}) \in {\mathbb{Z}_p}^{z-1}$, as required. Indeed, let $d_{i,\mathsf{ID}_{|z-1}} = (a_0, \ldots, a_{z-1})$ be the private key for node $(i, \mathsf{ID}_{|z-1})$. To generate $d_{i,\mathsf{ID}}$, pick a random $s_z \in \mathbb{Z}_p$ and output $d_{i,\mathsf{ID}} = (a_0 \cdot F_z(\mathrm{I}_z)^{s_z}, a_1, \ldots, a_{z-1}, g^{s_z})$.

Encrypt$(\mathsf{pk}, S, \mathsf{ID})$**:** Pick a random $t \in \mathbb{Z}_p$ and set $K = e(g_{n+1}, g)^t$. The value $e(g_{n+1}, g)$ can be computed as $e(g_n, g_1)$. Let $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_z)$. It outputs (hdr, K) by setting

$$\mathsf{hdr} = \left(g^t, \ (v \cdot \prod_{j \in S} g_{n+1-j})^t \ , \ F_1(\mathrm{I}_1)^t \ , \ \ldots \ , \ F_z(\mathrm{I}_z)^t \right) \in \mathbb{G}^{z+2}. \tag{5.1}$$

Decrypt$(\mathsf{pk}, S, i, d_{i,\mathsf{ID}}, \mathsf{hdr})$**:** Parse the header as $\mathsf{hdr} = (C_0, C_1, A_1, \ldots, A_z) \in \mathbb{G}^{z+2}$. Also

parse $d_{i,\mathsf{ID}} = (a_0, \ldots, a_z) \in \mathbb{G}^{z+1}$. Then output

$$K = e(g_i, C_1) \cdot \prod_{j=1}^{z} e(A_j, a_j) \;/\; e(a_0 \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i},\ C_0).$$

We verify the correctness of BasicHICBE1 as follows. We use the fact that $g_i^{\alpha^j} = g_{i+j}$ for any i, j. User $i \in S$ decrypts the ciphertext as:

$$\begin{aligned}
K &= e(g_i, C_1) \cdot \prod_{j=1}^{z} e(A_j, a_j) \;/\; e(a_0 \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i},\ C_0) \\
&= e(g^{(\alpha^i)}, (v \cdot \prod_{j \in S} g_{n+1-j})^t) \cdot \prod_{j=1}^{z} e(F_j(\mathrm{I}_j)^t, g^{s_j}) \;/\; e(v^{(\alpha^i)} \cdot \prod_{j=1}^{z} F_j(\mathrm{I}_j)^{s_j} \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i},\ g^t) \\
&= e(g^{(\alpha^i)}, (v \cdot \prod_{j \in S} g_{n+1-j})^t) \cdot \prod_{j=1}^{z} e(F_j(\mathrm{I}_j)^t, g^{s_j}) \;/\; e(v^{(\alpha^i)} \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i},\ g^t) \cdot e(\prod_{j=1}^{z} F_j(\mathrm{I}_j)^{s_j}, g^t) \\
&= e(g^{(\alpha^i)}, (g_{n+1-i})^t) \cdot e(g^{(\alpha^i)}, (v \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j})^t) \;/\; e(v^{(\alpha^i)} \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i},\ g^t) \\
&= e(g_{n+1}, g)^t \cdot e(g, (v^{(\alpha^i)} \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i})^t) \;/\; e(v^{(\alpha^i)} \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i},\ g^t) \quad = \quad e(g_{n+1}, g)^t,
\end{aligned}$$

as required.

The scheme inherits a good property of the BGW scheme: the ciphertext size and user private key size are independent of n. Indeed, when we let $\mathsf{ID} = \varepsilon$, the corresponding algorithms become those of the basic BGW scheme. Note that the ciphertext size and the private key size is the same as in the BB scheme, which is $z + 1$ elements in $\mathbb{G}$, where z is the depth of ID. The public key contains $2n + L + 1$ elements of $\mathbb{G}$. The derivation time will be dominated by $O(z)$ exponentiations. The encryption time will be dominated by n multiplications or $O(z)$ exponentiations. The decryption time will be dominated by n multiplications or z pairings. Finally, to connect the whole intuitive picture that was given in the design guideline, we note that in the private key, the term $(g_i)^\gamma$ is the "core" part, $\prod_{j=1}^{z} F_j(\mathrm{I}_j)^{s_j}$ is the "perturbation" part, and those g^{s_j} terms compose the cancelation part.

The security proof, although vaguely resembles those of BGW and BB, is not straightforward as we have to simulate both sub-systems *simultaneously*.

Theorem 5.5. *Let $\mathbb{G}$ be a bilinear group of prime order p. Suppose the Decision (t, ϵ, n)-BDHE assumption holds in $\mathbb{G}$. Then the BasicHICBE1 system for n users and maximum depth L is $(t', q_\mathsf{P}, \epsilon)$-IND-sID-sSet-CPA-secure for arbitrary n, L and q_P, and any $t' < t - \Theta(\tau_{\mathsf{exp}} L q_\mathsf{P})$ where τ_{exp} is the maximum time for an exponentiation in $\mathbb{G}$.*

Proof. Suppose there exists an adversary, $\mathcal{A}$, that has advantage ϵ in attacking the HICBE scheme. We build an algorithm $\mathcal{B}$ that solves the Decision n-BDHE problem in $\mathbb{G}$. Algorithm $\mathcal{B}$ is given as input a random n-BDHE challenge $(g, h, \vec{y}_{g,\alpha,n}, Z)$, where $\vec{y}_{g,\alpha,n} =$

$(g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$ and Z is either $e(g_{n+1}, h)$ or a random element in $\mathbb{G}_1$ (recall that $g_j = g^{(\alpha^j)}$). Algorithm $\mathcal{B}$ proceeds as follows.

Initialization. The selective (identity, subset) game begins with $\mathcal{A}$ first outputting a multi-node $(S^\star, \mathsf{ID}^\star)$ where $S^\star \subseteq \{1, \ldots, n\}$ and $\mathsf{ID}^\star = (\mathrm{I}_1^\star, \ldots, \mathrm{I}_z^\star) \in {\mathbb{Z}_p}^z$ of depth $z \leq L$ that it intends to attack. If necessary, $\mathcal{B}$ appends random elements in $\mathbb{Z}_p$ to $\mathsf{ID}^\star$ so that $\mathsf{ID}^\star$ is a vector of length L.

Setup. To generate the public key pk, algotrithm $\mathcal{B}$ chooses a random $u \in \mathbb{Z}_p$ and sets $v = g^u(\prod_{j \in S^\star} g_{n+1-j})^{-1}$. $\mathcal{B}$ then picks $\beta_1, \ldots, \beta_L \in \mathbb{Z}_p$ at random and define $h_j = {g_1}^{-\mathrm{I}_j^\star} g^{\beta_j} \in \mathbb{G}$ for $j = 1, \ldots, L$. It gives $\mathcal{A}$ the public key

$$\mathsf{pk} = (g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, v, h_1, \ldots, h_L) \in \mathbb{G}^{2n+L+1}.$$

Note that since g, α, u and the β_j values are chosen uniformly at random, the public key has an identical distribution to that in the actual construction. As before, for $j = 1, \ldots, L$ we define $F_j : \mathbb{Z}_p \to \mathbb{G}$ to be the function $F_j(x) = g_1^x h_j = {g_1}^{x - \mathrm{I}_j^\star} g^{\beta_j}$.

Phase 1. $\mathcal{A}$ issues up to q_P private key queries. Consider a query for the private key corresponding to node (i, ID), of which $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_w) \in {\mathbb{Z}_p}^w$ where $w \leq L$. We divide to two cases upon whether i is in $S^\star$ or not.

- If $i \notin S^\star$ then $\mathcal{B}$ responds to the query by first computing a root private key d_i from which it can then construct a private key $d_{i,\mathsf{ID}}$ for the request node (i, ID). In this case, $\mathcal{B}$ computes d_i as

$$d_i = g_i^u \cdot \Big(\prod_{j \in S^\star} g_{n+1-j+i}\Big)^{-1}.$$

  Indeed we have that $d_i = (g^u(\prod_{j \in S^\star} g_{n+1-j})^{-1})^{(\alpha^i)} = v^{(\alpha^i)}$ as required.

- If $i \in S^\star$ then from the restriction of the private key query, it must be that ID is neither $\mathsf{ID}^\star$ nor any prefix of $\mathsf{ID}^\star$. Let k be the smallest index such that $\mathrm{I}_k \neq \mathrm{I}_k^\star$. Necessarily $1 \leq k \leq w$. $\mathcal{B}$ responds to the query by first computing a private key for node $(i, \mathsf{ID}_{|k})$ from which it then construct a private key for the request node (i, ID). $\mathcal{B}$ picks random elements $s_1, \ldots, s_k \in \mathbb{Z}_p$ and sets

$$a_0 = g_i^u \Big(\prod_{\substack{j \in S^\star \\ j \neq i}} g_{n+1-j+i} \Big)^{-1} \cdot g_n^{\frac{\beta_k}{\mathrm{I}_k - \mathrm{I}_k^\star}} \cdot \left(\prod_{r=1}^{k} F_r(\mathrm{I}_r)^{s_r} \right), \tag{5.2}$$

$$a_1 = g^{s_1} \ , \ \ldots \ , \ a_{k-1} = g^{s_{k-1}} \ , \ a_k = g_n^{\frac{1}{\mathrm{I}_k - \mathrm{I}_k^\star}} g^{s_k}. \tag{5.3}$$

  We claim that $(a_0, a_1, \ldots, a_k)$ is a valid random private key for node $(i, \mathsf{ID}_{|k})$. To see this, let $\tilde{s}_k = s_k + \alpha^n/(\mathrm{I}_k - \mathrm{I}_k^\star)$. Then we have that

$$g_n^{\frac{\beta_k}{\mathrm{I}_k - \mathrm{I}_k^\star}} \cdot F_k(\mathrm{I}_k)^{s_k} = g_n^{\frac{\beta_k}{\mathrm{I}_k - \mathrm{I}_k^\star}} \cdot (g_1^{\mathrm{I}_k - \mathrm{I}_k^\star} g^{\beta_k})^{s_k} = g_{n+1}^{-1} (g_1^{\mathrm{I}_k - \mathrm{I}_k^\star} g^{\beta_k})^{s_k + \frac{\alpha^n}{\mathrm{I}_k - \mathrm{I}_k^\star}} = g_{n+1}^{-1} F_k(\mathrm{I}_k)^{\tilde{s}_k}. \tag{5.4}$$

Consequently, the private key $(a_0, a_1, \ldots, a_k)$ defined above satisfies

$$a_0 = g_i^u \Big(\prod_{j \in S^\star} g_{n+1-j+i} \Big)^{-1} \cdot \Big(\prod_{r=1}^{k-1} F_r(\mathrm{I}_r)^{s_r} \Big) F_k(\mathrm{I}_k)^{\tilde{s}_k} = v^{(\alpha^i)} \cdot \Big(\prod_{r=1}^{k-1} F_r(\mathrm{I}_r)^{s_r} \Big) F_k(\mathrm{I}_k)^{\tilde{s}_k},$$
$$a_1 = g^{s_1} \ , \ \ldots \ , \ a_{k-1} = g^{s_{k-1}} \ , \ a_k = g^{\tilde{s}_k},$$

where $s_1, \ldots, s_{k-1}, \tilde{s}_k$ are uniform in $\mathbb{Z}_p$. This matched the definition for a private key for node $(i, \mathsf{ID}_{|k})$. Therefore, $(a_0, a_1, \ldots, a_k)$ is a valid private key for node $(i, \mathsf{ID}_{|k})$. $\mathcal{B}$ derives a private key for the requested node (i, ID) from $(a_0, a_1, \ldots, a_k)$ and gives $\mathcal{A}$ the result.

Challenge. To generate the challenge, $\mathcal{B}$ computes $\mathsf{hdr}^\star$ as $(h, h^u, h^{\beta_1}, \ldots, h^{\beta_z})$. It then randomly chooses a bit $b \in \{0, 1\}$ and sets $K_b = Z$ and picks a random K_{1-b} in $\mathbb{G}_1$. $\mathcal{B}$ then gives $(\mathsf{hdr}^\star, K_0, K_1)$ to $\mathcal{A}$.

We claim that when $Z = e(g_{n+1}, h)$ (that is, the input to $\mathcal{B}$ is a n-BDHE tuple) then $(\mathsf{hdr}^\star, K_0, K_1)$ is a valid challenge to $\mathcal{A}$ as in a real attack game. To see this, write $h = g^t$ for some (unknown) $t \in \mathbb{Z}_p$. Then, we have that

$$h^u = (g^u)^t = (g^u (\prod_{j \in S^\star} g_{n+1-j})^{-1} (\prod_{j \in S^\star} g_{n+1-j}))^t = (v \prod_{j \in S^\star} g_{n+1-j})^t, \tag{5.5}$$

and that $h^{\beta_j} = (g^{\beta_j})^t = F_j(\mathrm{I}_j^\star)^t$, for $j = 1, \ldots, z$. Thus, by definition, $(h, h^u, h^{\beta_1}, \ldots, h^{\beta_z})$ is a valid encryption of the key $e(g_{n+1}, g)^t$. Furthermore, $e(g_{n+1}, g)^t = e(g_{n+1}, h) = Z = K_b$ and hence $(\mathsf{hdr}^\star, K_0, K_1)$ is a valid challenge to $\mathcal{A}$.

On the other hand, when Z is random in $\mathbb{G}_1$ (that is, the input to $\mathcal{B}$ is a random tuple) then K_0, K_1 are just random independent elements of $\mathbb{G}_1$.

Phase 2. $\mathcal{A}$ continues to ask queries not issued in Phase 1. $\mathcal{B}$ responds as before.

Guess. Finally, $\mathcal{A}$ outputs $b' \in \{0, 1\}$ for guessing b. If $b = b'$ then $\mathcal{B}$ outputs 1 (meaning $Z = e(g_{n+1}, h)$). Otherwise, it outputs 0 (meaning Z is random in $\mathbb{G}_1$).

We see that if $(g, h, \vec{y}_{g,\alpha,n}, Z)$ is sampled from $\mathcal{R}_{BDHE}$ then $\Pr[\mathcal{B}(g, h, \vec{y}_{g,\alpha,n}, Z) = 0] = \frac{1}{2}$. On the other hand, if $(g, h, \vec{y}_{g,\alpha,n}, Z)$ is sampled from $\mathcal{P}_{BDHE}$ then $|\Pr[\mathcal{B}(g, h, \vec{y}_{g,\alpha,n}, Z) = 0] - \frac{1}{2}| \geq \epsilon$. It follows that $\mathcal{B}$ has advantage at least ϵ in solving n-BDHE problem in $\mathbb{G}$. This concludes the proof of Theorem 5.5. □

5.3.2 Our Second HICBE Construction Based on BGW and BBG

Our method of integrating the BGW system can also be applied to the BBG HIBE scheme analogously to the previous integration. In contrast, this time we achieve a feature of "reusing" the public key from the BGW portion to be used for the BBG portion. Consequently, the resulting scheme has exactly the same public key as the BGW scheme except for only one additional element of $\mathbb{G}$.

We will assume that $L \leq n$, otherwise just create dummy users so as to be so; a more efficient way will be discussed in the next subsection. As usual we can assume that $\mathcal{I}$ is $\mathbb{Z}_p$. The scheme, denoted by BasicHICBE2, works as follows.

Construction 5-2. Our Second HICBE: BasicHICBE2

$\mathsf{Setup}(n, L)$**:** The algorithm first picks a random generator $g \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p$. It computes $g_i = g^{(\alpha^i)} \in \mathbb{G}$ for $i = 1, 2, \ldots, n, n+2, \ldots, 2n$. Next, it randomly picks $y \in \mathbb{G}$, $\gamma \in \mathbb{Z}_p$ and sets $v = g^\gamma \in \mathbb{G}$. The public key is:

$$\mathsf{pk} = (g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, v, y) \in \mathbb{G}^{2n+2}.$$

The master key is $\mathsf{msk} = \gamma$. It outputs $(\mathsf{pk}, \mathsf{msk})$. For conceptual purpose, let $h_j = g_{n+1-j}$ for $j = 1, \ldots, L$; intuitively, the h_j terms will be used to visually indicate the BBG portion, while the g_j terms are for the BGW portion.

$\mathsf{PrivKeyGen}(i, \mathsf{pk}, \mathsf{msk})$**:** Set a root private key for i as $d_i = (g_i)^\gamma = v^{(\alpha^i)} \in \mathbb{G}$.

$\mathsf{Derive}(\mathsf{pk}, i, \mathsf{ID}, d_{i,\mathsf{ID}_{|z-1}})$**:** To generate the private key for node (i, ID) where $i \in \{1, \ldots, n\}$ and $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_z) \in {\mathbb{Z}_p}^z$ of depth $z \leq L$, pick a random element $s \in \mathbb{Z}_p$ and output

$$d_{i,\mathsf{ID}} = \left((g_i)^\gamma \cdot (h_1^{\mathrm{I}_1} \cdots h_z^{\mathrm{I}_z} \cdot y)^s \ , \ g^s \ , \ h_{z+1}^s \ , \ \ldots \ , \ h_L^s \right) \in \mathbb{G}^{2+L-z}.$$

Note that the private key for node (i, ID) can be generated just given a private key for node $(i, \mathsf{ID}_{|z-1})$ where $\mathsf{ID}_{|z-1} = (\mathrm{I}_1, \ldots \mathrm{I}_{z-1}) \in {\mathbb{Z}_p}^{z-1}$, as required. Indeed, let $d_{i,\mathsf{ID}_{|z-1}} = (a_0, a_1, b_z, \ldots, b_L)$ be the private key for node $(i, \mathsf{ID}_{|z-1})$. To generate $d_{i,\mathsf{ID}}$, pick a random $\delta \in \mathbb{Z}_p$ and output $d_{i,\mathsf{ID}} = \big(a_0 \cdot b_z^{\mathrm{I}_z} \cdot (h_1^{\mathrm{I}_1} \cdots h_z^{\mathrm{I}_z} \cdot y)^\delta \ , \ a_1 \cdot g^\delta \ , \ b_{z+1} \cdot h_{z+1}^\delta \ \ldots \ , \ b_L \cdot h_L^\delta \big)$. This key has a proper distribution as a private key for node (i, ID) with the randomness $s = s' + \delta \in \mathbb{Z}_p$, where s' is the randomness in $d_{i,\mathsf{ID}_{|z-1}}$. Note that the private key $d_{i,\mathsf{ID}}$ becomes shorter as the depth of ID increases.

$\mathsf{Encrypt}(\mathsf{pk}, S, \mathsf{ID})$**:** Pick a random $t \in \mathbb{Z}_p$ and set $K = e(g_{n+1}, g)^t$. The value $e(g_{n+1}, g)$ can be computed as $e(g_n, g_1)$. Let $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_z)$. It outputs (hdr, K) where we let

$$\mathsf{hdr} = \left(g^t, \ (v \cdot \prod_{j \in S} g_{n+1-j})^t \ , \ (h_1^{\mathrm{I}_1} \cdots h_z^{\mathrm{I}_z} \cdot y)^t \right) \in \mathbb{G}^3.$$

$\mathsf{Decrypt}(\mathsf{pk}, S, i, d_{i,\mathsf{ID}}, \mathsf{hdr})$**:** Let $\mathsf{hdr} = (C_0, C_1, C_2) \in \mathbb{G}^3$ and let $d_{i,\mathsf{ID}} = (a_0, a_1, b_{z+1}, \ldots, b_L) \in \mathbb{G}^{2+L-z}$. Then output

$$K = e(g_i, C_1) \cdot e(C_2, a_1) \ / \ e(a_0 \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i}, \ C_0).$$

The scheme inherits good properties from both the BGW scheme: the ciphertext size and user private key size are independent of n, and the BBG scheme: the ciphertext size is constant. One difference from the BBG system is that we let the h_j terms be of special forms, namely $h_j = g_{n+1-j}$, instead of random elements in $\mathbb{G}$ as in [BBG05]. This allows us to save the public key size since those g_j terms are already used for the BGW system. Indeed, suppose that the BGW BE system has been already established, it can be augmented to a HICBE version by just once publishing one random element, namely $y \in \mathbb{G}$, as an additional

public key. Note that defining h_j terms in this way is also crucial to the security proof. We prove the security under the Decision n-BDHE assumption. This strong assumption is already necessary for both the (stand-alone) BGW and BBG systems.[3]

Theorem 5.6. *Let $\mathbb{G}$ be a bilinear group of prime order p. Suppose the Decision (t, ϵ, n)-BDHE assumption holds in $\mathbb{G}$. Then the BasicHICBE2 scheme for n users and maximum depth L is $(t', q_{\mathsf{P}}, \epsilon)$-IND-sID-sSet-CPA-secure for arbitrary n, L such that $L \leq n$ and q_{P}, and any $t' < t - \Theta(\tau_{\mathsf{exp}} L q_{\mathsf{P}})$ where τ_{exp} is the maximum time for an exponentiation in $\mathbb{G}$.*

Proof. Suppose there exists an adversary, $\mathcal{A}$, that has advantage ϵ in attacking the HICBE scheme. We build an algorithm $\mathcal{B}$ that solves the Decision n-BDHE problem in $\mathbb{G}$. $\mathcal{B}$ is given as input a random n-BDHE challenge $(g, h, \vec{y}_{g,\alpha,n}, Z)$, where $\vec{y}_{g,\alpha,n} = (g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$ and Z is either $e(g_{n+1}, h)$ or a random element in $\mathbb{G}_1$ (recall that $g_j = g^{(\alpha^j)}$). Algorithm $\mathcal{B}$ proceeds as follows.

Initialization. The selective (identity, subset) game begins with $\mathcal{A}$ first outputting a multi-node $(S^\star, \mathsf{ID}^\star)$ where $S^\star \subseteq \{1, \ldots, n\}$ and $\mathsf{ID}^\star = (\mathrm{I}_1^\star, \ldots, \mathrm{I}_z^\star) \in {\mathbb{Z}_p}^z$ of depth $z \leq L$ that it intends to attack.

Setup. To generate pk, algorithm $\mathcal{B}$ randomly chooses $u, \sigma \in \mathbb{Z}_p$ and sets

$$v = g^u \cdot \Big(\prod_{j \in S^\star} g_{n+1-j}\Big)^{-1}, \quad y = g^\sigma \cdot \prod_{j=1}^{z} g_{n+1-j}^{-\mathrm{I}_j^\star}.$$

It gives $\mathcal{A}$ the $\mathsf{pk} = (g, \vec{y}_{g,\alpha,n}, v, y)$. Since g, α, u, σ are chosen randomly and independently, pk has an identical distribution to that in the actual construction.

Phase 1. $\mathcal{A}$ issues up to q_{P} private key queries. Consider a query for the private key corresponding to node (i, ID), of which $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_w) \in {\mathbb{Z}_p}^w$ where $w \leq L$. We distinguish two cases according to whether i is in $S^\star$ or not.

If $i \notin S^\star$ then $\mathcal{B}$ responds to the query by first computing a root private key d_i from which it can then construct a private key $d_{i,\mathsf{ID}}$ for the request node (i, ID). In this case, $\mathcal{B}$ computes d_i as $d_i = g_i^u \cdot (\prod_{j \in S^\star} g_{n+1-j+i})^{-1}$. Indeed, we have $d_i = (g^u (\prod_{j \in S^\star} g_{n+1-j})^{-1})^{(\alpha^i)} = v^{(\alpha^i)}$, as required.

If $i \in S^\star$ then from the restriction of the private key query, it must be that ID is neither $\mathsf{ID}^\star$ nor any prefix of $\mathsf{ID}^\star$. We further distinguish two cases according to whether $\mathsf{ID}^\star$ is a prefix of ID or not.

Case 1: $\mathsf{ID}^\star$ is *not* a prefix of ID. Then there must exist $k \leq z$ such that it is the smallest index satisfying $\mathrm{I}_k \neq \mathrm{I}_k^\star$. $\mathcal{B}$ responds to the query by first computing a private key for node $(i, \mathsf{ID}_{|k})$ from which it then constructs a private key for the request node (i, ID). $\mathcal{B}$ picks random elements $s \in \mathbb{Z}_p$. We pose $\tilde{s} = s + \alpha^k / (\mathrm{I}_k - \mathrm{I}_k^\star)$. Note that $\tilde{s}$ is unknown to $\mathcal{B}$. Next, $\mathcal{B}$ generates the private key

$$(a_0, a_1, b_{k+1}, \ldots, b_L) = \Big(v^{(\alpha^i)} \cdot (h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} \cdot y)^{\tilde{s}} \ , \ g^{\tilde{s}} \ , \ h_{k+1}^{\tilde{s}} \ , \ \ldots \ , \ h_L^{\tilde{s}} \Big) \tag{5.6}$$

which is a valid random private key for node $(i, \mathsf{ID}_{|k})$ by definition. We show that $\mathcal{B}$ can compute all elements of this private key given the values that it knows. Recall that

[3] It was later shown in [BBG05, full] that a *truncated* form of Decision n-BDHE, namely the Decision n-wBDHI*, indeed suffices for BBG. This assumption is defined exactly the same as the former except that we change the vector $\vec{y}_{g,\alpha,n}$ to $\vec{y}^*_{g,\alpha,n} := (g_1, \ldots, g_n)$.

$h_j = g_{n+1-j}$. To generate a_0, we first assume that $k < z$, and observe

$$a_0 = g_i^u \Big(\prod_{j \in S^\star} g_{n+1-j+i} \Big)^{-1} \cdot (g^\sigma \cdot \underbrace{\prod_{j=1}^{k-1} g_{n+1-j}^{\mathrm{I}_j - \mathrm{I}_j^\star}}_{=1} \cdot g_{n+1-k}^{\mathrm{I}_k - \mathrm{I}_k^\star} \cdot \prod_{j=k+1}^{z} g_{n+1-j}^{-\mathrm{I}_j^\star})^{\tilde{s}}$$

$$= g_i^u \Big(\prod_{\substack{j \in S^\star \\ j \neq i}} g_{n+1-j+i} \Big)^{-1} \cdot \underbrace{g_{n+1}^{-1} \cdot g_{n+1-k}^{(\mathrm{I}_k - \mathrm{I}_k^\star)\tilde{s}}}_{T_1} \cdot \underbrace{g^{\sigma\tilde{s}}}_{T_2} \cdot \underbrace{\prod_{j=k+1}^{z} g_{n+1-j}^{-\mathrm{I}_j^\star \tilde{s}}}_{T_3} \cdot$$

The term T_1 can be computed by $\mathcal{B}$ since

$$T_1 = g_{n+1}^{-1} \cdot g_{n+1-k}^{(\mathrm{I}_k - \mathrm{I}_k^\star)(s + \frac{\alpha^k}{\mathrm{I}_k - \mathrm{I}_k^\star})} = g_{n+1}^{-1} \cdot g_{n+1-k}^{(\mathrm{I}_k - \mathrm{I}_k^\star)s} \cdot g_{n+1-k}^{\alpha^k} = g_{n+1-k}^{(\mathrm{I}_k - \mathrm{I}_k^\star)s},$$

where the unknown term g_{n+1} is canceled out. The term T_2 can be computed by using g_k, which is not g_{n+1} since $k \leq z \leq L \leq n$. Each term in the product T_3 is computable since $g_{n+1-j}^{\tilde{s}} = g_{n+1-j}^s \cdot g_{n+1-j+k}^{1/(\mathrm{I}_k - \mathrm{I}_k^\star)}$ and for $j = k+1, \ldots, z$, the terms $g_{n+1-j}, g_{n+1-j+k}$ are not equal to g_{n+1} hence can be computed. It is left to consider the case $k = z$. In this case, a_0 is exactly the same as above except that the last product term, i.e., T_3, does not appear. The analysis of computability by $\mathcal{B}$ thus follows from the same argument.

The component a_1 can be generated since $a_1 = g^{\tilde{s}} = g^s \cdot g_k^{1/(\mathrm{I}_k - \mathrm{I}_k^\star)}$. For $j = k+1, \ldots, L$, the value b_j can be computed as $b_j = h_j^{\tilde{s}} = h_j^s \cdot g_{n+1-j+k}^{1/(\mathrm{I}_k - \mathrm{I}_k^\star)}$.

Case 2: $\mathsf{ID}^\star$ is a prefix of ID. Then it holds that $z+1 \leq w$. $\mathcal{B}$ responds to the query by first computing a private key for node $(i, \mathsf{ID}_{|z+1})$ from which it then construct a private key for the request node (i, ID). $\mathcal{B}$ picks random elements $s \in \mathbb{Z}_p$. We pose $\tilde{s} = s + \alpha^{z+1}/\mathrm{I}_{z+1}$. Note that $\tilde{s}$ is unknown to $\mathcal{B}$. Next, $\mathcal{B}$ generates the private key in exactly the same form as Eq.(5.6) (change k to $z+1$, of course). From a similar observation as above, one can show that $\mathcal{B}$ can compute this key.

Challenge. To generate the challenge, $\mathcal{B}$ computes $\mathsf{hdr}^\star$ as (h, h^u, h^σ). It then randomly chooses a bit $b \in \{0,1\}$ and sets $K_b = Z$ and picks a random K_{1-b} in $\mathbb{G}_1$. $\mathcal{B}$ then gives $(\mathsf{hdr}^\star, K_0, K_1)$ to $\mathcal{A}$.

We claim that when $Z = e(g_{n+1}, h)$ (that is, the input to $\mathcal{B}$ is a n-BDHE tuple) then $(\mathsf{hdr}^\star, K_0, K_1)$ is a valid challenge to $\mathcal{A}$ as in a real attack game. To see this, write $h = g^t$ for some (unknown) $t \in \mathbb{Z}_p$. Then, we have that

$$h^u = (g^u)^t = (g^u (\prod_{j \in S^\star} g_{n+1-j})^{-1} (\prod_{j \in S^\star} g_{n+1-j}))^t = (v \prod_{j \in S^\star} g_{n+1-j})^t,$$

$$h^\sigma = \Big(\prod_{j=1}^{z} g_{n+1-j}^{\mathrm{I}_j^\star} \cdot (g^\sigma \cdot \prod_{j=1}^{z} g_{n+1-j}^{-\mathrm{I}_j^\star}) \Big)^t = (h_1^{\mathrm{I}_1^\star} \cdots h_z^{\mathrm{I}_z^\star} \cdot y)^t.$$

Thus, by definition, (h, h^u, h^σ) is a valid encryption of the key $e(g_{n+1}, g)^t$. Also, $e(g_{n+1}, g)^t = e(g_{n+1}, h) = Z = K_b$ and hence $(\mathsf{hdr}^\star, K_0, K_1)$ is a valid challenge.

On the other hand, when Z is random in $\mathbb{G}_1$ (that is, the input to $\mathcal{B}$ is a random tuple) then K_0, K_1 are just random independent elements of $\mathbb{G}_1$.

Phase 2. $\mathcal{A}$ continues to ask queries not issued in Phase 1. $\mathcal{B}$ responds as before.

Guess. Finally, $\mathcal{A}$ outputs $b' \in \{0,1\}$. If $b = b'$ then $\mathcal{B}$ outputs 1 (meaning $Z = e(g_{n+1}, h)$). Otherwise, it outputs 0 (meaning Z is random in $\mathbb{G}_1$).

We see that if $(g, h, \vec{y}_{g,\alpha,n}, Z)$ is sampled from $\mathcal{R}_{BDHE}$ then $\Pr[\mathcal{B}(g, h, \vec{y}_{g,\alpha,n}, Z) = 0] = \frac{1}{2}$. On the other hand, if $(g, h, \vec{y}_{g,\alpha,n}, Z)$ is sampled from $\mathcal{P}_{BDHE}$ then $|\Pr[\mathcal{B}(g, h, \vec{y}_{g,\alpha,n}, Z) = 0] - \frac{1}{2}| \geq \epsilon$. It follows that $\mathcal{B}$ has advantage at least ϵ in solving n-BDHE problem in $\mathbb{G}$. This concludes the proof of Theorem 5.6. □

5.3.3 Extensions

Modification. Recall that for BasicHICBE2 when $L > n$, we created dummy users so that the effective number of users is L. The resulting pk contained $2L+2$ elements of $\mathbb{G}$. We now give a more efficient scheme in this case ($L > n$). First, we change 'n' in all appearances in the description of BasicHICBE2 to 'L' except that the user indexes are as usual: $\{1, \ldots, n\}$. Then we modify the public key to $\mathsf{pk} = (g, g_1, \ldots, g_L, g_{L+2}, \ldots, g_{L+n}, v, y) \in \mathbb{G}^{L+n+2}$, which is of smaller size than that of the above method. This modified scheme is secure under the Decision L-BDHE assumption. However, it can be shown to be secure under a weaker one which is a new assumption that we call Decision $\langle L, n\rangle$-BDHE. (Two values are specified instead of only one). It is defined exactly the same as the Decision L-BDHE except that we change $\vec{y}_{g,\alpha,L}$ to $\vec{y}_{g,\alpha,\langle L,n\rangle} := (g_1, \ldots, g_L, g_{L+2}, \ldots, g_{L+n})$.

Generalizations. Without going into details, we can also combine the BGW system with the Hybrid BB/BBG scheme [BBG05, full §4.2], which can trade off the public key and private key sizes with the ciphertext size. We denote this scheme by BasicHICBE(ω) for parameter $\omega \in [0,1]$. It becomes BasicHICBE1 when $\omega = 1$ and BasicHICBE2 when $\omega = 0$. In this scheme, the public key, the private key, and the ciphertext contains $L^\omega + \max(L^{1-\omega}, n) + n + 1$, $\leq L^{1-\omega} + L^\omega + 1$, and $\leq L^\omega + 2$ elements in $\mathbb{G}$ respectively. It can also be further generalized in the other dimension, namely the user dimension, in the same manner as the generalized BGW scheme [BGW05], which can trade off the public key size with the ciphertext size while the private key size remains fixed. In the resulting scheme, denoted by GenHICBE(ω, μ), for $\mu \in [0,1]$, the public key, the private key, and the ciphertext contains $L^\omega + \max(L^{1-\omega}, n^\mu) + n^\mu + n^{1-\mu}$, $\leq L^{1-\omega} + L^\omega + 1$, $\leq L^\omega + n^{1-\mu} + 1$ elements in $\mathbb{G}$ respectively. Note that it becomes BasicHICBE(ω) when $\mu = 1$.

Chosen Ciphertext Security. We use the conversion due to Canetti, Halevi, and Katz [CHK04] (adapted to the case of HICBE, see Section 5.2.3) to obtain a chosen ciphertext secure scheme. Denote the converted scheme from BasicHICBE(w) and GenHICBE(w) as FullHICBE(w) and FullGenHICBE(w) respectively. This can also be done via a bit more efficient conversions of [BK05] and [BMW05] (although the latter is not generic, it is compatible with our schemes).

Moreover, following [BGW05], we also give a direct scheme which can save the ciphertext size from FullHICBE1 by one group element in $\mathbb{G}$. Denote this scheme as FullHICBE1′. The details are as follows.

Construction 5-3. FullHICBE1′

The first three algorithms Setup, PrivKeyGen, Derive are exactly the same as in Section 5.3.1, except that this time the scheme is for users $1, \ldots, n-1$. The Encrypt, Decrypt are as follow. We will use a signature scheme Sig = (Gen, Sign, Vrfy) as in Section 5.2.3.

$\mathsf{Encrypt}(\mathsf{pk}, S, \mathsf{ID})$**:** Run Gen to obtain a verification key V_{SIG} and a signing key K_{SIG}. Recall that for simplicity, we assume that $V_{\mathsf{SIG}} \in \mathbb{Z}_p$. Pick a random $t \in \mathbb{Z}_p$ and set $K = e(g_{n+1}, g)^t$. Let $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_z)$. Next, set

$$C = \left(g^t,\ (v \cdot g_1^{V_{\mathsf{SIG}}} \cdot \prod_{j \in S} g_{n+1-j})^t\ ,\ F_1(\mathrm{I}_1)^t\ ,\ \ldots\ ,\ F_z(\mathrm{I}_z)^t \right) \in \mathbb{G}^{z+2},$$
$$\mathsf{hdr} = (C, \mathsf{Sign}_{K_{\mathsf{SIG}}}(C), V_{\mathsf{SIG}}),$$

and output the pair (hdr, K).

$\mathsf{Decrypt}(\mathsf{pk}, S, i, d_{i,\mathsf{ID}}, \mathsf{hdr})$**:** Parse the header as $\mathsf{hdr} = (C, \varphi, V_{\mathsf{SIG}}) = ((C_0, C_1, A_1, \ldots, A_z), \varphi, V_{\mathsf{SIG}})$. Also parse $d_{i,\mathsf{ID}} = (a_0, \ldots, a_z) \in \mathbb{G}^{z+1}$. It then checks whether $\mathsf{Vrfy}_{V_{\mathsf{SIG}}}(C, \varphi) = 1$. If not, it outputs $\perp$. Otherwise, it picks a random $\omega \in \mathbb{Z}_p$ and computes

$$\hat{a}_0 = \left(a_0 \cdot g_{i+1}^{V_{\mathsf{SIG}}} \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i} \right) \cdot \left(v \cdot g_1^{V_{\mathsf{SIG}}} \cdot \prod_{j \in S} g_{n+1-j} \right)^{\omega} \quad \text{and} \quad \hat{g} = g_i g^{\omega}.$$

It then outputs $K = e(\hat{g}, C_1) \cdot \prod_{j=1}^{z} e(A_j, a_j)\ /\ e(\hat{a}_0,\ C_0)$.

We emphasize that the decryption requires a randomization. This ensures that for any $i \in S$ the pair $(\hat{a}_0, \hat{g})$ is chosen from the following distribution

$$\hat{a}_0 = g_{n+1}^{-1} \cdot (v \cdot g_1^{V_{\mathsf{SIG}}} \cdot \prod_{j \in S} g_{n+1-j})^r \cdot \prod_{j=1}^{k} F_j(\mathrm{I}_j)^{s_j} \quad \text{and} \quad \hat{g} = g^r.$$

for uniform $r, s_1, \ldots, s_k \in \mathbb{Z}_p$. The independency of i implies that all members of S generate a sample from the same distribution. This randomization is crucial to the security proof.

Theorem 5.7. *Let $\mathbb{G}$ be a bilinear group of prime order p. Suppose the Decision (t, ϵ_1, n)-BDHE assumption holds in $\mathbb{G}$ and the signature scheme is $(t, \epsilon_2, 1)$ strongly existentially unforgeable. Then the FullHICBE1′ system for $n-1$ users and maximum depth L is $(t', q_{\mathsf{P}}, q_{\mathsf{D}}, \epsilon_1 + \epsilon_2)$-IND-sID-sSet-CCA-secure for arbitrary n, L, q_{P}, q_{D}, and any $t' < t - \Theta(\tau_{\mathsf{exp}} L q_{\mathsf{P}})$ where τ_{exp} is the maximum time for an exponentiation in $\mathbb{G}$.*

Proof. Suppose there exists an adversary, $\mathcal{A}$, that has advantage $\epsilon_1 + \epsilon_2$ in attacking the HICBE scheme (for $n-1$ users). We build an algorithm $\mathcal{B}$ that solves the Decision n-BDHE problem in $\mathbb{G}$ with advantage ϵ_1. Algorithm $\mathcal{B}$ is given as input a random n-BDHE challenge $(g, h, \vec{y}_{g,\alpha,n}, Z)$, where $\vec{y}_{g,\alpha,n} = (g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n})$ and Z is either $e(g_{n+1}, h)$ or a random element in $\mathbb{G}_1$ (recall that $g_j = g^{(\alpha^j)}$). Algorithm $\mathcal{B}$ proceeds as follows.

Initialization. The selective (identity, subset) game begins with $\mathcal{A}$ first outputting a multi-node $(S^\star, \mathsf{ID}^\star)$ where $S^\star \subseteq \{1, \ldots, n\}$ and $\mathsf{ID}^\star = (\mathrm{I}_1^\star, \ldots, \mathrm{I}_z^\star) \in {\mathbb{Z}_p}^z$ of depth $z \leq L$ that it intends to attack. If necessary, $\mathcal{B}$ appends random elements in $\mathbb{Z}_p$ to $\mathsf{ID}^\star$ so that $\mathsf{ID}^\star$ is a vector of length L.

Setup. To generate the public key pk, algotrithm $\mathcal{B}$ first runs Gen to obtain $V_{\mathsf{SIG}}^\star$ and $K_{\mathsf{SIG}}^\star$. Next, it chooses a random $u \in \mathbb{Z}_p$ and sets $v = g^u g_1^{-V_{\mathsf{SIG}}^\star} (\prod_{j \in S^\star} g_{n+1-j})^{-1}$. $\mathcal{B}$ then picks

$\beta_1, \ldots, \beta_L \in \mathbb{Z}_p$ at random and define $h_j = g_1^{-\mathrm{I}_j^\star} g^{\beta_j} \in \mathbb{G}$ for $j = 1, \ldots, L$. It gives $\mathcal{A}$ the public key

$$\mathsf{pk} = (g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, v, h_1, \ldots, h_L) \in \mathbb{G}^{2n+L+1}.$$

Note that since g, α, u and the β_j values are chosen uniformly at random, the public key has an identical distribution to that in the actual construction. As before, for $j = 1, \ldots, L$ we define $F_j : \mathbb{Z}_p \to \mathbb{G}$ to be the function $F_j(x) = g_1^x h_j = g_1^{x-\mathrm{I}_j^\star} g^{\beta_j}$.

***Phase 1* (Private key query).** $\mathcal{A}$ issues up to q_P private key queries. The simulation is very similar to that of BasicHICBE1. The only difference is that the term $g_{1+i}^{-V_{\mathsf{SIG}}^\star}$ appears in d_i in the first case and a_0 in the second case (see below). Consider a query for the private key corresponding to node (i, ID), of which $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_w) \in {\mathbb{Z}_p}^w$ where $w \leq L$. We divide to two cases upon whether i is in $S^\star$ or not.

- If $i \notin S^\star$ then $\mathcal{B}$ responds to the query by first computing a root private key d_i from which it can then construct a private key $d_{i,\mathsf{ID}}$ for the request node (i, ID). In this case, $\mathcal{B}$ computes d_i as

$$d_i = g_i^u \cdot g_{1+i}^{-V_{\mathsf{SIG}}^\star} \cdot \Big(\prod_{j \in S^\star} g_{n+1-j+i}\Big)^{-1}.$$

  Indeed we have that $d_i = (g^u g_1^{-V_{\mathsf{SIG}}^\star} (\prod_{j \in S^\star} g_{n+1-j})^{-1})^{(\alpha^i)} = v^{(\alpha^i)}$ as required.

- If $i \in S^\star$ then from the restriction of the private key query, it must be that ID is neither $\mathsf{ID}^\star$ nor any prefix of $\mathsf{ID}^\star$. Let k be the smallest index such that $\mathrm{I}_k \neq \mathrm{I}_k^\star$. Necessarily $1 \leq k \leq w$. $\mathcal{B}$ responds to the query by first computing a private key for node $(i, \mathsf{ID}_{|k})$ from which it then construct a private key for the request node (i, ID). $\mathcal{B}$ picks random elements $s_1, \ldots, s_k \in \mathbb{Z}_p$ and sets

$$a_0 = g_i^u \cdot g_{1+i}^{-V_{\mathsf{SIG}}^\star} \cdot \Big(\prod_{\substack{j \in S^\star \\ j \neq i}} g_{n+1-j+i}\Big)^{-1} \cdot g_n^{\frac{\beta_k}{\mathrm{I}_k - \mathrm{I}_k^\star}} \cdot \left(\prod_{r=1}^{k} F_r(\mathrm{I}_r)^{s_r}\right),$$

$$a_1 = g^{s_1} \ , \ \ldots \ , \ a_{k-1} = g^{s_{k-1}} \ , \ a_k = g_n^{\frac{1}{\mathrm{I}_k - \mathrm{I}_k^\star}} g^{s_k}.$$

  We claim that $(a_0, a_1, \ldots, a_k)$ is a valid random private key for node $(i, \mathsf{ID}_{|k})$. To see this, let $\tilde{s}_k = s_k + \alpha^n/(\mathrm{I}_k - \mathrm{I}_k^\star)$. Exactly the same argument as in BasicHICBE1 (cf. Equation (5.4)) implies that

$$a_0 = g_i^u g_{1+i}^{-V_{\mathsf{SIG}}^\star} \Big(\prod_{j \in S^\star} g_{n+1-j+i}\Big)^{-1} \cdot \Big(\prod_{r=1}^{k-1} F_r(\mathrm{I}_r)^{s_r}\Big) F_k(\mathrm{I}_k)^{\tilde{s}_k} = v^{(\alpha^i)} \cdot \Big(\prod_{r=1}^{k-1} F_r(\mathrm{I}_r)^{s_r}\Big) F_k(\mathrm{I}_k)^{\tilde{s}_k},$$

$$a_1 = g^{s_1} \ , \ \ldots \ , \ a_{k-1} = g^{s_{k-1}} \ , \ a_k = g^{\tilde{s}_k},$$

  where $s_1, \ldots, s_{k-1}, \tilde{s}_k$ are uniform in $\mathbb{Z}_p$. This matched the definition for a private key for node $(i, \mathsf{ID}_{|k})$. Therefore, $(a_0, a_1, \ldots, a_k)$ is a valid private key for node $(i, \mathsf{ID}_{|k})$. $\mathcal{B}$ derives a private key for the requested node (i, ID) from $(a_0, a_1, \ldots, a_k)$ and gives $\mathcal{A}$ the result.

***Phase 1* (Decryption query).** $\mathcal{A}$ issues up to q_D private key queries. Let $(S, \mathsf{ID}, i, \mathsf{hdr})$

be such a query, where $S \subseteq S^\star$ and $i \in S$. Let $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k)$. Let $\mathsf{hdr} = (C, \varphi, V_{\mathsf{SIG}}) = ((C_0, C_1, A_1, \ldots, A_z), \varphi, V_{\mathsf{SIG}})$. $\mathcal{B}$ responds by first checking whether $\mathsf{Vrfy}_{V_{\mathsf{SIG}}}(C, \varphi) = 1$. If not, it returns '`invalid`' to $\mathcal{A}$. Otherwise, we consider three cases:

- $[V_{\mathsf{SIG}} = V^\star_{\mathsf{SIG}}, (C, \varphi) \neq (C^\star, \varphi^\star)]$. In this case, $\mathcal{B}$ outputs a random bit $b \xleftarrow{R} \{0,1\}$ and aborts the simulation.
- $[V_{\mathsf{SIG}} = V^\star_{\mathsf{SIG}}, (C, \varphi) = (C^\star, \varphi^\star)]$. From the restriction of decryption query, it must be that ID is neither $\mathsf{ID}^\star$ nor any prefix of $\mathsf{ID}^\star$. $\mathcal{B}$ responds by simply constructing a private key of node (i, ID) as if responding to a private key query, then uses it to decrypt $C^\star$, and returns the result to $\mathcal{A}$.
- $[V_{\mathsf{SIG}} \neq V^\star_{\mathsf{SIG}}]$. In this case, $\mathcal{B}$ picks random elements $r, s_1, \ldots, s_k \in \mathbb{Z}_p$ and sets

$$\begin{aligned}
\hat{g} &= g^r \cdot g_n^{\frac{1}{V_{\mathsf{SIG}} - V^\star_{\mathsf{SIG}}}}, \\
\hat{a}_0 &= \hat{g}^u \cdot g_1^{r(V_{\mathsf{SIG}} - V^\star_{\mathsf{SIG}})} \cdot \prod_{j \in S^\star \setminus S} (g^r_{n+1-j} \cdot g_{2n+1-j}^{\frac{-1}{V_{\mathsf{SIG}} - V^\star_{\mathsf{SIG}}}}) \cdot \prod_{j=1}^{k} F_j(\mathrm{I}_j)^{s_j}, \\
\hat{a}_1 &= g^{s_1} \ , \ \ldots \ , \ \hat{a}_k = g^{s_k}.
\end{aligned}$$

  $\mathcal{B}$ then responds with $K = e(\hat{g}, C_1) \cdot \prod_{j=1}^{k} e(A_j, \hat{a}_j) \ / \ e(\hat{a}_0, \ C_0)$. To see that this respond is as in a real attack game, we pose $\tilde{r} = r + \alpha^n/(V_{\mathsf{SIG}} - V^\star_{\mathsf{SIG}})$ and observe that

$$\hat{a}_0 = g_{n+1}^{-1} \cdot (v \cdot g_1^{V_{\mathsf{SIG}}} \cdot \prod_{j \in S} g_{n+1-j})^{\tilde{r}} \cdot \prod_{j=1}^{k} F_j(\mathrm{I}_j)^{s_j} \quad \text{and} \quad \hat{g} = g^{\tilde{r}}.$$

  Furthermore, since $r, \ s_1, \ldots, s_k$ are uniform in $\mathbb{Z}_p$ (and so does $\tilde{r}$), we have that $(\hat{g}, \hat{a}_0, \ldots, \hat{a}_k)$ is correctly distributed as in the real decryption algorithm. Therefore, $\mathcal{B}$'s response to the query is identical to $\mathsf{Decrypt}(\mathsf{pk}, S, i, d_{i,\mathsf{ID}}, \mathsf{hdr})$, as required.

Challenge. To generate the challenge, $\mathcal{B}$ computes $C = (h, h^u, h^{\beta_1}, \ldots, h^{\beta_z})$ and lets $\mathsf{hdr}^\star = (C, \mathsf{Sign}_{K^\star_{\mathsf{SIG}}}(C), V^\star_{\mathsf{SIG}})$. It then randomly chooses a bit $b \in \{0,1\}$ and sets $K_b = Z$ and picks a random K_{1-b} in $\mathbb{G}_1$. $\mathcal{B}$ then gives $(\mathsf{hdr}^\star, K_0, K_1)$ to $\mathcal{A}$.

We claim that when $Z = e(g_{n+1}, h)$ (that is, the input to $\mathcal{B}$ is a n-BDHE tuple) then $(\mathsf{hdr}^\star, K_0, K_1)$ is a valid challenge to $\mathcal{A}$ as in a real attack game. To see this, write $h = g^t$ for some (unknown) $t \in \mathbb{Z}_p$. Then, we have that

$$h^u = (g^u)^t = (g^u (g_1^{V^\star_{\mathsf{SIG}}} \prod_{j \in S^\star} g_{n+1-j})^{-1} (g_1^{V^\star_{\mathsf{SIG}}} \prod_{j \in S^\star} g_{n+1-j}))^t = (v \prod_{j \in S^\star} g_{n+1-j})^t,$$

and that $h^{\beta_j} = (g^{\beta_j})^t = F_j(\mathrm{I}^\star_j)^t$, for $j = 1, \ldots, z$. Thus, by definition, $(h, h^u, h^{\beta_1}, \ldots, h^{\beta_z})$ is a valid encryption of the key $e(g_{n+1}, g)^t$. Furthermore, $e(g_{n+1}, g)^t = e(g_{n+1}, h) = Z = K_b$ and hence $(\mathsf{hdr}^\star, K_0, K_1)$ is a valid challenge to $\mathcal{A}$.

On the other hand, when Z is random in $\mathbb{G}_1$ (that is, the input to $\mathcal{B}$ is a random tuple) then K_0, K_1 are just random independent elements of $\mathbb{G}_1$.

Phase 2. $\mathcal{A}$ continues to ask queries not issued in Phase 1. $\mathcal{B}$ responds as before.

Guess. Finally, $\mathcal{A}$ outputs $b' \in \{0,1\}$ for guessing b. If $b = b'$ then $\mathcal{B}$ outputs 1 (meaning $Z = e(g_{n+1}, h)$). Otherwise, it outputs 0 (meaning Z is random in $\mathbb{G}_1$).

We see that if $(g, h, \vec{y}_{g,\alpha,n}, Z)$ is sampled from $\mathcal{R}_{BDHE}$ then $\Pr[\mathcal{B}(g, h, \vec{y}_{g,\alpha,n}, Z) = 0] = \frac{1}{2}$. On the other hand, if $(g, h, \vec{y}_{g,\alpha,n}, Z)$ is sampled from $\mathcal{P}_{BDHE}$ then $|\Pr[\mathcal{B}(g, h, \vec{y}_{g,\alpha,n}, Z) = 0] - \frac{1}{2}| \geq \epsilon_1 + \epsilon_2 - \Pr[\mathsf{abort}]$, where abort denotes the event that $\mathcal{B}$ aborted during the simulation. This is since when $(g, h, \vec{y}_{g,\alpha,n}, Z)$ is sampled from $\mathcal{P}_{BDHE}$ the simulation is perfect if $\mathcal{B}$ does not abort. It follows that $\mathcal{B}$ has advantage at least $\epsilon_1 + \epsilon_2 - \Pr[\mathsf{abort}]$ in solving n-BDHE problem in $\mathbb{G}$.

Next we claim that $\Pr[\mathsf{abort}] < \epsilon_2$. Otherwise $\mathcal{A}$ can be used to forge a signature with probability ϵ_2. We construct another simulator $\mathcal{B}_2$ that knows the master secret, γ, but receives V_{SIG} as a challenge in an existential forgery game. $\mathcal{B}_2$ makes one (and only one) query to generate the signature of $C^\star$ to construct the challenge $\mathsf{hdr}^\star$. But the event abort lets $\mathcal{B}_2$ to know another valid message-signature pair $(C, \varphi) \neq (C^\star, \varphi^\star)$. $\mathcal{B}_2$ then just outputs this and wins the game. This concludes the proof of Theorem 5.7. □

Adaptive Security. An IND-aID-sSet-CCA-secure scheme can be constructed by combining the BGW system with Waters' HIBE [Wat05] in essentially the same way as previous two schemes.

5.4 Forward-Secure Public-key Broadcast Encryption

In this section, we describe the notion of FS-BE schemes, followed by a generic construction from HICBE and a direct construction. We end up this section by comparing their performances.

5.4.1 Syntax for FS-BE

Forward-Secure BE. The syntax of a forward-secure public-key broadcast encryption (FS-BE) scheme is introduced in [YFDL04]. Following [BGW05], for simplicity we define it as a key encapsulation mechanism. A key-evolving broadcast encryption is made up of six randomized algorithms. Via $(\mathsf{pk}, \mathsf{msk}_0) \xleftarrow{R} \mathsf{Setup}(n, T)$, where n is the number of receiver and T is the total number of time period, the setup algorithm produces a public key pk and an initial master private key msk_0; via $\mathsf{msk}_{i,\tau} \xleftarrow{R} \mathsf{MasUpdate}(\tau, \mathsf{msk}_{\tau-1})$ the master key update algorithm outputs a new private key $\mathsf{msk}_{i,\tau}$ of user i for time period τ; via $\mathsf{sk}_{i,\tau} \xleftarrow{R} \mathsf{Regist}(i, \tau, \mathsf{pk}, \mathsf{msk}_\tau)$ the center outputs a private key $\mathsf{sk}_{i,\tau}$ of user i for time period τ; via $\mathsf{sk}_{i,\tau} \xleftarrow{R} \mathsf{Update}(i, \tau, \mathsf{sk}_{i,\tau-1})$ the user i updates his private key to $\mathsf{sk}_{i,\tau}$ for a consecutive time period; via $(\mathsf{hdr}, K) \xleftarrow{R} \mathsf{Encrypt}(\mathsf{pk}, S, \tau)$, where S is the set of recipients, a sender outputs a pair (hdr, K), a header and a message encryption key; via $K \xleftarrow{R} \mathsf{Decrypt}(\mathsf{pk}, S, i, \mathsf{sk}_{i,\tau}, \mathsf{hdr})$ a recipient $i \in S$ outputs $K \in \mathcal{K}$. A scheme is correct if (1) when $\mathsf{pk}, \mathsf{msk}_\tau, \mathsf{sk}_{i,\tau-1}$ are correctly generated, the private key output either from $\mathsf{Regist}(i, \tau, \mathsf{pk}, \mathsf{msk}_\tau)$ or from $\mathsf{Update}(i, \tau, \mathsf{sk}_{i,\tau-1})$ must conform the same distribution; (2) $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$ are consistent (in the standard way).

5.4.2 Security Notions for FS-BE

We define chosen ciphertext security of a key-evolving broadcast encryption by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Both $\mathcal{C}$ and $\mathcal{A}$ are given n, T as input.

Setup. The challenger $\mathcal{C}$ runs $\mathsf{Setup}(n, T)$ to obtain a public key pk and the initial master key msk_0. It then gives the public key pk to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ adaptively issues queries $q_1, \ldots, q_\mu$ where query q_k is one of the following:

- Private key query $\langle i, \tau \rangle$. $\mathcal{C}$ responds by running algorithm $\mathsf{MasUpdate}$ and Regist to derive the private key $\mathsf{sk}_{i,\tau}$, corresponding to user i at time τ and sends $\mathsf{sk}_{i,\tau}$ to $\mathcal{A}$.
- Master key query $\langle \tau \rangle$. $\mathcal{C}$ responds by running algorithm $\mathsf{MasUpdate}$ to obtain msk_τ and gives it to $\mathcal{A}$.
- Decryption query $\langle S, \tau, i, \mathsf{hdr} \rangle$ where $i \in S$. $\mathcal{C}$ responds by running $\mathsf{MasUpdate}$ and Regist to derive $\mathsf{sk}_{i,\tau}$. It then gives to $\mathcal{A}$ the output from $\mathsf{Decrypt}(\mathsf{pk}, S, i, \mathsf{sk}_{i,\tau}, \mathsf{hdr})$.

Challenge. Once $\mathcal{A}$ decides that Phase 1 is over, it outputs $(S^\star, \tau^\star)$ which is the target recipient set and the time it wants to attack, where $S^\star \subseteq \{1, \ldots, n\}$ and $\tau^\star \leq T$. The only restriction is that $\mathcal{A}$ did not previously issue a private key query for $\langle i, \tau \rangle$ such that $i \in S^\star$ and $\tau \leq \tau^\star$ or a master key query for $\tau \leq \tau^\star$. $\mathcal{C}$ then compute $(\mathsf{hdr}^\star, K) \stackrel{R}{\leftarrow} \mathsf{Encrypt}(\mathsf{pk}, S^\star, \tau^\star)$ where $K \in \mathcal{K}$. Next $\mathcal{C}$ picks a random $b \in \{0, 1\}$. It sets $K_b = K$ and picks a random K_{1-b} in $\mathcal{K}$. It then gives $(\mathsf{hdr}^\star, K_0, K_1)$ to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ issues additional queries $q_{\mu+1}, \ldots, q_\nu$ where query q_k is one of the following:

- Private key query $\langle i, \tau \rangle$ where either $i \in S^\star$ and $\tau > \tau^\star$ or $i \notin S^\star$ with arbitrary τ.
- Master key query $\langle \tau \rangle$ where $\tau > \tau^\star$.
- Decryption query $\langle S, \tau, i, \mathsf{hdr} \rangle$ where $i \in S$ and $S \subseteq S^\star$. The only constraint is that $\mathsf{hdr} \neq \mathsf{hdr}^\star$ if $\tau \leq \tau^\star$.

In both cases, $\mathcal{C}$ responds as in Phase 1. These queries may be adaptive.

Guess Finally $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$ for b and wins the game if $b = b'$.

We refer to such an adversary $\mathcal{A}$ as an $\mathsf{IND\text{-}aFS_2\text{-}aSet\text{-}CCA}$ adversary and the above game as the $\mathsf{IND\text{-}aFS_2\text{-}aSet\text{-}CCA}$ game. The $\mathsf{IND\text{-}aFS_1\text{-}aSet\text{-}CCA}$ game is the exactly the same one except that adversary is not allowed to ask master key queries. Weaker notions of security can be defined by modifying the above game similarly to the case of HICBE. Following a similar terminology, we have 4 possible combinations: the game $\mathsf{IND\text{-}xFS_i\text{-}ySet\text{-}CCA}$ where $(\mathsf{x}, \mathsf{y}) \in \{(\mathsf{a}, \mathsf{a}), (\mathsf{a}, \mathsf{s}), (\mathsf{s}, \mathsf{a}), (\mathsf{s}, \mathsf{s})\}$, corresponding to whether $\tau^\star$ and/or $S^\star$ must be disclosed before the Setup phase or not.

We define the advantage of the adversary $\mathcal{A}$ in attacking the key-evolving broadcast encryption scheme $\mathcal{E}$ in the game $\mathsf{IND\text{-}xFS_2\text{-}ySet\text{-}CCA}$ as $\mathsf{AdvFSBE_{xy,2}}(\mathcal{E}, \mathcal{A}) = |\Pr[b = b'] - \frac{1}{2}|$, where the probability is over the random bits used by $\mathcal{C}$ and $\mathcal{A}$ in that game.

Definition 5.8. We say that a key-evolving broadcast encryption scheme $\mathcal{E}$ is $(t, q_\mathsf{P}, q_\mathsf{M}, q_\mathsf{D}, \epsilon)$-$\mathsf{IND\text{-}xFS_2\text{-}ySet\text{-}CCA}$-secure if for any t-time $\mathsf{IND\text{-}xFS_2\text{-}ySet\text{-}CCA}$ adversary $\mathcal{A}$ that makes at most q_P chosen private key queries, q_M chosen master key queries, and at most q_D chosen decryption queries, we have that $\mathsf{AdvFSBE_{xy,2}}(\mathcal{E}, \mathcal{A}) < \epsilon$.

Definition 5.9. We say that a key-evolving broadcast encryption scheme $\mathcal{E}$ is $(t, q_\mathsf{P}, q_\mathsf{D}, \epsilon)$-$\mathsf{IND\text{-}xFS_1\text{-}ySet\text{-}CCA}$-secure if $\mathcal{E}$ is $(t, q_\mathsf{P}, 0, q_\mathsf{D}, \epsilon)$-$\mathsf{IND\text{-}xFS_2\text{-}ySet\text{-}CCA}$-secure.

Definition 5.10. We say that a key-evolving broadcast encryption scheme $\mathcal{E}$ is $(t, q_\mathsf{P}, q_\mathsf{M}, \epsilon)$-$\mathsf{IND\text{-}xFS_2\text{-}ySet\text{-}CPA}$-secure if $\mathcal{E}$ is $(t, q_\mathsf{P}, q_\mathsf{M}, 0, \epsilon)$-$\mathsf{IND\text{-}xFS_2\text{-}ySet\text{-}CCA}$-secure. (And similar for $\mathsf{FS_1}$ notion).

Proposition 5.11. *A key-evolving broadcast encryption for maximum time T which is $(t, q_{\mathsf{P}}, q_{\mathsf{M}}, q_{\mathsf{D}}, \epsilon)$-IND-sFS$_2$-ySet-CCA-secure is also $(t, q_{\mathsf{P}}, q_{\mathsf{M}}, q_{\mathsf{D}}, \epsilon/T)$-IND-aFS$_2$-ySet-CCA-secure. The result also holds for the FS$_1$ case and the CPA case.*

For almost all applications, the IND-xFS$_1$-ySet-CCA-security is sufficient. In this case, it is useful to consider the MasUpdate as a trivial algorithm as we let $\mathsf{msk}_\tau = \mathsf{msk}_0$ for all τ (and simply denote this by msk).

To convert a IND-xFS$_1$-ySet-CCA-secure scheme to achieve IND-xFS$_2$-ySet-CCA-security, where master key queries are also allowed, we simply modify the scheme by letting msk_τ contains all the user keys of time τ. The MasUpdate just runs Update for each key. Obviously, this modified scheme is $(t, q'_{\mathsf{P}}, q'_{\mathsf{M}}, q_{\mathsf{D}}, \epsilon)$-IND-xFS$_2$-ySet-CCA-secure for $q'_{\mathsf{P}} + nq'_{\mathsf{M}} \leq q_{\mathsf{P}}$ if the original scheme is $(t, q_{\mathsf{P}}, q_{\mathsf{D}}, \epsilon)$-IND-xFS$_1$-ySet-CCA-secure.

5.4.3 Conversion C

Given a HICBE scheme, we construct a FS-BE scheme using the "time tree" technique of [CHK03], which was used to construct a forward-secure encryption from a binary tree encryption. The construction here is essentially the same as in [CHK03] except that in our description, the user dimension is introduced.

For a forward-secure BE with T time periods, we image a complete balance binary tree of depth $L = \log_2(T+1) - 1$. Let each node be labeled with a string in $\{0,1\}^{\leq L}$. We assign the root with the empty string. The left and right child of w is labeled $w0$ and $w1$ respectively. From now, to distinguish the abstract 'node' of a HICBE system from nodes in the binary tree, we refer the former as h-node and the latter as usual. Following the notation in [CHK03], we let w^τ to be the τ-th node in a pre-order traversal of the binary tree.[4] Without loss of generality, we assume that $0, 1 \in \mathcal{I}$, the identity space. Hence, we can view a binary string of length $z \leq L$ as an identity vector of length z. Encryption in time τ for a set S of recipient uses the encryption function of the HICBE scheme to the multi-node (S, w^τ). At time τ the private key also contains, beside the private key of h-node (i, w^τ) of the HICBE scheme, all the keys of h-nodes (i, y) where y is a right sibling of the nodes on the path from the root to w^τ in the binary tree. When updating the key to time $\tau + 1$, we compute the private key of h-node $(i, w^{\tau+1})$ and erase the one of (i, w^τ). Since $w^{\tau+1}$ is a left child of w^τ or one of the nodes whose keys are stored as the additional keys at time τ, the derivation can be done, in particular using at most one application of Derive. We denote this conversion as $\mathsf{C}(\cdot)$ and write its formal description as follows.

Given a HICBE scheme HICBE = (Setup, PrivKeyGen, Derive, Encrypt, Decrypt), we construct a key-evolving broadcast encryption scheme C(HICBE) = fsBE′ = (Setup′, MasUpdate′, Regist′, Update′, Encrypt′, Decrypt′) as follows.

Construction 5-4. Forward-secure Broadcast Encryption: C(HICBE)

Setup′(n, T): Let $L = \log_2(T+1) - 1$. Run Setup(n, L) and obtain pk, msk. It then outputs $\mathsf{pk}' = \mathsf{pk}$, and $\mathsf{msk}' = \mathsf{msk}$.

Regist′$(i, \tau, \mathsf{pk}', \mathsf{msk}')$: If $\tau = 0$, then it outputs $\mathsf{sk}'_{i,0} = d_i = \mathsf{PrivKeyGen}(i, \mathsf{pk}', \mathsf{msk}')$. Otherwise ($\tau \geq 1$), it runs Update recursively starting from $\mathsf{sk}'_{i,0}$ to obtain $\mathsf{sk}'_{i,\tau}$ and outputs it.

---

[4] The pre-order traversal is started from the root, $w^1 = \varepsilon$ (the empty string). From a node w it goes to $w0$ if w is not a leaf otherwise it goes to $v1$ if $v0$ is the largest string that is a prefix of w.

$\mathsf{Update}'(i, \tau, \mathsf{sk}'_{i,\tau-1})$**:** Parse $w^{\tau-1}$ as a binary string $\langle \tau-1 \rangle_{L-1} \langle \tau-1 \rangle_{L-2} \ldots \langle \tau-1 \rangle_{L-k} \in \{0,1\}^{\leq L}$. The private key $\mathsf{sk}'_{i,\tau-1}$ is organized as $\mathsf{sk}'_{i,\tau-1} = (a, c_{L-1}, c_{L-2} \ldots, c_0)$ where $a = d_{i,w^{\tau-1}}$ and for $j = 1, \ldots, k$,

$$c_{L-j} = \begin{cases} d_{i,(\langle \tau-1 \rangle_{L-1} \ldots \langle \tau-1 \rangle_{L-j+1} 1)} & \text{if } \langle \tau-1 \rangle_{L-j} = 0, \\ \varepsilon & \text{if } \langle \tau-1 \rangle_{L-j} = 1, \end{cases}$$

and $c_{L-(k+1)} = \cdots = c_0 = \varepsilon$ (the empty string). To update to $\mathsf{sk}'_{i,\tau}$ we consider two cases.

Case 1: [$w^{\tau-1}$ is a leaf] (i.e., $k = L$). Let j^* be the largest j such that $c_{L-j} \neq \varepsilon$. Then let $\bar{a} = d_{i,w^\tau}$ $(= c_{L-j^*})$ and $\bar{c}_{L-j^*} = \varepsilon$. It then outputs

$$\mathsf{sk}'_{i,\tau} = (\bar{a}, c_{L-1}, \ldots, c_{L-j^*+1}, \bar{c}_{L-j^*}, c_{L-j^*-1}, \ldots, c_0).$$

Case 2: [$w^{\tau-1}$ is an internal node] (i.e., $k < L$). We compute $d_{i,w^{\tau-1}0} \leftarrow \mathsf{Derive}(i, w^{\tau-1}0, d_{i,w^{\tau-1}})$ and $d_{i,w^{\tau-1}1} \leftarrow \mathsf{Derive}(i, w^{\tau-1}1, d_{i,w^{\tau-1}})$. Let $\bar{a} = d_{i,w^{\tau-1}0}$ and $\bar{c}_{L-(k+1)} = d_{i,w^{\tau-1}1}$. It outputs

$$\mathsf{sk}'_{i,\tau} = (\bar{a}, c_{L-1}, \ldots, c_{L-k}, \bar{c}_{L-(k+1)}, c_{L-(k+2)}, \ldots, c_0).$$

$\mathsf{Encrypt}'(\mathsf{pk}', S, \tau)$**:** Run $\mathsf{Encrypt}(\mathsf{pk}, S, w^\tau)$.

$\mathsf{Decrypt}'(\mathsf{pk}', S, i, \mathsf{sk}'_{i,\tau}, \mathsf{hdr})$**:** Run $\mathsf{Decrypt}(\mathsf{pk}, S, i, d_{i,w^\tau}, \mathsf{hdr})$.

Theorem 5.12. *Suppose that the scheme HICBE for L levels is $(t, q_\mathsf{P}, q_\mathsf{D}, \epsilon)$-IND-xID-ySet-CCA-secure (resp., $(t, q_\mathsf{P}, \epsilon)$-IND-xID-ySet-CPA-secure) for some $(\mathsf{x}, \mathsf{y}) \in \{(\mathsf{a}, \mathsf{a}), (\mathsf{a}, \mathsf{s}), (\mathsf{s}, \mathsf{a}), (\mathsf{s}, \mathsf{s})\}$. Then the scheme C(HICBE) for T time periods is $(t, q'_\mathsf{P}, q_\mathsf{D}, \epsilon)$-IND-xFS$_1$-ySet-CCA-secure (resp., $(t, q'_\mathsf{P}, \epsilon)$-IND-xFS$_1$-ySet-CPA-secure) for $q'_\mathsf{P} \leq q_\mathsf{P}/L$, where $L = \log(T+1) - 1$.*

Proof. We will deal with the case where $(\mathsf{x},\mathsf{y})=(\mathsf{a},\mathsf{a})$ first. Suppose there exists an adversary, $\mathcal{A}$, that has advantage ϵ in attacking the fsBE' scheme. We build an algorithm $\mathcal{B}$ that has the same advantage ϵ in attacking the HICBE. $\mathcal{B}$ proceeds as follows. First $\mathcal{B}$ receives the public key pk from its challenger. $\mathcal{B}$ just forwards this to $\mathcal{A}$. When $\mathcal{A}$ asks for a private key query (i, τ), $\mathcal{B}$ responds by asking its challenger for queries $(i, \langle \tau \rangle_{L-1} \ldots \langle \tau \rangle_{L-j+1} 1)$ for all j such that $\langle \tau \rangle_{L-j} = 0$. Indeed, these keys comprise the private key for $\mathsf{sk}_{i,\tau}$ by the definition of the scheme. $\mathcal{B}$ thus gives this set of keys to $\mathcal{A}$. When $\mathcal{A}$ asks for a decryption query $(S, \tau, i, \mathsf{hdr})$, $\mathcal{B}$ responds by asking its challenger for the decryption query $(S, w^\tau, i, \mathsf{hdr})$ and forwarding the result to $\mathcal{A}$. When $\mathcal{A}$ decides that phase 1 is over, it will output $(S^\star, \tau^\star)$. $\mathcal{B}$ responds by sending $(S^\star, w^{\tau^\star})$ to its challenger asking for the challenge then forwarding the received challenge ciphertext to $\mathcal{A}$. $\mathcal{B}$ continues to respond to $\mathcal{A}$ in phase 2 in the same way as before. It is easy to see that by the definition of the scheme, the restriction of queries in the game attacking HICBE can be translated exactly to the restriction in the game attacking fsBE'. Therefore, $\mathcal{A}$ will never ask for queries that $\mathcal{B}$ cannot answer. Finally in the guess phase, $\mathcal{B}$ just outputs whatever $\mathcal{A}$ outputs. Since the simulation is perfect, $\mathcal{B}$ has the same advantage as $\mathcal{A}$ in guessing the challenge bit.

For the other cases of (x,y), we construct $\mathcal{A}$ in exactly the same way except that the target $S^\star$ or/and $\tau^\star$ from $\mathcal{A}$ is received at the initialization time. $\mathcal{B}$ responds by outputting $S^\star$ or/and $w^{\tau^\star}$ to its challenger in the initialization time also. The same argument follows. This completes the proof. □

Resulting Efficiency. It is easy to see that the in resulting scheme, the private key size is expanded by the factor $O(\log T)$ while the other parameters are unchanged from the original HICBE scheme (instantiated for $\log(T+1)-1$ levels of identities). We have that the C(BasicHICBE1) scheme achieves ciphertext of size $O(\log T)$ and user private keys of size $O(\log^2 T)$ while the C(BasicHICBE2) scheme achieves ciphertexts of size $O(1)$ and user private keys of size $O(\log^2 T)$.

5.4.4 Direct Construction

We also construct a more efficient but specific FS-BE scheme, which is not built via the generic conversion. This scheme, denoted by DirFSBE, achieves $O(\log T)$ size for both ciphertext and private key, in contrast to the C(BasicHICBE1) and C(BasicHICBE2), in which the private key is of size $O(\log^2 T)$. The scheme is very similar to C(BasicHICBE1). Recall that BasicHICBE1 is based on the BB scheme. With the observation that in the BB scheme, most private key components are unchanged from parent to child, we can thus cut some redundancy and store only different components. This component reuse technique is reminiscent of the "Linear fs-HIBE" scheme in [BBG05]. Note that this scheme can be converted to a CCA-secure version by using the same technique as used for constructing the FullHICBE1′ (Construction 5-3).

The following description is written in an analogous way to the generic conversion C in Section 5.4.3; we encourage the reader to compare each other for better understanding.

Construction 5-5. Direct Forward-secure Broadcast Encryption: DirFSBE

Setup(n, T): Let $L = \log_2(T+1) - 1$. The algorithm is exactly the same as Setup(n, L) of the BasicHICBE1 in Section 5.3.1. The algorithm first picks a random generator $g \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p$. It computes $g_i = g^{(\alpha^i)} \in \mathbb{G}$ for $i = 1, 2, \ldots, n, n+2, \ldots, 2n$. Next, it picks a random $\gamma \in \mathbb{Z}_p$ and sets $v = g^\gamma \in \mathbb{G}$. It then picks random elements $h_1, \ldots, h_L \in \mathbb{G}$. The public key is:

$$\mathsf{pk} = (g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, v, h_1, \ldots, h_L) \in \mathbb{G}^{2n+L+1}.$$

The master key is $\mathsf{msk} = \gamma$. For $j = 1, \ldots, L$, we define $F_j : \mathbb{Z}_p \to \mathbb{G}$ to be the function: $F_j(x) = g_1^x h_j$. The algorithm outputs the public key pk and the master key msk.

Regist$(i, \tau, \mathsf{pk}, \mathsf{msk})$: If $\tau = 0$, then it outputs $d_i = (g_i)^{\mathsf{msk}} = (g_i)^\gamma \in \mathbb{G}$. Otherwise ($\tau \geq 1$), it runs Update recursively starting from $\mathsf{sk}'_{i,0}$ to obtain $\mathsf{sk}'_{i,\tau}$ and outputs it.

Update$(i, \tau, \mathsf{sk}_{i,\tau-1})$: Parse w^τ as a binary string $\langle\tau\rangle_{L-1}\langle\tau\rangle_{L-2}\ldots\langle\tau\rangle_{L-k} \in \{0,1\}^{\leq L}$. To generate the private key $\mathsf{sk}_{i,\tau}$ for user i at time τ, pick random $s_1, \ldots, s_k, s'_1, \ldots, s'_k \in \mathbb{Z}_p$ and output $\mathsf{sk}_{i,\tau} = (a_0, a_1, \ldots, a_L, (c_{L-1}, e_{L-1}), \ldots, (c_0, e_0))$ where

$$a_0 = d_i \cdot \prod_{x=1}^{k} F_x(\langle\tau\rangle_{L-x})^{s_x} \ , \ a_1 = g^{s_1} \ , \ \ldots \ , \ a_{L-k} = g^{s_{L-k}} \ , \ a_{L-k+1} = \cdots = a_L = \varepsilon,$$

and for $j = 1, \ldots, k$,

$$(c_{L-j}, e_{L-j}) = \begin{cases} \left(d_i \cdot \prod_{x=1}^{j-1} F_x(\langle\tau\rangle_{L-x})^{s_x} \cdot F_j(1)^{s'_j} \quad , \quad g^{s'_j} \right) & \text{if } \langle\tau\rangle_{L-j} = 0, \\ (\varepsilon, \varepsilon) & \text{if } \langle\tau\rangle_{L-j} = 1, \end{cases}$$

and $(c_{L-(k+1)}, e_{L-(k+1)}) = \cdots = (c_0, e_0) = (\varepsilon, \varepsilon)$, where ε is the empty string.

Intuitively, $(c_{L-j}, a_1, \ldots, a_{j-1}, e_{L-j})$ forms a proper key of node $(i, \langle\tau\rangle_{L-1}\langle\tau\rangle_{L-2} \ldots \langle\tau\rangle_{L-j+1}1)$ in the BasicHICBE1 scheme. Note that $\mathsf{sk}_{i,\tau} \in \mathbb{G}^{1+L+2L'}$ where $L' = |\{\langle\tau\rangle_{L-j} = 0 : 1 \leq j \leq k\}| \leq k$. Note that $\mathsf{sk}_{i,\tau}$ can be derived from $\mathsf{sk}_{i,\tau-1}$, as required. To see this, we first let $\tilde{k} = |w^{\tau-1}|$ and $\mathsf{sk}_{i,\tau-1} = (\tilde{a}_0, \tilde{a}_1, \ldots, \tilde{a}_L, (\tilde{c}_{L-1}, \tilde{e}_{L-1}), \ldots, (\tilde{c}_0, \tilde{e}_0))$. Consider two cases:

Case 1: [$w^{\tau-1}$ is a leaf] (i.e., $\tilde{k} = L$). Let j^* be the largest j such that $\tilde{c}_{L-j} \neq \varepsilon$. Let

$$\bar{a}_0 = \tilde{c}_{L-j^*} \ , \ \bar{a}_{j^*} = \tilde{e}_{L-j^*} \ \text{ and } \ \bar{c}_{L-j^*} = \varepsilon \ , \ \bar{e}_{L-j^*} = \varepsilon.$$

For $j = 1, \ldots, j^* - 1$, let $\bar{a}_j = \tilde{a}_j$. If $j^* < \tilde{k}$, for $j = j^* + 1, \ldots, \tilde{k}$, let $\bar{a}_j = \varepsilon$. It then outputs

$$\mathsf{sk}_{i,\tau} = \Big(\bar{a}_0, \bar{a}_1, \ldots, \bar{a}_L, (\tilde{c}_{L-1}, \tilde{e}_{L-1}), \ldots, (\tilde{c}_{L-j^*+1}, \tilde{e}_{L-j^*+1}), \\ (\bar{c}_{L-j^*}, \bar{e}_{L-j^*}), (\tilde{c}_{L-j^*-1}, \tilde{e}_{L-j^*-1}), \ldots, (\tilde{c}_0, \tilde{e}_0) \Big).$$

In this case, the same randomness from $\mathsf{sk}_{i,\tau-1}$ is transfer to $\mathsf{sk}_{i,\tau}$ (with some being erased), in particular, if $(\tilde{s}_1, \ldots, \tilde{s}_{\tilde{k}}, \tilde{s}'_1, \ldots, \tilde{s}'_{\tilde{k}})$ is the randomness in $\mathsf{sk}_{i,\tau-1}$, then the randomness in $\mathsf{sk}_{i,\tau}$ is $(\tilde{s}_1, \ldots, \tilde{s}_{j^*}, \tilde{s}'_1, \ldots, \tilde{s}'_{j^*})$.

Case 2: [$w^{\tau-1}$ is an internal node] (i.e., $\tilde{k} < L$). We first pick random elements $\bar{s}_{\tilde{k}+1}, \bar{s}'_{\tilde{k}+1} \in \mathbb{Z}_p$. Then compute

$$\bar{a}_0 = \tilde{a}_0 \cdot F_{\tilde{k}+1}(0)^{\bar{s}_{\tilde{k}+1}} \ , \ \bar{a}_{\tilde{k}+1} = g^{\bar{s}_{\tilde{k}+1}} \text{ and } \bar{c}_{L-(\tilde{k}+1)} = \tilde{a}_0 \cdot F_{\tilde{k}+1}(1)^{\bar{s}'_{\tilde{k}+1}} \ , \ \bar{e}_{L-(\tilde{k}+1)} = g^{\bar{s}'_{\tilde{k}+1}}.$$

For $j = 1, \ldots, \tilde{k}, \tilde{k}+2, \ldots, L$, let $\bar{a}_j = \tilde{a}_j$. It outputs

$$\mathsf{sk}_{i,\tau} = \Big(\bar{a}_0, \bar{a}_1, \ldots, \bar{a}_L, (\tilde{c}_{L-1}, \tilde{e}_{L-1}), \ldots, (\tilde{c}_{L-\tilde{k}}, \tilde{e}_{L-\tilde{k}}), \\ (\bar{c}_{L-(\tilde{k}+1)}, \bar{e}_{L-(\tilde{k}+1)}), (\tilde{c}_{L-(\tilde{k}+2)}, \tilde{e}_{L-(\tilde{k}+2)}), \ldots, (\tilde{c}_0, \tilde{e}_0) \Big).$$

In this case, the randomness in $\mathsf{sk}_{i,\tau}$ is $(\tilde{s}_1, \ldots, \tilde{s}_{\tilde{k}}, \bar{s}_{\tilde{k}+1}\tilde{s}'_1, \ldots, \tilde{s}'_{\tilde{k}}, \bar{s}'_{\tilde{k}+1})$.

Encrypt(pk, S, τ): Pick a random $t \in \mathbb{Z}_p$ and set $K = e(g_{n+1}, g)^t$. The value $e(g_{n+1}, g)$ can be computed as $e(g_n, g_1)$. Write w^τ as $\langle\tau\rangle_{L-1}\langle\tau\rangle_{L-2} \ldots \langle\tau\rangle_{L-z} \in \{0,1\}^{\leq L}$. Next, set

$$\mathsf{hdr} = \left(g^t, \ (v \cdot \prod_{j \in S} g_{n+1-j})^t \ , \ F_1(\langle\tau\rangle_{L-1})^t \ , \ \ldots \ , \ F_z(\langle\tau\rangle_{L-z})^t \right) \in \mathbb{G}^{z+2},$$

and output the pair (hdr, K).

$\mathsf{Decrypt}(\mathsf{pk}, S, i, \mathsf{sk}_{i,\tau}, \mathsf{hdr})$**:** Let $z = |w^\tau|$. Parse the header as $\mathsf{hdr} = (C_0, C_1, A_1, \ldots, A_z) \in \mathbb{G}^{z+2}$. Also parse $\mathsf{sk}_{i,\tau} = (a_0, a_1, \ldots, a_L, (c_{L-1}, e_{L-1}), \ldots, (c_0, e_0))$. Then output

$$K = e(g_i, C_1) \cdot \prod_{j=1}^{z} e(A_j, a_j) \;/\; e(a_0 \cdot \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i},\ C_0).$$

Note that $a_{z+1}, \ldots, a_L, (c_{L-1}, e_{L-1}), \ldots, (c_0, e_0)$ are not used for decryption.

Theorem 5.13. *Let $\mathbb{G}$ be a bilinear group of prime order p. Suppose the Decision (t, ϵ, n)-BDHE assumption holds in $\mathbb{G}$. Then the DirFSBE scheme for n users and maximum time T is $(t', q_\mathsf{P}, \epsilon)$-IND-sFS$_1$-sSet-CPA-secure for arbitrary n, T, q_P, and any $t' < t - \Theta(\tau_{\mathsf{exp}} L q_\mathsf{P})$ where τ_{exp} is the maximum time for an exponentiation in $\mathbb{G}$.*

The proof is similar to that of Theorem 5.5 (for the BasicHICBE1), and thus is omitted here.

5.4.5 Performance Comparisons and Some Terminologies

In this section, we draw comparisons among previous and our FS-BE schemes by wrapping up in Table 1. We name the three previous schemes intuitively from their approaches as follows. The first scheme, $\mathsf{GS}_{(\mathsf{NNL})} \times_{\mathsf{YFDL}} \mathsf{GS}$, is the scheme in [YFDL04] that is instantiated with the subset difference (SD) method [NNL01]. It can be considered intuitively as a "cross-product" (thus, "$\times$" is used) of two copies of the HIBE schemes by [GS02]: one hierarchy provides key derivations required in the SD method and the other provides key derivations for updating keys in the time dimension. This "cross-product" nature results in the expansion factor $O(\log n \log T)$ in performances from the original SD method (in which ciphertext size and private key size is $O(r)$ and $O(\log^2 n)$ resp.).

Boneh et al. [BBG05, Section 5.2] replaced the underlying HIBE from the GS scheme to the basic BBG scheme, which is a more efficient scheme with constant-size ciphertexts, and obtained another scheme, in our terminology, $\mathsf{BBG}_{(\mathsf{NNL})} \times_{\mathsf{YFDL}} \mathsf{BBG}$. This can reduce the ciphertext size to $O(r)$, the same as in the original SD method.

Boneh et al. [BBG05, Section C] subsequently removed the cross-product structure by using two HIBE systems which are "orthogonal" and "complementary" to each other, namely, the basic BBG scheme and the BB scheme. This system, which we denote $\mathsf{BBG}_{(\mathsf{NNL})} \perp_{\mathsf{BBG}} \mathsf{BB}$ ("$\perp$" for orthogonal integration), can thus reduce the expansion factor from $O(\log n \log T)$ to $O(\log n + \log T)$. Note that the term $\log n$ does not appear in the ciphertext size (see Table 1) since the hierarchy corresponding to the SD method is the BBG system. (And recall that BBG achieves constant-size ciphertext).

On the other hand, our schemes outperform the previous schemes in terms of both private key and ciphertext sizes. They inherit good properties from their underlying scheme: (1) the ciphertext size and the private key size are independent of n and r (as in the BGW scheme) and (2) the ciphertext size is constant (as in the BBG scheme) if the BBG scheme is used. Note that the $|S| - 2 = O(n)$ group multiplications for the decryption in our schemes are due to the calculation of $\prod_{j \in S, j \neq i} g_{n+1-j+i}$, which indeed can be precomputed. This suggestion of [BGW05] is useful if the privileged group S is incrementally changed to S': in this case, the computation can be reduced to only $|S' \setminus S| + |S \setminus S'|$ group operations.

Table 5.1: Comparison among previous and our FS-BE schemes (upper and lower table resp.). $T = |\text{total time periods}|$. $n = |\text{all users}|$. $r = |\text{revoked users}|$. The time complexity is expressed in terms of number of operations where [e] is exponentiation, [p] is bilinear pairing, and [m] is group multiplication, while [o] indicates the time complexity for some other process. '⇐' means that it has the same value as the entry on its left.

Params↓	$\mathsf{GS}_{(\mathsf{NNL})} \times_{\mathsf{YFDL}} \mathsf{GS}$ [YFDL04]	$\mathsf{BBG}_{(\mathsf{NNL})} \times_{\mathsf{YFDL}} \mathsf{BBG}$ [BBG05, full §5.2]	$\mathsf{BBG}_{(\mathsf{NNL})} \perp_{\mathsf{BBG}} \mathsf{BB}$ [BBG05, full §C]
Reg time	$O(\log^3 n \log T)$ [e]	⇐	$O((\log^2 n)(\log n + \log T))$ [e]
Enc time	$O(r \log n \log T)$ [e]	⇐	$O(r(\log n + \log T))$ [e]
Dec time	$O(\log n \log T)$ [p] $+ O(r)$ [o]	⇐	$O(\log T)$ [p] $+ O(r)$ [o]
Upd time	$O(\log^3 n)$ [e]	⇐	$O(\log^2 n \log T)$ [e]
|Pub key|	$O(\log n + \log T)$	⇐	⇐
|Pri key|	$O(\log^3 n \log T)$	⇐	$O((\log^2 n)(\log n + \log T))$
|Cipher|	$O(r \log n \log T)$	$O(r)$	$O(r \log T)$

Params↓	C(BasicHICBE1)	DirFSBE	C(BasicHICBE2)	C(GenHICBE(0.5, 0.5))
Reg time	$O(\log T)$ [e]	⇐	⇐	$O(\sqrt{\log T})$ [e]
Enc time	$O(n)$ [m] $+ O(\log T)$ [e]	⇐	⇐	$O(\sqrt{n})$ [m] $+ O(\sqrt{\log T})$ [e]
Dec time	$O(n)$ [m]^9+ $O(\log T)$ [p]	⇐	$O(n)$ [m]^9+$O(1)$ [p]	$O(\sqrt{n})$ [m] $+ O(\sqrt{\log T})$ [p]
Upd time	$O(1)$ [e]	⇐	⇐	⇐
|Pub key|	$O(n + \log T)$	⇐	⇐	$O(\sqrt{n} + \sqrt{\log T})$
|Pri key|	$O(\log^2 T)$	$O(\log T)$	$O(\log^2 T)$	$O(\log^{1.5} T)$
|Cipher|	$O(\log T)$	⇐	$O(1)$	$O(\sqrt{n} + \sqrt{\log T})$

Along the discussion in this section, we may also name our schemes, for future reference, as '$\mathsf{BGW} \odot_{\mathsf{new}} \mathsf{X}$', for a X HIBE, where $\odot_{\mathsf{new}}$ denotes our method of combination. Note that all the considered operations, $\times_{\mathsf{YFDL}}, \perp_{\mathsf{BBG}}, \odot_{\mathsf{new}}$, are neither black-box, generic, nor even well-defined; they provide only rough guides for combinations. These terminologies will be sometimes appropriately abused as HICBE or FS-HIBE or Double-HIBE (cf. Section 5.7), instead of FS-BE here, depended upon the usage of underlying hierarchies which should be clear from the context.

5.5 Public-key Broadcast Encryption with Keyword Search

5.5.1 Definitions and Relation to Anonymous ICBE

BE with Keyword Search. A public-key broadcast encryption with keyword search (BEKS) consists of four algorithms. Via $(\mathsf{pk}, \{\mathsf{sk}_1, \ldots, \mathsf{sk}_n\}) \xleftarrow{R} \mathsf{Setup}(n)$ the setup algorithm produces a public key and n user keys; via $C \xleftarrow{R} \mathsf{BEKS}(\mathsf{pk}, S, \mathsf{w})$ a sender encrypts a keyword w to get a ciphertext (C, S) intended for recipients in $S \subseteq \{1, \ldots, n\}$; via $t_{i,\mathsf{w}} \xleftarrow{R} \mathsf{Td}(i, \mathsf{w}, \mathsf{sk}_i)$ the receiver i computes a trapdoor $(t_{i,\mathsf{w}}, i)$ for keyword w and provides it to the gateway; via $b \leftarrow \mathsf{Test}(\mathsf{pk}, i, t_{i,\mathsf{w}}, C, S)$ for $i \in S$ the gateway can test whether C encrypts w where $b = 1$ means "positive" and $b = 0$ means "negative". Here if $i \notin S$ it always outputs '?'.

We describe the right-keyword consistency (correctness), the computational consistency (in the sense of [ABC+05]), and the security notion, which we name IND-xKW-ySet-CPA as follows. The security captures the property that the adversary be unable to distinguish the encryption of chosen keyword with a random one.

5.5.2 Consistency Properties

Right-Keyword Consistency. This property states the correctness of a BEKS scheme in the sense that there should be no true negative error. The scheme is said to be right-keyword consistent if for all $S \subseteq \{1, \ldots, n\}$, all $i \in S$, and all w in the keyword space $\mathcal{W}$, if $(\mathsf{pk}, \{\mathsf{sk}_1, \ldots, \mathsf{sk}_n\}) \xleftarrow{R} \mathsf{Setup}(n)$, $C \xleftarrow{R} \mathsf{BEKS}(\mathsf{pk}, S, \mathsf{w})$, $t_{i,\mathsf{w}} \xleftarrow{R} \mathsf{Td}(i, \mathsf{w}, \mathsf{sk}_i)$, then it must hold that $\mathsf{Test}(\mathsf{pk}, i, t_{i,\mathsf{w}}, C, S) = 1$.

Computational Consistency. This property states the correctness of a BEKS scheme in the sense that no false positive error should be produced by any computationally-bounded adversary. It can be defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Both $\mathcal{C}$ and $\mathcal{A}$ are given n as input. This is generalized from [ABC+05].

Setup. The challenger $\mathcal{C}$ runs $\mathsf{Setup}(n)$ to obtain pk and $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$. It then gives pk to $\mathcal{A}$.

Find. $\mathcal{A}$ outputs $(i^\star, S^\star, \mathsf{w}_0^\star, \mathsf{w}_1^\star)$. $\mathcal{C}$ then computes $C^\star \xleftarrow{R} \mathsf{BEKS}(\mathsf{pk}, S^\star, \mathsf{w}_0^\star)$ and $t_{i^\star,\mathsf{w}_1^\star} \xleftarrow{R} \mathsf{Td}(i^\star, \mathsf{w}_1^\star, \mathsf{sk}_{i^\star})$. If $\mathsf{w}_0^\star \neq \mathsf{w}_1^\star$ and $\mathsf{Test}(\mathsf{pk}, i^\star, t_{i^\star,\mathsf{w}_1^\star}, C^\star, S^\star) = 1$ then $\mathcal{A}$ wins the game.

We refer to such an adversary $\mathcal{A}$ as a $\mathsf{CON\text{-}aKW\text{-}aSet\text{-}CPA}$ adversary. Following a similar terminology as before, we have 4 possible combinations: $\mathsf{CON\text{-}aKW\text{-}aSet\text{-}CPA}$ where $(\mathsf{x}, \mathsf{y}) \in \{(\mathsf{a}, \mathsf{a}), (\mathsf{a}, \mathsf{s}), (\mathsf{s}, \mathsf{a}), (\mathsf{s}, \mathsf{s})\}$, corresponding to whether $(\mathsf{w}_0^\star, \mathsf{w}_1^\star)$ and/or $(i^\star, S^\star)$ must be disclosed before the Setup phase or not. We say that a BEKS scheme $\mathcal{E}$ is (t, ϵ)-$\mathsf{CON\text{-}xKW\text{-}ySet\text{-}CPA}$-secure if for the winning probability of any t-time $\mathsf{CON\text{-}xKW\text{-}ySet\text{-}CPA}$ adversary is less than ϵ.

Note that we do not need to take care about the false-positive type of consistency regarding i, S since S is always included in the ciphertext and i is always included in the trapdoor and the user private key. (In particular, since i, S are needed to be specified as inputs to the Test algorithm).

5.5.3 Security Notion for BEKS

We define semantic security of a BEKS system by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. Both $\mathcal{C}$ and $\mathcal{A}$ are given n as input.

Setup. The challenger $\mathcal{C}$ runs $\mathsf{Setup}(n)$ to obtain a public key pk and the private key $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$. It then gives the public key pk to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ adaptively issues queries $q_1, \ldots, q_\mu$ where query q_k is one of the following:

- Private key query $\langle i \rangle$. $\mathcal{C}$ responds by sending sk_i to $\mathcal{A}$.
- Trapdoor query $\langle i, \mathsf{w} \rangle$. $\mathcal{C}$ responds by running algorithm Td to derive $t_{i,\mathsf{w}}$ and sends to $\mathcal{A}$.

Challenge. Once $\mathcal{A}$ decides that Phase 1 is over, it outputs $(S^\star, \mathsf{w}^\star)$. The only restriction is that $\mathcal{A}$ did not previously issue a private key query for $i \in S^\star$ or a trapdoor query for $\langle i, \mathsf{w}^\star \rangle$ such that $i \in S^\star$. $\mathcal{C}$ then picks a random $b \in \{0, 1\}$ and computes $C^\star \xleftarrow{R} \mathsf{BEKS}(\mathsf{pk}, S^\star, \mathsf{w}_b)$ and return it to $\mathcal{A}$ where $\mathsf{w}_0 = \mathsf{w}^\star$ and w_1 is a random keyword of the same length as $\mathsf{w}^\star$.

Phase 2. $\mathcal{A}$ issues additional queries $q_{\mu+1}, \ldots, q_\nu$ where query q_k is one of the following:

- Private key query $\langle i \rangle$ such that $i \notin S^\star$.

- Trapdoor query $\langle i, \mathsf{w}\rangle$ such that if $i \in S^\star$ then $\mathsf{w} \neq \mathsf{w}^\star$, else w can be arbitrary.

In both cases, $\mathcal{C}$ responds as in Phase 1. These queries may be adaptive.

Guess Finally $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$ for b and wins the game if $b = b'$.

We refer to such an adversary $\mathcal{A}$ as an IND-aKW-aSet-CPA adversary (KW for "keyword") and the above game as the IND-aKW-aSet-CPA game. Following a similar terminology as before, we have 4 possible combinations: the game IND-xKW-ySet-CPA where $(\mathsf{x},\mathsf{y}) \in \{(\mathsf{a},\mathsf{a}),(\mathsf{a},\mathsf{s}),(\mathsf{s},\mathsf{a}),(\mathsf{s},\mathsf{s})\}$, corresponding to whether $\mathsf{w}^\star$ and/or $S^\star$ must be disclosed before the Setup phase or not.

We define the advantage of the adversary $\mathcal{A}$ in attacking the BEKS scheme $\mathcal{E}$ in the game IND-xKW-ySet-CPA as $\mathsf{AdvBEKS}_{\mathsf{xy}}(\mathcal{E},\mathcal{A}) = |\Pr[b = b'] - \frac{1}{2}|$, where the probability is over the random bits used by $\mathcal{C}$ and $\mathcal{A}$ in that game.

Definition 5.14. We say that a BEKS scheme $\mathcal{E}$ is $(t, q_{\mathsf{P}}, q_{\mathsf{T}}, \epsilon)$-IND-xKW-ySet-CPA-secure if for any t-time IND-xKW-ySet-CPA adversary $\mathcal{A}$ that makes at most q_{P} chosen private key queries and q_{T} chosen trapdoor queries, we have that $\mathsf{AdvBEKS}_{\mathsf{xy}}(\mathcal{E},\mathcal{A}) < \epsilon$.

5.5.4 Conversion K

The conversion of [ABC+05] that compiles any anonymous IBE into a PEKS can be generalized to a broadcast version straightforwardly. More concretely, we construct BEKS from ICBE as follows. $\mathsf{Setup}_{\mathsf{BEKS}}(n)$ can be constructed from $\mathsf{Setup}_{\mathsf{ICBE}}$ and $\mathsf{PrivKeyGen}_{\mathsf{ICBE}}$ by relating the same pk, and $\mathsf{sk}_i = d_i$. We let the remaining algorithms work as follows: $t_{i,\mathsf{w}} \xleftarrow{R} \mathsf{Td}(i,\mathsf{w},\mathsf{sk}_i) = \mathsf{Derive}_{\mathsf{ICBE}}(i,\mathsf{w},d_i)$; $(C_1,C_2) \xleftarrow{R} \mathsf{BEKS}(\mathsf{pk},S,\mathsf{w}) = \mathsf{Encrypt}_{\mathsf{ICBE}}(\mathsf{pk},S,\mathsf{w})$; and $\mathsf{Test}(\mathsf{pk},i,t_{\mathsf{w}},(C_1,C_2),S)$ outputs '?' if $i \notin S$, else outputs 1 if $\mathsf{Decrypt}_{\mathsf{ICBE}}(\mathsf{pk},S,i,t_{\mathsf{w}},C_1) = C_2$, else outputs 0. Denote this conversion as $\mathsf{K}(\cdot)$. Its correctness is immediate from that of ICBE; here $t_{i,\mathsf{w}}, C_1, C_2$ are related to $d_{i,\mathsf{w}}, \mathsf{hdr}, K$ in the ICBE scheme respectively. Note that our conversion is a little bit different from (and simpler than) that of [ABC+05] since we have formalized the ICBE as KEM. We have the following result, which is generalized from [ABC+05].

Theorem 5.15. *If the scheme ICBE is ANO-xID-ySet-CPA$[\{1\}]$-secure for some $(\mathsf{x},\mathsf{y}) \in \{(\mathsf{a},\mathsf{a}),(\mathsf{a},\mathsf{s}),(\mathsf{s},\mathsf{a}),(\mathsf{s},\mathsf{s})\}$, then the BEKS scheme K(ICBE) is IND-xKW-ySet-CPA-secure Further, if ICBE is semantically secure, then K(ICBE) is computationally consistent.*

The proof is exactly the same as that of the IBE-to-PEKS conversion of [ABC+05] except for only two differences: (1) our conversion is based on the ICBE that is formalized as KEM, and (2) we have to simulate also the private key oracle, instead of only the trapdoor oracle. But these make no problem and the proof is immediate. We thus omit the details here.

5.6 Difficulty on Constructing Anonymous HICBE

As one may expect, the first attempt is to use our integration method to combine the BGW system with the anonymous HIBE, BW, by Boyen-Waters [BW06]. Somewhat surprisingly and unfortunately, the resulting HICBE scheme, $\mathsf{BGW} \odot_{\mathsf{new}} \mathsf{BW}$, is *not* ANO-sID-sSet-CPA-secure. Essentially, this is precisely due to the implicit orthogonality of BGW and BW (which has a BB-like structure). Such a property enables any user outside the target subset

$S^\star$ to use the independent part of private keys corresponding to the BW portion to easily distinguish whether a ciphertext is intended for $(S^\star, \mathsf{ID}^\star)$ or $(S^\star, R)$ for random R, thus breaking anonymity. Dilemmatically, on the one hand, this orthogonality enables us to prove the confidentiality of the combined scheme; on the other hand, this very property gives an attack to the anonymity.

In this section, we explain why the $\mathsf{BGW} \odot_{\mathsf{new}} \mathsf{BW}$ HICBE system does not preserve the anonymity from the stand-alone BW HIBE system.

Background. We first recall the reason why the BB and BBG IBE systems are not anonymous [ABC+05, BW06]. In both systems, the ciphertext are of the form $(g^t, F(\mathsf{ID})^t) \in \mathbb{G}^2$, for some computable F (see Eq.(5.1),(5-2)). Denote it by (G, F). To break the anonymity, one just checks whether $e(\mathsf{G}, F(\mathsf{ID})) \stackrel{?}{=} e(g, \mathsf{F})$ to see if this ciphertext was intended for ID or not.

An elegant idea in [BW06] to prevent such a check is to "split" one of the term G or F; moreover, intuitively speaking, this splitting is done in a "linear" way, so that bilinearity will enable only the receiver, who knows some secret, to recover a useful information. Depending on whether G-term or F-term is splitted, we have two initial candidates of ciphertext to be used in a modified scheme as follows:

$$1.\ \Big(\underline{(g^x)^{t-t_1}, (g^y)^{t_1}}, F(\mathsf{ID})^t \Big) \in \mathbb{G}^3 \quad \text{or} \quad 2.\ \Big(g^t, \underline{(\ F(\mathsf{ID})^x)^{t-t_1}, (F(\mathsf{ID})^y)^{t_1}} \Big) \in \mathbb{G}^3,$$

where the underlying pairs represent the result from splitting; x, y are master secrets, which "destroy" the information for checking; t, t_1 are the randomness of the ciphertext. In [BW06], the first type underlies their first IBE scheme in their primer section (which we will denote BW_1), while the second underlies their full HIBE scheme (denoted BW_2). BW_1 is a little bit more efficient than BW_2 (when considered as IBE): the size of pk in the first and second are 6 and 8 elements in $\mathbb{G}$, both plus 1 elements in $\mathbb{G}_1$, while the other overheads are the same. The security is based on the Decision Linear assumption.

Integrating BGW. To integrate the BGW scheme with the BW system, we use our methodology similarly as before, along with some further modification. Concretely, we have to add re-randomization and delegation parts for the user root private keys, besides the term $(g_i)^\gamma$, which was the only element of private key. These parts must be essentially depended on the master key of the BW portion (and hence must be output from PrivKeyGen) since, as seen from the stand-alone BW system, intuitively these terms let the receiver recovers useful information to decrypt the ciphertext whose structure once seemingly has been destroyed by the "linear splitting" technique. Furthermore, for only type (1), we must also split the term $C_1 = (v \cdot \prod_{j \in S} g_{n+1-j})^t$ (see Eq.(5.1)), otherwise the adversary can check $e(C_1, F(\mathsf{ID})) \stackrel{?}{=} e(v \cdot \prod_{j \in S} g_{n+1-j}, \mathsf{F})$. For simplicity, we will confine to only the case of (non-hierarchical) ICBE schemes; indeed, we will state some negative results, the confinement only makes our result stronger. We will consider the integration to both types, BW_1 and BW_2.

In the following, we describe the $\mathsf{BGW} \odot_{\mathsf{new}} \mathsf{BW}_1$ and $\mathsf{BGW} \odot_{\mathsf{new}} \mathsf{BW}_2$ ICBE system and their (in)security analysis.

Construction 5-6. BGW $\odot_{\mathsf{new}}$ BW$_1$ ICBE Scheme

$\mathsf{Setup}(n, L)$**:** Let $\mathbb{G}$ be a bilinear group of prime order p. The algorithm first picks a random generator $g \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p$. It computes $g_i = g^{(\alpha^i)} \in \mathbb{G}$ for $i = 1, 2, \ldots, n, n+2, \ldots, 2n$. Next, it picks a random $\gamma \in \mathbb{Z}_p^\times$ and sets $v = g^\gamma \in \mathbb{G}$. It then picks random elements $h_0, h_1 \in \mathbb{G}$ and $a_1, b_1, a_2, b_2 \in \mathbb{Z}_p$ The public key is:

$$\mathsf{pk} \leftarrow \left(\begin{pmatrix} g^{a_j}, & g_1^{a_j}, & \ldots, & g_n^{a_j}, & g_{n+2}^{a_j}, & \ldots, & g_{2n}^{a_j}, & v^{a_j} \\ g^{b_j}, & g_1^{b_j}, & \ldots, & g_n^{b_j}, & g_{n+2}^{b_j}, & \ldots, & g_{2n}^{b_j}, & v^{b_j} \end{pmatrix}_{j=1,2}, \; h_0, h_1 \right).$$

The master key is $\mathsf{msk} = (\gamma, a_1, b_1, a_2, b_2)$. The algorithm outputs pk and msk.

$\mathsf{PrivKeyGen}(i, \mathsf{pk}, \mathsf{msk})$**:** Pick random elements $(\rho_1, \rho_1', \rho_2, \rho_2') \in (\mathbb{Z}_p)^4$. Let

$$\begin{aligned} d_i^{\mathsf{main}} &= \begin{pmatrix} A_i, B_i \end{pmatrix} \leftarrow \begin{pmatrix} (g_i^\gamma)^{a_1}, & (g_i^\gamma)^{b_1} \end{pmatrix} \\ d_i^{\mathsf{rerand}} &= \begin{pmatrix} f, f' \end{pmatrix} \leftarrow \begin{pmatrix} g^{\rho_1 a_1 b_1 + \rho_2 a_2 b_2}, & g^{\rho_1' a_1 b_1 + \rho_2' a_2 b_2} \end{pmatrix} \\ d_i^{\mathsf{deleg}} &= \begin{pmatrix} \mathsf{a}_{\ell,1}, & \mathsf{a}'_{\ell,1}, & \mathsf{b}_{\ell,1}, & \mathsf{b}'_{\ell,1}, \\ \mathsf{a}_{\ell,2}, & \mathsf{a}'_{\ell,2}, & \mathsf{b}_{\ell,2}, & \mathsf{b}'_{\ell,2} \end{pmatrix}_{\ell=0,1} \leftarrow \begin{pmatrix} h_\ell^{a_1\rho_1}, & h_\ell^{a_1\rho_1'}, & h_\ell^{b_1\rho_1}, & h_\ell^{b_1\rho_1'}, \\ h_\ell^{a_2\rho_2}, & h_\ell^{a_2\rho_2'}, & h_\ell^{b_2\rho_2}, & h_\ell^{b_2\rho_2'} \end{pmatrix}_{\ell=0,1} \end{aligned}$$

The algorithm outputs $d_i = d_i^{\mathsf{main}} || d_i^{\mathsf{rerand}} || d_i^{\mathsf{deleg}}$.

$\mathsf{Derive}(i, \mathsf{ID}, d_i)$**:** Pick $r, r' \in \mathbb{Z}_p$. Let

$$d_{i,\mathsf{ID}} = \begin{pmatrix} f_{i,\mathsf{ID}}, \\ A_{i,\mathsf{ID},1}, \quad B_{i,\mathsf{ID},1}, \\ A_{i,\mathsf{ID},2}, \quad B_{i,\mathsf{ID},2} \end{pmatrix} \leftarrow \begin{pmatrix} f^r \cdot f'^{r'}, \\ A_i \cdot (\mathsf{a}_{0,1}^r \mathsf{a}_{0,1}'^{r'}) \cdot (\mathsf{a}_{1,1}^r \mathsf{a}_{1,1}'^{r'})^{\mathsf{ID}}, \quad B_i \cdot (\mathsf{b}_{0,1}^r \mathsf{b}_{0,1}'^{r'}) \cdot (\mathsf{b}_{1,1}^r \mathsf{b}_{1,1}'^{r'})^{\mathsf{ID}}, \\ (\mathsf{a}_{0,2}^r \mathsf{a}_{0,2}'^{r'}) \cdot (\mathsf{a}_{1,2}^r \mathsf{a}_{1,2}'^{r'})^{\mathsf{ID}}, \quad (\mathsf{b}_{0,2}^r \mathsf{b}_{0,2}'^{r'}) \cdot (\mathsf{b}_{1,2}^r \mathsf{b}_{1,2}'^{r'})^{\mathsf{ID}} \end{pmatrix}$$

$\mathsf{Encrypt}(\mathsf{pk}, S, \mathsf{ID})$**:** Pick a random $t, t_1, t_2 \in \mathbb{Z}_p$ and set $K = e(g_{n+1}, g)^{a_1 b_1 t}$, which can be computed from $e((g_n^{a_1}), (g_1^{b_1}))^t$. (Note that $(g_n^{a_1})$ and $(g_1^{b_1})$ are contained in the public key). Next, set

$$\mathsf{hdr} = \begin{pmatrix} F, \\ \begin{pmatrix} C_{(a),1}, & C_{(b),1}, \\ C_{(a),2}, & C_{(b),2} \end{pmatrix}, \\ \begin{pmatrix} V_{(a),1}, & V_{(b),1}, \\ V_{(a),2}, & V_{(b),2} \end{pmatrix} \end{pmatrix} \leftarrow \begin{pmatrix} \left(h_0 \cdot h_1^{\mathsf{ID}}\right)^t, \\ \begin{pmatrix} (g^{a_1})^{t_1}, & (g^{b_1})^{t-t_1}, \\ (g^{a_2})^{t_2}, & (g^{b_2})^{t-t_2} \end{pmatrix}, \\ \begin{pmatrix} ((v^{a_1}) \cdot \prod_{j \in S} (g_{n+1-j}^{a_1}))^{t_1}, & ((v^{b_1}) \cdot \prod_{j \in S} (g_{n+1-j}^{b_1}))^{t-t_1} \\ ((v^{a_2}) \cdot \prod_{j \in S} (g_{n+1-j}^{a_2}))^{t_2}, & ((v^{b_2}) \cdot \prod_{j \in S} (g_{n+1-j}^{b_2}))^{t-t_2} \end{pmatrix} \end{pmatrix}$$

and output the pair (hdr, K). Here it should be clear which terms are from pk.

$\mathsf{Decrypt}(\mathsf{pk}, S, i, d_{i,\mathsf{ID}}, \mathsf{hdr})$**:** Compute

$$K = e(F, f_{i,\mathsf{ID}}) \cdot \cdot \prod_{\kappa=1}^{2} \left(\frac{e\Big((g_i^{a_\kappa}), V_{(b),\kappa}\Big) \cdot e\Big((g_i^{b_\kappa}), V_{(a),\kappa}\Big)}{e\Big(A_{i,\mathsf{ID},\kappa} \cdot \prod_{\substack{j \in S \\ j \neq i}} (g_{n+1-j+i}^{a_\kappa}),\ C_{(b),\kappa}\Big) \cdot e\Big(B_{i,\mathsf{ID},\kappa} \cdot \prod_{\substack{j \in S \\ j \neq i}} (g_{n+1-j+i}^{b_\kappa}),\ C_{(a),\kappa}\Big)} \right)$$

We leave the correctness verification to the reader. To convince that our $\mathsf{BGW} \odot_{\mathsf{new}} \mathsf{BW}_1$ is a natural and correct candidate in the first place, we show that it satisfies minimum requirements in the following proposition.

Proposition 5.16. *Let $\mathbb{G}$ be a bilinear group of prime order p. We have the following.*

1. *Suppose the Decision (t, ϵ, n)-BDHE assumption holds in $\mathbb{G}$. Then the $\mathsf{BGW} \odot_{\mathsf{new}} \mathsf{BW}_1$ ICBE system for n users is $(t', q_{\mathsf{P}}, \epsilon')$-IND-sID-sSet-CPA-secure for arbitrary n and q_{P} with $t' \simeq t$ and $\epsilon' \simeq \epsilon - \Theta(q_{\mathsf{P}}/p)$.*

2. *Suppose the Decision (t, ϵ)-Linear assumption holds in $\mathbb{G}$. Then the $\mathsf{BGW} \odot_{\mathsf{new}} \mathsf{BW}_1$ ICBE system for n users is secure against a* restricted *ANO-sID-sSet-CPA$[\{1\}]$ where the private key queries are confined to only $\langle i \in S^\star, \mathsf{ID}\rangle$ where ID is neither $\mathsf{ID}^\star$ nor its prefix.*

We give only the proof idea. For the confidentiality proof, the simulator will simulate the challenge ciphertext and the BGW portion of the private keys using the given D-BDHE instance, while it chooses the randomness due to the BW portion, namely (a_1, a_2, b_1, b_2), itself. This can be done exactly the same way as our previous two schemes, except that those known blinding factors from the BW portion are required to be lifted appropriately. The re-randomization and delegation parts of private keys can be simulated since they are independent of unknown terms.

For the *restricted* anonymity proof, the simulator will simulate the game by using the given D-Linear instance, while it chooses α, γ itself. The proof of the stand-alone BW system enables the simulation of private key when ID is neither $\mathsf{ID}^\star$ nor its prefix, which is exactly the same restriction as in our restricted game. Therefore the simulation can be done in the same as in the stand-alone BW system, except that the factor $\alpha^i \gamma$ in the core part of the key is required to be lifted appropriately.

Theorem 5.17. *The $\mathsf{BGW} \odot_{\mathsf{new}} \mathsf{BW}_1$ ICBE system is not ANO-sID-sSet-CPA$[\{1\}]$-secure.*

Proof. Consider an adversary that outputs $(S^\star, \mathsf{ID}^\star)$ at the initialization. It then asks a private key query for a user $i \notin S^\star$. When it is given the challenge ciphertext, it checks whether

$$e(F^\star, f) \stackrel{?}{=} \prod_{\kappa=1}^{2} \Big(e(\mathsf{a}_{0,\kappa} \mathsf{a}_{1,\kappa}^{\mathsf{ID}^\star}, C^\star_{(b),\kappa}) \cdot e(\mathsf{b}_{0,\kappa} \mathsf{b}_{1,\kappa}^{\mathsf{ID}^\star}, C^\star_{(a),\kappa}) \Big)$$

If so, it returns 0, else returns 1. It is easy to verify that the advantage of the adversary over the game $\mathsf{ANO\text{-}sID\text{-}sSet\text{-}CPA}[\{1\}]$ is $\geq 1 - 1/p$. □

Construction 5-7. BGW $\odot_{\mathsf{new}}$ BW$_2$ ICBE Scheme

$\mathsf{Setup}(n, L)$**:** Let $\mathbb{G}$ be a bilinear group of prime order p. The algorithm first picks a random generator $g \in \mathbb{G}$ and a random $\alpha \in \mathbb{Z}_p$. It computes $g_i = g^{(\alpha^i)} \in \mathbb{G}$ for $i = 1, 2, \ldots, n, n+2, \ldots, 2n$. Next, it picks a random $\gamma \in \mathbb{Z}_p^\times$ and sets $v = g^\gamma \in \mathbb{G}$. It then picks random elements $h_{0,1}, h_{0,2}, h_{1,1}, h_{1,2} \in \mathbb{G}$ and $a_1, b_1, a_2, b_2 \in \mathbb{Z}_p$ The public key is:

$$\mathsf{pk} \leftarrow \left(g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, v, \ \left(h_{\ell,1}^{a_1}, h_{\ell,1}^{b_1}, h_{\ell,2}^{a_2}, h_{\ell,2}^{b_2} \right)_{\ell=0,1} \right).$$

The master key is $\mathsf{msk} = (\gamma, a_1, b_1, a_2, b_2)$. The algorithm outputs pk and msk.

$\mathsf{PrivKeyGen}(i, \mathsf{pk}, \mathsf{msk})$**:** Pick random elements $(\rho_1, \rho_1', \rho_2, \rho_2') \in (\mathbb{Z}_p)^4$. Let

$$\begin{aligned}
d_i^{\mathsf{main}} &= \begin{pmatrix} w \end{pmatrix} \leftarrow \begin{pmatrix} g_i^\gamma \end{pmatrix} \\
d_i^{\mathsf{rerand}} &= \begin{pmatrix} \mathsf{a}_1, & \mathsf{a}_1', & \mathsf{b}_1, & \mathsf{b}_1', \\ \mathsf{a}_2, & \mathsf{a}_2', & \mathsf{b}_2, & \mathsf{b}_2' \end{pmatrix} \leftarrow \begin{pmatrix} g^{a_1\rho_1}, & g^{a_1\rho_1'}, & g^{b_1\rho_1}, & g^{b_1\rho_1'}, \\ g^{a_2\rho_2}, & g^{a_2\rho_2'}, & g^{b_2\rho_2}, & g^{b_2\rho_2'} \end{pmatrix} \\
d_i^{\mathsf{deleg}} &= \left(f_\ell, \ f_\ell' \right)_{\ell=0,1} \leftarrow \left(h_{\ell,1}^{a_1 b_1 \rho_1} \cdot h_{\ell,2}^{a_2 b_2 \rho_2}, \ h_{\ell,1}^{a_1 b_1 \rho_1'} \cdot h_{\ell,2}^{a_2 b_2 \rho_2'} \right)_{\ell=0,1}
\end{aligned}$$

The algorithm outputs $d_i = d_i^{\mathsf{main}} || d_i^{\mathsf{rerand}} || d_i^{\mathsf{deleg}}$.

$\mathsf{Derive}(i, \mathsf{ID}, d_i)$**:** Pick $r, r' \in \mathbb{Z}_p$. Let

$$d_{i,\mathsf{ID}} = \begin{pmatrix} w_{i,\mathsf{ID}}, \\ X_{i,\mathsf{ID},1}, & Y_{i,\mathsf{ID},1}, \\ X_{i,\mathsf{ID},2}, & Y_{i,\mathsf{ID},2} \end{pmatrix} \leftarrow \begin{pmatrix} w \cdot (f_0^r \cdot f_0'^{r'}) \cdot (f_1^r \cdot f_1'^{r'})^{\mathsf{ID}}, \\ \mathsf{a}_1^r \cdot \mathsf{a}_1'^{r'}, & \mathsf{b}_1^r \cdot \mathsf{b}_1'^{r'}, \\ \mathsf{a}_2^r \cdot \mathsf{a}_2'^{r'}, & \mathsf{b}_2^r \cdot \mathsf{b}_2'^{r'} \end{pmatrix}$$

$\mathsf{Encrypt}(\mathsf{pk}, S, \mathsf{ID})$**:** Pick a random $t, t_1, t_2 \in \mathbb{Z}_p$ and set $K = e(g_{n+1}, g)^{a_1 b_1 t}$, which can be computed from $e((g_n^{a_1}), (g_1^{b_1}))^t$. (Note that $(g_n^{a_1})$ and $(g_1^{b_1})$ are contained in the public key). Next, set $\mathsf{hdr} =$

$$\begin{pmatrix} \begin{pmatrix} F_{(a),1}, & F_{(b),1}, \\ F_{(a),2}, & F_{(b),2} \end{pmatrix}, \\ C, \\ V \end{pmatrix} \leftarrow \begin{pmatrix} \begin{pmatrix} \left((h_{0,1}^{a_1}) \cdot (h_{1,1}^{a_1})^{\mathsf{ID}} \right)^{t_1}, & \left((h_{0,1}^{b_1}) \cdot (h_{1,1}^{b_1})^{\mathsf{ID}} \right)^{t-t_1}, \\ \left((h_{0,2}^{a_2}) \cdot (h_{1,2}^{a_2})^{\mathsf{ID}} \right)^{t_2}, & \left((h_{0,2}^{b_2}) \cdot (h_{1,2}^{b_2})^{\mathsf{ID}} \right)^{t-t_2} \end{pmatrix}, \\ g^t, \\ (v \cdot \textstyle\prod_{j \in S} g_{n+1-j})^t \end{pmatrix}$$

and output the pair (hdr, K). Here it should be clear which terms are from pk.

$\mathsf{Decrypt}(\mathsf{pk}, S, i, d_{i,\mathsf{ID}}, \mathsf{hdr})$**:** Compute

$$K = \frac{e(g_i, V)}{e(w_{i,\mathsf{ID}} \prod_{\substack{j \in S \\ j \neq i}} g_{n+1-j+i}, \ C)} \cdot \prod_{\kappa=1}^{2} \left(e\Big(F_{(a),\kappa}, \ Y_{i,\mathsf{ID},\kappa}\Big) \cdot e\Big(F_{(b),\kappa}, \ X_{i,\mathsf{ID},\kappa}\Big) \right)$$

We can state a similar proposition for confidentiality and restricted anonymity as in the previous scheme. We omit here. Now we state that the scheme is not anonymous as follows.

Theorem 5.18. *The* $\mathsf{BGW} \odot_{\mathsf{new}} \mathsf{BW}_2$ *ICBE system is not* ANO-sID-sSet-CPA$[\{1\}]$*-secure.*

Proof. Consider an adversary that outputs $(S^\star, \mathsf{ID}^\star)$ at the initialization. It then asks a private key query for a user $i \notin S^\star$. When it is given the challenge ciphertext, it checks whether

$$e(f_0 \cdot f_1^{\mathsf{ID}^\star},\ C^\star) \stackrel{?}{=} \prod_{\kappa=1}^{2} \Big(e(F^\star_{(a),\kappa}, \mathsf{b}_\kappa) \cdot e(F^\star_{(b),\kappa}, \mathsf{a}_\kappa) \Big)$$

If so, it returns 0, else returns 1. It is easy to verify that the advantage of the adversary over the game ANO-sID-sSet-CPA$[\{1\}]$ is $\geq 1 - 1/p$. □

5.7 Anonymous HICBE Construction

From the failed attempt, it is then natural to implement both the broadcast and identity dimensions from two non-orthogonal sub-systems. Therefore, we construct our scheme, denoted AnonHICBE, from the YFDL (cross-product) approach instantiated to two copies of the BW hierarchies, or in our terminology, $\mathsf{BW} \times_{\mathsf{YFDL}} \mathsf{BW}$. Such a scheme was already noticed, without any details, in [BW06] in the context of anonymous FS-BE. (Thus, in particular, the two hierarchies correspond to time-tree and anonymous identity dimension there, as opposed to broadcast and anonymous identity dimension here). There, the authors merely pointed the reference to [YFDL04, BBG05] for how to construct a forward-secure version of their BW system. We found that, however, the approaches of $\mathsf{BW} \perp_{\mathsf{BBG}} \mathsf{BB}$ and $\mathsf{BW} \perp_{\mathsf{BBG}} \mathsf{BBG}$ do *not* preserve the anonymity of BW. This is precisely due to the orthogonality of BW and BB or BBG; indeed, an attack can be argued essentially in a similar manner as the one on $\mathsf{BGW} \odot_{\mathsf{new}} \mathsf{BW}$ as described above, we thus omit the detail of the attack here.

To conclude, we construct our default AnonHICBE scheme via the $\mathsf{BW}_{(\mathsf{BE})} \times_{\mathsf{YFDL}} \mathsf{BW}$ approach, which we are the first to flesh out explicit details here. For generality, we present our scheme for arbitrary subset-cover broadcast encryption BE. When instantiating with the SD method, the resulting anonymous ICBE system achieves ciphertext of size $O(r \log n)$ and private key of size $O(\log^4 n)$ for the user level (level 0) and private key of size $O(\log^3 n)$ for level 1. These translate to the sizes of ciphertext, private key, and trapdoor in BEKS respectively.

We now describe the AnonHICBE scheme postponed from Section 5.6. We do it in two steps: first we construct anonymous HICBE from anonymous double-HIBE in a black-box manner. We then later specify the construction of anonymous double-HIBE, which is done via the $\mathsf{BW} \times_{\mathsf{YFDL}} \mathsf{BW}$ approach.

5.7.1 Double-HIBE

We first reformalize the notion of Multiple-HIBE, first mentioned in [YFDL04]. Such a primitive allows multiple hierarchies that are evolved simultaneously. For our purpose it suffices to consider the case of two hierarchies. It consists of 6 algorithms:

Setup(L_1, L_2)**:** Takes as input the maximum depth L_1, L_2 of the two hierarchies. It outputs a public key pk and a master key msk.

$\mathsf{Extract}(\mathsf{pk}, \mathsf{msk}, \mathsf{ID}^{(1)}, \mathsf{ID}^{(2)})$**:** Take as input $\mathsf{pk}, \mathsf{msk}$ and a pair of identity tuple. It outputs the private key $d_{\mathsf{ID}^{(1)},\mathsf{ID}^{(2)}}$.

$\mathsf{Derive}_1(\mathsf{ID}^{(1)}, \mathsf{ID}^{(2)}, d_{\mathsf{ID}^{(1)}_{|z_1-1},\mathsf{ID}^{(2)}})$**:** Takes as input a pair of identity tuple $\mathsf{ID}^{(1)}$ of depth z_1 and $\mathsf{ID}^{(2)}$ of depth z_2 and the private key of $(\mathsf{ID}^{(1)}_{|z_1-1}, \mathsf{ID}^{(2)})$. It outputs the private key $d_{\mathsf{ID}^{(1)},\mathsf{ID}^{(2)}}$.

$\mathsf{Derive}_2(\mathsf{ID}^{(1)}, \mathsf{ID}^{(2)}, d_{\mathsf{ID}^{(1)},\mathsf{ID}^{(2)}_{|z_2-1}})$**:** Takes as input a pair of identity tuple $\mathsf{ID}^{(1)}$ of depth z_1 and $\mathsf{ID}^{(2)}$ of depth z_2 and the private key of $(\mathsf{ID}^{(1)}, \mathsf{ID}^{(2)}_{|z_2-1})$. It outputs the private key $d_{\mathsf{ID}^{(1)},\mathsf{ID}^{(2)}}$.

$\mathsf{Encrypt}(\mathsf{pk}, \mathsf{ID}^{(1)}, \mathsf{ID}^{(2)})$**:** Takes as input pk and an identity tuple pair $(\mathsf{ID}^{(1)}, \mathsf{ID}^{(2)})$. It outputs a pair (hdr, K).

$\mathsf{Decrypt}(\mathsf{pk}, d_{\mathsf{ID}^{(1)},\mathsf{ID}^{(2)}}, \mathsf{hdr})$**:** Takes as input the pk, the private key of an identity tuple pair $(\mathsf{ID}^{(1)}, \mathsf{ID}^{(2)})$, and the header hdr. It outputs K.

The scheme is correct if the private keys $d_{\mathsf{ID}^{(1)},\mathsf{ID}^{(2)}}$ output from either one of $\mathsf{Extract}, \mathsf{Derive}_1, \mathsf{Derive}_2$ must conform the same distribution, and the $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$ are consistent. The security notions for confidentiality, $\mathsf{IND\text{-}x_1ID_1\text{-}x_2ID_2\text{-}CCA}$, and anonymity, $\mathsf{ANO\text{-}x_1ID_1\text{-}x_2ID_2\text{-}CCA}[\Delta_1, \Delta_2]$, can be defined in the standard manner.

5.7.2 From Double-HIBE to HICBE

We now describe our $\mathsf{AnonHICBE}$ Scheme. The description of our scheme below is reminiscent of the FS-BE scheme of [YFDL04], albeit we do it more generally: our description can be instantiated to any arbitrary subset-cover based BE [NNL01]. We follow the formalization via the tree decomposition framework of [AI05a]. We begin with some terminologies.

Definition 5.19. (Complement-Cover Set System) For a map $\mathsf{c} : \mathbb{Z}^2_{>0} \to \mathbb{Z}_{>0}$, a set system $\mathcal{S} = \{S_1, ..., S_m\}$ over a base set $N = \{1, ..., n\}$ is c-*complement-cover* if there is a polynomial-time algorithm $\mathsf{Cover}_{\mathcal{S}}$ such that upon input any subset $R \subset N$, outputs $\{S_{i_1}, ..., S_{i_t}\}$ for some $1 \leq i_1, \ldots, i_t \leq m$ such that $N \setminus R = \bigcup_{j=1}^{t} S_{i_j}$ and that $t \leq \mathsf{c}(n, |R|)$.

A public-key version of subset-cover-based broadcast encryption will be based on a *tree decomposition* of the inclusion poset $(\mathcal{S}, \subseteq)$. We first review such a notion. First, a poset $\mathcal{S}$ can be represented by a directed acyclic graph, denoted by $\mathsf{DAG}(\mathcal{S}) = (V, E)$, whose the node set being $V = \mathcal{S}$ and the (directed) edge set being $E = \{(S, S') : S \subset S';\ S, S' \in \mathcal{S}\}$. A graph decomposition of a poset $\mathcal{S}$ is a family of connected subgraphs whose sets of nodes partition the set of all nodes in the $\mathsf{DAG}(\mathcal{S})$. When each subgraph is a tree whose edges are directed away from the root, we call it a tree decomposition. From a tree decomposition of $\{(V_i, E_i) : i = 1, \ldots, k\}$ of $\mathcal{S}$, we construct a big tree $\mathcal{T}_{\mathcal{S}} = (V', E')$ by setting $V' = \{\mathsf{r}\} \cup \bigcup_{i=1}^{k} V_i$ and $E' = \{(\mathsf{r}, \mathsf{r}_i) : i = 1, \ldots, k\} \cup \bigcup_{i=1}^{k} E_i$, where r is a newly added root node and r_i is the root node of the subgraph (V_i, E_i). Let D denote the deepest depth of $\mathcal{T}_{\mathcal{S}}$. For $i \in \{1, \ldots, n\}$, let $\mathsf{Reach}(i) = \{S \mid i \in S; i \notin \mathsf{par}(S), S \in V'\}$, where $\mathsf{par}(S)$ is the parent of S in the tree $\mathcal{T}_{\mathcal{S}}$.

For each node S of the tree $\mathcal{T}_{\mathcal{S}}$, we assign a different identity $\mathsf{f}(S) \in \{0,1\}^{\leq \log s}$ where s is the number of all siblings of S in $\mathcal{T}_{\mathcal{S}}$. We let the identity-tuple corresponding to S be the

vector $\mathsf{V}_S = \langle \mathsf{f}(S'_1), \ldots, \mathsf{f}(S'_j), \mathsf{f}(S) \rangle$, where $\mathsf{r}, S'_1, \ldots, S'_j, S$ are the nodes on the path from the root r to S in $\mathcal{T}_{\mathcal{S}}$.

Construction. From a double-HIBE $(\mathsf{Setup}, \mathsf{Extract}, \mathsf{Derive}_1, \mathsf{Derive}_2, \mathsf{Encrypt}, \mathsf{Decrypt})$, we construct a HICBE $(\mathsf{Setup}^\diamond, \mathsf{PrivKeyGen}^\diamond, \mathsf{Derive}^\diamond, \mathsf{Encrypt}^\diamond, \mathsf{Decrypt}^\diamond)$ as follows. DEM is any CCA-secure data encapsulation mechanism.

$\mathsf{Setup}^\diamond(n, L)$**:** Run $\mathsf{Setup}(D, L)$. Let $\mathsf{pk}^\diamond = \mathsf{pk}$ and $\mathsf{msk}^\diamond = \mathsf{msk}$.

$\mathsf{PrivKeyGen}^\diamond(i, \mathsf{pk}^\diamond, \mathsf{msk}^\diamond)$**:** Using msk, it runs $d_{(\mathsf{V}_S,\varepsilon)} \xleftarrow{R} \mathsf{Extract}(\mathsf{pk}, \mathsf{msk}, (\mathsf{V}_S, \varepsilon))$ for each $S \in \mathsf{Reach}(i)$. It then let $d_i^\diamond = \{(S, d_{(\mathsf{V}_S,\varepsilon)}) \mid S \in \mathsf{Reach}(i)\}$.

$\mathsf{Derive}^\diamond(i, \mathsf{ID}, d^\diamond_{i,\mathsf{ID}_{|z-1}})$**:** Parse $d^\diamond_{i,\mathsf{ID}_{|z-1}} = \{(S, d_{(\mathsf{V}_S,\mathsf{ID}_{|z-1})}) \mid S \in \mathsf{Reach}(i)\}$. For each $S \in \mathsf{Reach}(i)$, run $d_{(\mathsf{V}_S,\mathsf{ID})} \xleftarrow{R} \mathsf{Derive}_2(\mathsf{V}_S, \mathsf{ID}, d_{(\mathsf{V}_S,\mathsf{ID}_{|z-1})})$. It then outputs $d^\diamond_{i,\mathsf{ID}} = \{(S, d_{(\mathsf{V}_S,\mathsf{ID})}) \mid S \in \mathsf{Reach}(i)\}$.

$\mathsf{Encrypt}^\diamond(\mathsf{pk}^\diamond, \bar{S}, \mathsf{ID})$**:** To encrypt to the set $\bar{S}$, it first runs $\{S_{j_1}, \ldots, S_{j_t}\} \leftarrow \mathsf{Cover}_{\mathcal{S}}(N \setminus \bar{S})$. For $k = 1, \ldots, t$, it then runs $(\mathsf{hdr}_{j_k}, K_{j_k}) \xleftarrow{R} \mathsf{Encrypt}(\mathsf{pk}, V_{S_{j_k}}, \mathsf{ID})$. It randomly chooses $\bar{K} \in \mathcal{M}_{\mathsf{DEM}}$. It lets $\mathsf{hdr}^\diamond = \langle j_1, \mathsf{hdr}_{j_1}, \mathsf{DEM\text{-}Enc}_{(K_{j_1})}(\bar{K}) \rangle || \ldots || \langle j_t, \mathsf{hdr}_{j_t}, \mathsf{DEM\text{-}Enc}_{(K_{j_t})}(\bar{K}) \rangle$ and $K^\diamond = \bar{K}$. It outputs the pair $(\mathsf{hdr}^\diamond, K^\diamond)$.

$\mathsf{Decrypt}^\diamond(\mathsf{pk}^\diamond, \bar{S}, i, d^\diamond_{i,\mathsf{ID}}, \mathsf{hdr}^\diamond)$**:** Parse $\mathsf{hdr}^\diamond = \langle j_1, \mathsf{hdr}_{j_1}, C_{j_1} \rangle || \ldots || \langle j_t, \mathsf{hdr}_{j_t}, C_{j_t} \rangle$. From the definition of the complement-cover set system, if $i \in \bar{S}$ then there exists j_k such that $i \in S_{j_k}$. Necessarily, S_{j_k} is a descendent of one set in the collection $\mathsf{Reach}(i)$, say set $\tilde{S}$. From $d_{(\mathsf{V}_{\tilde{S}},\mathsf{ID})}$, it thus recursively applies Derive_1 to derive $d_{(\mathsf{V}_{S_{j_k}},\mathsf{ID})}$. It then runs $K_{j_k} \leftarrow \mathsf{Decrypt}(\mathsf{pk}, \mathsf{V}_{S_{j_k}}, \mathsf{ID}, \mathsf{hdr}_{j_k})$ and $\bar{K} \leftarrow \mathsf{DEM\text{-}Dec}_{(K_{j_k})}(C_{j_k})$.

Our full scheme will further use the OR-multiple encryption of [DK05] to achieve CCA security. It can be argued straightforwardly that if the underlying double-HIBE is $\mathsf{IND}\text{-}\mathsf{x}_1\mathsf{ID}_1\text{-}\mathsf{x}_2\mathsf{ID}_2\text{-}\mathsf{CCA}$-secure then the resulting HICBE will be $\mathsf{IND}\text{-}\mathsf{x}_2\mathsf{ID}\text{-}\mathsf{x}_1\mathsf{Set}\text{-}\mathsf{CCA}$-secure. Also, if the underlying double-HIBE is $\mathsf{ANO}\text{-}\mathsf{x}_1\mathsf{ID}_1\text{-}\mathsf{x}_2\mathsf{ID}_2\text{-}\mathsf{CCA}[\emptyset, \Delta]$-secure then the resulting HICBE will be $\mathsf{ANO}\text{-}\mathsf{x}_2\mathsf{ID}\text{-}\mathsf{x}_1\mathsf{Set}\text{-}\mathsf{CCA}[\Delta]$-secure.

Instantiation by the SD method. Consider the balance complete binary tree of n leaves. Denote this tree as BT. For node u, v such that u is an ancestor of v, let subset difference $\mathsf{dif}_{(u,v)}$ be the set of leaves under node u but not under node v. The SD method defines $\mathcal{S}_{\mathsf{SD}} = \{\mathsf{dif}_{(u,v)} : u \text{ is ancestor of } v\} \cup \{N\}$. It is known that $\mathcal{S}_{\mathsf{SD}}$ is $(2r-1)$-complement-cover set system [NNL01]. The tree decomposition $\mathcal{T}_{\mathcal{S}_{\mathsf{SD}}}$ is defined by specifying the edge set $E = \{(\mathsf{dif}_{(u,v_1)}, \mathsf{dif}_{(u,v_2)}) \mid v_1 \text{ is the parent of } v_2 \text{ in } \mathsf{BT}, u \text{ is an ancestor of } v_1\}$. It is known that $|\mathsf{Reach}(i)| = \frac{\log^2 n + \log n}{2} + 1$ [NNL01].

5.7.3 A Construction of Anonymous Double-HIBE

In this section, we explicitly describe the $\mathsf{BW} \times_{\mathsf{YFDL}} \mathsf{BW}$ double-HIBE scheme. Although, such an approach was already noticed by Boyen and Waters [BW06] towards constructing forward-secure version of their anonymous BW HIBE, they did not give an explicit description (indeed, since the main objection of their paper was to construct anonymous HIBE). They only pointed the reference to [YFDL04, BBG05] for how to construct. At this point, it may be misunderstood by a non-savvy reader to use the $\mathsf{BW} \perp_{\mathsf{BBG}} \mathsf{BB}$ approach, as this

methodology was contained in the latter reference (we write the name in our terminology, though). However, as we have already discussed in Section 5.6, the $\mathsf{BW} \perp_{\mathsf{BBG}} \mathsf{BB}$ scheme does *not* preserve anonymity from the stand-alone BW HIBE. Therefore, it is important to give an explicit description to avoid such ambiguities. On the other hand, we will omit the security proof since once the description is clear, the proof can be considered straightforward from the BW scheme.

Let $\mathbb{G}$ be a bilinear group of prime order p. Let $\mathcal{I}_1^{\leq L_1}, \mathcal{I}_2^{\leq L_2}$ be the identity-tuple spaces for both hierarchies in the double-HIBE. Let $H : \mathcal{I}_1^{\leq L_1} \times \mathcal{I}_2^{\leq L_2} \to \mathbb{Z}_p$ be a collision-resistant hash function, where $H(\varepsilon, \varepsilon) = 1$. (Recall that ε is the empty string). When there is no confusion, we denotes $\mathrm{I}_{\langle \ell_1, \ell_2 \rangle} = H(\mathsf{ID}^{(1)}_{|\ell_1}, \mathsf{ID}^{(2)}_{|\ell_2})$ for all $0 \leq \ell_1 \leq z_1$, $0 \leq \ell_2 \leq z_2$, where z_1, z_2 are the lengths of vector $\mathsf{ID}^{(1)}$ and $\mathsf{ID}^{(2)}$ respectively. Thus $\mathrm{I}_{0,0} = 1$. Let $J = (L_1 + 1)(L_2 + 1)$. Let $\mathsf{A} = \{0, \ldots, L_1\} \times \{0, \ldots, L_2\}$. For all $0 \leq \ell_1 \leq L_1$, $0 \leq \ell_2 \leq L_2$, let $\mathsf{R}_{\ell_1, \ell_2} = \mathsf{A} \setminus (\{0, \ldots, \ell_1\} \times \{0, \ldots, \ell_2\})$.

Construction 5-8. Anonymous Double-HIBE

$\mathsf{Setup}(L_1, L_2)$**:** The algorithm first picks a random generator $g \in \mathbb{G}$ and a random $\omega \in \mathbb{Z}_p^\times$. It then picks random elements $\left[\alpha_\kappa,\ \beta_\kappa,\ [\theta_{\kappa,\langle \ell_1,\ell_2\rangle}]_{\langle \ell_1,\ell_2\rangle \in \mathsf{A}}\ \right]_{\kappa=1}^{J+1} \in ((\mathbb{Z}_p^\times)^2 \times (\mathbb{Z}_p)^{J+1})^{J+1}$ then let

$$\mathsf{pk} = \left[\begin{array}{c} \Omega, \\ \left[\ \left[\ a_{\kappa,\langle \ell_1,\ell_2\rangle},\ b_{\kappa,\langle \ell_1,\ell_2\rangle}\ \right]_{\langle \ell_1,\ell_2\rangle \in \mathsf{A}}\ \right]_{\kappa=1}^{J+1} \end{array}\right] \leftarrow \left[\begin{array}{c} e(g,g)^\omega, \\ \left[\ \left[\ g^{\alpha_\kappa \theta_{\kappa,\langle \ell_1,\ell_2\rangle}},\ g^{\beta_\kappa \theta_{\kappa,\langle \ell_1,\ell_2\rangle}}\ \right]_{\langle \ell_1,\ell_2\rangle \in \mathsf{A}}\ \right]_{\kappa=1}^{J+1} \end{array}\right]$$

$$\mathsf{msk} = \left[\begin{array}{c} w, \\ \left[\ a_\kappa,\ b_\kappa,\ \left[\ y_{\kappa,\langle \ell_1,\ell_2\rangle}\ \right]_{\langle \ell_1,\ell_2\rangle \in \mathsf{A}}\ \right]_{\kappa=1}^{J+1} \end{array}\right] \leftarrow \left[\begin{array}{c} g^\omega, \\ \left[\ g^{\alpha_\kappa},\ g^{\beta_\kappa},\ \left[\ g^{\alpha_\kappa \beta_\kappa \theta_{\kappa,\langle \ell_1,\ell_2\rangle}}\ \right]_{\langle \ell_1,\ell_2\rangle \in \mathsf{A}}\ \right]_{\kappa=1}^{J+1} \end{array}\right]$$

$\mathsf{Extract}(\mathsf{pk}, \mathsf{msk}, \mathsf{ID}^{(1)}, \mathsf{ID}^{(2)})$**:** From msk, to generate a private key for a pair of identity-tuples $(\mathsf{ID}^{(1)}, \mathsf{ID}^{(2)})$ of depth $z_1 \leq L_1$ and $z_2 \leq L_2$ respectively, the algorithm first picks random elements $\left[\ \rho_\kappa,\ \left[\ \rho_{\kappa,m}\ \right]_{m=1}^{J+1}\ \right]_{\kappa=1}^{J+1} \in (\mathbb{Z}_p)^{(J+2)(J+1)}$ and lets

$$d^{\mathrm{decrypt}}_{\mathsf{ID}^{(1)},\mathsf{ID}^{(2)}} = \left[\begin{array}{c} k_0, \\ \left[\ k_{\kappa,(a)},\ k_{\kappa,(b)}\ \right]_{\kappa=1}^{J+1} \end{array}\right] \leftarrow \left[\begin{array}{c} w \cdot \prod_{\kappa=1}^{J+1} \prod_{\langle \ell_1,\ell_2\rangle=\langle 0,0\rangle}^{\langle z_1,z_2\rangle} (y_{\kappa,\langle \ell_1,\ell_2\rangle}^{\mathrm{I}_{\langle \ell_1,\ell_2\rangle}})^{\rho_\kappa}, \\ \left[\ a_\kappa^{\rho_\kappa},\ b_\kappa^{\rho_\kappa}\ \right]_{\kappa=1}^{J+1} \end{array}\right]$$

$$d^{\mathrm{rerand}}_{\mathsf{ID}^{(1)},\mathsf{ID}^{(2)}} = \left[\begin{array}{c} f_{m,0}, \\ \left[\ f_{m,\kappa,(a)},\ f_{m,\kappa,(b)}\ \right]_{\kappa=1}^{J+1} \end{array}\right]_{m=1}^{J+1} \leftarrow \left[\begin{array}{c} \prod_{\kappa=1}^{J+1} \prod_{\langle \ell_1,\ell_2\rangle=\langle 0,0\rangle}^{\langle z_1,z_2\rangle} (y_{\kappa,\langle \ell_1,\ell_2\rangle}^{\mathrm{I}_{\langle \ell_1,\ell_2\rangle}})^{\rho_{\kappa,m}}, \\ \left[\ a_\kappa^{\rho_{\kappa,m}},\ b_\kappa^{\rho_{\kappa,m}}\ \right]_{\kappa=1}^{J+1} \end{array}\right]_{m=1}^{J+1}$$

$$d^{\text{deleg}}_{\mathsf{ID}^{(1)},\mathsf{ID}^{(2)}} = \left[\begin{array}{c} h_{\langle \ell_1,\ell_2\rangle}, \\ \left[\, h_{m,\langle \ell_1,\ell_2\rangle} \,\right]_{m=1}^{J+1} \end{array}\right]_{\langle \ell_1,\ell_2\rangle \in \mathsf{R}_{z_1,z_2}} \leftarrow \left[\begin{array}{c} \prod_{\kappa=1}^{J+1} y_{\kappa,\langle \ell_1,\ell_2\rangle}^{\rho_\kappa}, \\ \left[\prod_{\kappa=1}^{J+1} y_{\kappa,\langle \ell_1,\ell_2\rangle}^{\rho_{\kappa,m}}\right]_{m=1}^{J+1} \end{array}\right]_{\langle \ell_1,\ell_2\rangle \in \mathsf{R}_{z_1,z_2}}$$

The private key for $(\mathsf{ID}^{(1)}, \mathsf{ID}^{(2)})$ is $d_{\mathsf{ID}^{(1)},\mathsf{ID}^{(2)}} = d^{\text{decrypt}}_{\mathsf{ID}^{(1)},\mathsf{ID}^{(2)}} || d^{\text{rerand}}_{\mathsf{ID}^{(1)},\mathsf{ID}^{(2)}} || d^{\text{deleg}}_{\mathsf{ID}^{(1)},\mathsf{ID}^{(2)}}$ which is an element in $\mathbb{G}^{2J+3} \times \mathbb{G}^{(2J+3)(J+1)} \times \mathbb{G}^{(J+2)(J-(z_1+1)(z_2+1))}$.

$\mathsf{Derive}_1(\mathsf{ID}^{(1)}, \mathsf{ID}^{(2)}, d_{\mathsf{ID}^{(1)}_{|z_1-1},\mathsf{ID}^{(2)}})$**:** To generate the private key for a pair of identity-tuples $(\mathsf{ID}^{(1)}, \mathsf{ID}^{(2)})$ of depth $z_1 \le L_1$ and $z_2 \le L_2$ from $d_{\mathsf{ID}^{(1)}_{|z_1-1},\mathsf{ID}^{(2)}}$, it recursively computes a sequence of temporary keys as follows.

$$d_{\mathsf{ID}^{(1)}_{|z_1-1},\mathsf{ID}^{(2)}} = s_0 \Rightarrow s_1 \Rightarrow \cdots \Rightarrow s_{z_2} \Rightarrow s_{z_2+1} = d_{\mathsf{ID}^{(1)},\mathsf{ID}^{(2)}}.$$

Let $\mathsf{T}_0 = \mathsf{R}_{z_1-1,z_2}$ and for $1 \le j \le z_2+1$ let $\mathsf{T}_j = \mathsf{R}_{z_1-1,z_2} \setminus (\{z_1\} \times \{0,\ldots,j-1\})$. Note that $\mathsf{T}_{z_2+1} = \mathsf{R}_{z_1,z_2}$. From s_j, it will compute s_{j+1}. First parsing s_j as

$$\left[\begin{array}{c} k_0, \\ \left[\, k_{\kappa,(a)},\ k_{\kappa,(b)} \,\right]_{\kappa=1}^{J+1} \end{array}\right], \left[\begin{array}{c} f_{m,0}, \\ \left[\, f_{m,\kappa,(a)},\ f_{m,\kappa,(b)} \,\right]_{\kappa=1}^{J+1} \end{array}\right]_{m=1}^{J+1}, \left[\begin{array}{c} h_{\langle \ell_1,\ell_2\rangle}, \\ \left[\, h_{m,\langle \ell_1,\ell_2\rangle} \,\right]_{m=1}^{J+1} \end{array}\right]_{\langle \ell_1,\ell_2\rangle \in \mathsf{T}_j}$$

which is an element in $\mathbb{G}^{2J+3} \times \mathbb{G}^{(2J+3)(J+1)} \times \mathbb{G}^{(J+2)(J-(z_1)(z_2+1)-j)}$.
It then picks $[\pi_m, [\pi_{m,m'}]_{m'=1}^{J+1}]_{m=1}^{J+1} \in (\mathbb{Z}_p)^{(J+2)(J+1)}$ and lets

$$s^{\text{decrypt}}_{j+1} \leftarrow \left[\begin{array}{c} (k_0 \prod_{m=1}^{J+1} (f_{m,0})^{\pi_m})(h_{\langle z_1,j\rangle} \prod_{m=1}^{J+1} (h_{m,\langle z_1,j\rangle})^{\pi_m})^{\mathrm{I}_{\langle z_1,j\rangle}}, \\ \left[k_{\kappa,(a)} \prod_{m=1}^{J+1} (f_{m,\kappa,(a)})^{\pi_m},\ k_{\kappa,(b)} \prod_{m=1}^{J+1} (f_{m,\kappa,(b)})^{\pi_m}\right]_{\kappa=1}^{J+1} \end{array}\right]$$

$$s^{\text{rerand}}_{j+1} \leftarrow \left[\begin{array}{c} (\prod_{m=1}^{J+1} (f_{m,0})^{\pi_{m,m'}})(\prod_{m=1}^{J+1} (h_{m,\langle z_1,j\rangle})^{\pi_{m,m'}})^{\mathrm{I}_{\langle z_1,j\rangle}}, \\ \left[\prod_{m=1}^{J+1} (f_{m,\kappa,(a)})^{\pi_{m,m'}},\ \prod_{m=1}^{J+1} (f_{m,\kappa,(b)})^{\pi_{m,m'}}\right]_{\kappa=1}^{J+1} \end{array}\right]_{m'=1}^{J+1}$$

$$s^{\text{deleg}}_{j+1} \leftarrow \left[\begin{array}{c} h_{\langle \ell_1,\ell_2\rangle} \prod_{m=1}^{J+1} (h_{m,\langle \ell_1,\ell_2\rangle})^{\pi_m}, \\ \left[\prod_{m=1}^{J+1} (h_{m,\langle \ell_1,\ell_2\rangle})^{\pi_{m,m'}}\right]_{m'=1}^{J+1} \end{array}\right]_{\langle \ell_1,\ell_2\rangle \in \mathsf{T}_{j+1}}$$

Then it lets $s_{j+1} = s^{\text{decrypt}}_{j+1} || s^{\text{rerand}}_{j+1} || s^{\text{deleg}}_{j+1} \in \mathbb{G}^{2J+3} \times \mathbb{G}^{(2J+3)(J+1)} \times \mathbb{G}^{(J+2)(J-(z_1)(z_2+1)-(j+1))}$.

$\mathsf{Derive}_2(\mathsf{ID}^{(1)}, \mathsf{ID}^{(2)}, d_{\mathsf{ID}^{(1)},\mathsf{ID}^{(2)}_{|z_2-1}})$**:** This algorithm is exactly the same as Derive_1 with the

role of two hierarchies swapped.

$\mathsf{Encrypt}(\mathsf{pk}, \mathsf{ID}^{(1)}, \mathsf{ID}^{(2)})$**:** Pick random elements $t, [t_\kappa]_{\kappa=1}^{J+1} \in (\mathbb{Z}_p)^{J+2}$ and set $K = \Omega^t$. Next, set

$$C = \left(g^t,\ \left[\left(\prod_{\langle \ell_1,\ell_2 \rangle = \langle 0,0 \rangle}^{\langle z_1,z_2 \rangle} b_{\kappa,\langle \ell_1,\ell_2 \rangle}^{\mathrm{I}_{\langle \ell_1,\ell_2 \rangle}} \right)^{t_\kappa},\ \left(\prod_{\langle \ell_1,\ell_2 \rangle = \langle 0,0 \rangle}^{\langle z_1,z_2 \rangle} a_{\kappa,\langle \ell_1,\ell_2 \rangle}^{\mathrm{I}_{\langle \ell_1,\ell_2 \rangle}} \right)^{t-t_\kappa} \right]_{\kappa=1}^{J+1} \right) \in \mathbb{G}^{2J+3},$$

and output the pair (C, K).

$\mathsf{Decrypt}(\mathsf{pk}, d_{\mathsf{ID}^{(1)},\mathsf{ID}^{(2)}}, C)$**:** Parse the ciphertext as $C = \left(C_0, [A_{\kappa,(b)}, A_{\kappa,(a)}]_{\kappa=1}^{J+1} \right) \in \mathbb{G}^{2J+3}$. Also parse $d^{\mathrm{decrypt}}_{\mathsf{ID}^{(1)},\mathsf{ID}^{(2)}} = \left(k_0, [k_{\kappa,(a)}, k_{\kappa,(b)}]_{\kappa=1}^{J+1} \right)$. Then output

$$K = e(k_0,\ C_0)\ /\ \prod_{\kappa=1}^{J+1} e(A_{\kappa,(b)}, k_{\kappa,(a)}) e(A_{\kappa,(a)}, k_{\kappa,(b)}).$$

Theorem 5.20. *Let $\mathbb{G}$ be a bilinear group of prime order p. Suppose the Decision (t,ϵ)-BDH assumption holds in $\mathbb{G}$.*[5] *Then the $\mathsf{BW} \times_{\mathsf{YFDL}} \mathsf{BW}$ double-HIBE scheme for maximum depth L_1, L_2 is $(t', q_\mathsf{P}, \epsilon')$-IND-sID$_1$-sID$_2$-CPA-secure for arbitrary L_1, L_2 and q_P with $t' \simeq t$ and $\epsilon' \simeq \epsilon - \Theta(L_1 L_2 q_\mathsf{P}/p)$.*

Theorem 5.21. *Let $\mathbb{G}$ be a bilinear group of prime order p. Suppose the Decision (t,ϵ)-Linear assumption holds in $\mathbb{G}$. Then the $\mathsf{BW} \times_{\mathsf{YFDL}} \mathsf{BW}$ double-HIBE scheme for maximum depth L_1, L_2 is $(t', q_\mathsf{P}, \epsilon')$-ANO-sID$_1$-sID$_2$-CPA$[\{1,\ldots,L_1\},\{1,\ldots,L_2\}]$-secure for arbitrary L_1, L_2 and q_P with $t' \simeq t$ and $\epsilon' \simeq \epsilon - \Theta(L_1^2 L_2^2 q_\mathsf{P}/p)$.*

The proofs can be done by a straightforward generalization from that of the BW scheme, albeit being quite complicated as the original one already was so. We thus omit them and state only the intuition. The potential attack that was successful to break the anonymity of those orthogonally combined systems is of the form that the adversary who possesses the private key for $(\mathsf{ID}^{(1)}, \mathsf{ID}^{(2)}_{|z_2-1})$ will correctly distinguish whether a ciphertext is intended for $(\mathsf{ID}^{(1)}_{|z_1-1}, \mathsf{ID}^{(2)})$ or $(\mathsf{ID}^{(1)}_{|z_1-1}, \mathsf{X})$ for a random identity-tuple X of depth z_2, something he should not have been able to do if the anonymity were held. However, in the cross-product approach, the attack of this type will not apply since intuitively all the useful information in the key of $(\mathsf{ID}^{(1)}, \mathsf{ID}^{(2)}_{|z_2-1})$ has already been *together-evolved* to index z_1 in such a way that, unlike in orthogonally combined schemes, he cannot unblind to index $z_1 - 1$. Therefore he cannot perform the check of identity as usual.

Efficiency of AnonHICBE. Combining all together, our anonymous HICBE of L levels, instantiated with a $\mathsf{c}_\mathcal{S}$-complement-cover set system $\mathcal{S}$, yields ciphertext of size $O(\mathsf{c}_\mathcal{S}(n,r) \cdot D_\mathcal{S} \cdot L)$ and private key of size $O(|\mathsf{Reach}(i)| \cdot D_\mathcal{S}^2 \cdot L^2)$ for all levels except the last level L at which private key is of size $O(|\mathsf{Reach}(i)| \cdot D_\mathcal{S} \cdot L)$ (since the rerand and deleg parts are no need

[5] The decision BDH problem [Jou00] in $\mathbb{G}$ is stated as follows: given a vector $(g, g^a, g^b, g^c, Z) \in \mathbb{G}^4 \times \mathbb{G}_1$ as input, determine whether $Z = e(g,g)^{abc}$. The decision (t,ϵ)-BDH assumption posits the hardness of this problem and can be formally defined in an analogous way as other assumptions.

anymore for the last level), where $D_{\mathcal{S}}$ is the deepest depth of the tree decomposition $\mathcal{T}_{\mathcal{S}}$. Instantiating with the SD method (where $D_{\mathcal{S}} = \log n, |\mathsf{Reach}(i)| = O(\log^2 n), \mathsf{c}_{\mathcal{S}}(n,r) = O(r)$) yields the scheme with ciphertext of size $O(r \cdot \log n \cdot L)$ and private key of size $O(\log^4 n \cdot L^2)$ for all levels except the last, at which it is of size $O(\log^3 n \cdot L^2)$.

5.8 Some Extended Primitives

In this section, we describe the details for some extensions of FS-BE and BEKS.

BETKS. A public-key broadcast encryption with *temporary* keyword search (BETKS) is a generalization of BEKS (and analogous to PETKS generalized from PEKS) in which a trapdoor can be issued for any desired window of time rather than forever. This can be implemented analogous to PETKS of [ABC+05] by using $\mathsf{ANO\text{-}xID\text{-}ySet\text{-}CPA}[\{1\}]$-secure HICBE of $1+\log T$ levels by putting keywords at the first level and a binary tree time frames on the levels below. In such a scheme, unlike the time-tree of [CHK03], which utilized *all* nodes in the tree as time-nodes, one uses only the ones at the *leaf* level (cf. [Kat02]), i.e., at level $1 + \log T$, as time-nodes and uses the other internal nodes as auxiliary nodes for key delegation, so that any arbitrary set of time period, say $A \subseteq \{1, \ldots, T\}$, can be covered exactly by subtrees outside the minimum Steiner tree over leaves in $\{1, \ldots, T\} \setminus A$ (cf. the complete-subtree method in [NNL01]); the key for this time interval corresponds to all the roots of those cover subtrees. This can be implemented via the $\mathsf{AnonHICBE}$ of Section 5.7.

Forward-Secure BE(T)KS. A forward-secure BE(T)KS scheme, FS-BE(T)KS, is a more esoteric system which enables the autonomous updating for both the private key of each user (done solely by that user i) and trapdoor (done solely by the gateway; furthermore, in the FS-BETKS case, this is allowed to be done only in a specified limited period). These schemes provide the security of any past-time BEKS-encrypted ciphertext even both the present-time trapdoor for a target keyword and the present-time private key for any user in a target set are given to the adversary. Many difficulties arise, for example, it must be ensured that the time where a trapdoor is issued from a user to the gateway is oblivious to a sender, who encrypts a keyword by specifying only the *present* time. This issuing-time-obliviousness property is analogous to the join-time-obliviousness when constructing FS-HIBE [YFDL04]. As expected, we solve this problem by constructing forward-secure anonymous HICBE. First we construct a triple-HIBE (cf. Section 5.7.1) by the approach $\mathsf{BW} \times_{\mathsf{YFDL}} \mathsf{BW} \times_{\mathsf{YFDL}} \mathsf{BW}$, which can be straightforwardly generalized (although complicated) from the $\mathsf{BW} \times_{\mathsf{YFDL}} \mathsf{BW}$ double-HIBE in Section 5.7.3. We then utilize the first hierarchy as time dimension (via the CHK time-tree conversion), and the other two hierarchies together as anonymous HICBE dimension (via the double-HIBE⇒HICBE conversion in Section 5.7.2).

5.9 Conclusions and Open Problems

We presented the first FS-BE schemes with ciphertext and private key size being independent of the number of users in the system. As a building block, we introduced the notion of HICBE, constructed concrete schemes and then converted them via a generic CHK time-tree conversion to obtain FS-BE schemes. A more efficient direct construction of FS-BE scheme was also given. Our methodology provides a method for securely integrating the broadcast encryption scheme of [BGW05] with the HIBE systems of [BB04a, BBG05, Wat05]. We proved the security of our schemes against static adversarial collusion in the standard model

under the Decision n-BDHE assumption. A similar open problem as posed in [BGW05] arises, that is, the question of building a FS-BE scheme with the same parameters as ours but is secure against adaptive collusion (IND-xID-aSet-CCA).

We also introduced the notion of public-key broadcast encryption with keyword search (BEKS) and provided a transform from anonymous (H)ICBE, of which a quasi-efficient construction, based on the cross-product approach [YFDL04] over the anonymous HIBE of [BW06], was explicitly presented. An open problem is to construct a BEKS with ciphertext and private key size being independent of the number of users. We constructively hinted that even adapting the technique used in our FS-BE schemes (which achieved the constant overheads), it may not be easy to do so.

Chapter 6

Conclusions

In this thesis, we have studied the class of *encryption primitives with high functionalities*. This area of research is not only considered one of the main streams of recent advancement in cryptography but some of them have also been already implemented and used as important tools in the real-world systems. Our goals consisted of both practical and theoretical sides.

For practical side, in Chapter 3, we have chosen to focus on *symmetric key broadcast encryption*, since it is one of only few highly-functional primitives that has been already used in real-world practice. As an evidence to support this, we have seen that the subset-difference broadcast encryption scheme by Naor *et al.* [NNL01] was recently chosen as a new standard called Advance Access Content System (AACS) and will be used in the next-generation DVD materials such as Blu-ray Disc (BD) and HD-DVD. Our aim regarding this area of research was to construct more efficient scheme in which main parameters, ciphertext sizes and private key sizes, are small and in particular "scalable", *i.e.,* being independent to the growth of the number of all users in the system. As a result, we achieved the first such schemes in the literature. Comparing to an independently proposed scheme of [BGW05] which also achieved the same goal, ours yield much less computational cost while have to pay off larger ciphertext size.

Our contributions regarding broadcast encryption do not confine only to some concrete schemes, indeed we have presented three generic frameworks based on different key derivation techniques. It turned out that almost all recent broadcast encryption schemes can be considered as instantiations from our frameworks.

For theoretical side, in Chapter 4, we presented a unified framework for *public key encryption with high functionalities*. Such kinds of primitives are extensions of normal public key encryption so as to strengthen the security or to achieve some useful functionalities which are specific to applications thereof. These include, but not limit to, existing primitives such as key-insulated encryption [DK02], forward-secure encryption [CHK03], certificate-based encryption [Gen03], intrusion-resilient encryption [DFK+03], hierarchical IBE (HIBE) [HL02, GS02], time-capsuled encryption [MHS03], and many more including some of their generalizations themselves.

The unified framework, called *directed acyclic graph encryption* (DAGE), presented a unified definition of algorithm syntax and a unified formalization of security notion for almost all public key encryption primitives with high functionalities known to date. In our framework, each encryption primitive is specified completely by the notion of *labeled access directed acyclic graph* (LAD), and its definitions for syntax and security notion will

be derived automatically from the unified ones. The merit of the framework is directed as it provides a convenient way to formalize a primitive without doing anew each time when one wants to propose a new kind of encryption.

A central tool to DAGE framework is the primitive implication theorem which provides a simple criterion stated in the propositional logic to check the relation between any pair of encryption primitives casted as DAGEs. As a result, this allows an automated verification of the relation proof.

We then presented a generic construction of DAGE for any graph from HIBE. This demonstrates a possibility result that we can base DAGE on any HIBE. Next we proposed efficient constructions based on bilinear maps for OR-graph and AND-graph DAGE that require only constant-size ciphertext. Combining these with multiple encryption technique gives the constructions of DAGE for general graph. We also gave the construction of OR-graph DAGE for the most complex graph (bounded-complete type). These constructions have merits not only in the theoretical point of view (where they show the possibility results) but also in the practical aspect: the protocol designer can simply specify a "tailor-made" graph for the on-purposed application and the implementation of the scheme will be prompted to use. Finally we gave prototypes for many functionalities; one consequence from this is that we can obtain DAGEs of any combinations of functionalities automatically.

Considering the combination of goals from two previous chapters, in Chapter 5, we presented public key broadcast encryption schemes that are *simultaneously* practical and featuring high functionalities. Although, broadcast encryption is already useful in the symmetric key setting as was focused in Chapter 3, we dealt with schemes in the public key setting in this chapter since they give more flexibility. To be able to attain such practical broadcast schemes, it is unavoidable to focus on more specific functionalities (not generic as in the second topic above). We focused on some most useful functionalities, namely forward security and keyword-based searchability. As a result, we achieved the most practical and scalable forward-secure public key broadcast encryption so far and the first searchable scheme broadcast encryption in the literature.

In conclusion, we have presented practical schemes and theoretical formalizations for encryption primitives with high functionalities. We believe that our techniques for broadcast encryption can be used to construct even more efficient schemes, and our unified framework for public key encryption with high functionalities can be utilized to define and construct even more useful new primitives. We also give a final remark for future research direction that our unified framework, although already covers almost all encryption primitives known to date, can be extended even more to include more primitives, such as recently proposed attribute-based encryption [GPSW06] in a meaningful way.

Bibliography

[ABC+05] 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 2005*, volume 3621 of LNCS, pages 205-222. Springer-Verlag, 2005.

[ACD+06] M. Abdalla, D. Catalano, A. W. Dent, J. Malone-Lee, G. Neven and N. P. Smart. Identity-based encryption gone wild. To appear in *Cryptographers Track — ICALP 2006*, 2006.

[AMN99] M. Abdalla, S. K. Miner, and C. Namprempre. Forward-secure threshold signature schemes. In *Topics in Cryptography — CT-RSA 2001*, volume 2020 of LNCS, pages 441-456. Springer- Verlag, 2001.

[AGKS05] M. Abe, R. Gennaro, K. Kurosawa, V. Shoup. Tag-KEM/DEM: A New Framework for Hybrid Encryption and A New Analysis of Kurosawa-Desmedt KEM. In *Advances in Cryptology — Eurocrypt 2005*, volume 3494 of LNCS, pages 128-146. Springer-Verlag, 2005.

[AT83] S. G. Akl, P. D. Taylor. Cryptographic Solution to a Problem of Access Control in a Hierarchy. In *ACM Transactions on Computer Systems*, Vol. 1, No. 3 (1983), pp. 239-248.

[AP03] S. Al-Riyami, K.G. Paterson. Certificateless Public Key Cryptography. In *Advances in Cryptology — Asiacrypt 2003*, volume 2894 of LNCS, pages 452-473. Springer-Verlag, 2003.

[ADR02] J.H. An, Y. Dodis, T. Rabin. On the Security of Joint Signature and Encryption. In *Advances in Cryptology — Eurocrypt 2002*, volume 2332 of LNCS, pages 83-107.

[And97] R. Anderson. Two remarks on public-key cryptology. Invited lecture, *4th ACM Conference on Computer and Communications Security*, 1997. Available at `http://www.cl.cam.ac.uk/ ftp/users/rja14/`.

[AMM99] J. Anzai, N. Matsuzaki and T. Matsumoto. Quick Group Key Distibution Scheme with Entity Revocation. In *Advances in Cryptology — Asiacrypt 1999*, volume 1716 of LNCS, pages 333-347, 1999.

[Asa02] T. Asano. A revocation scheme with minimal storage at receivers. In *Advances in Cryptology — Asiacrypt 2002*, volume 2501 of LNCS, pages 433-450. Springer-Verlag, 2002.

[AFI06] N. Attrapadung, J. Furukawa, and H. Imai. Forward-secure and searchable broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology — Asiacrypt 2006*, volume 4284 of LNCS, pages 161-177.

[AFIM06] N. Attrapadung, J. Furukawa, H. Imai, K. Matsuura. Searchable public-key broadcast encryption. In *Proc. of SITA 2006*.

[AHKI04] N. Attrapadung, G. Hanaoka, K. Kobara, H. Imai. ID-based Encryption for Directed Acyclic Graph Hierarchies and Application to Key-evolving Encryption Primitives. *Technical report of IEICE*, ISEC2004-77.

[AKI03a] N. Attrapadung, K. Kobara, and H. Imai. Sequential key derivation patterns for broadcast encryption and key predistribution schemes. In *Advances in Cryptology — Asiacrypt 2003*, volume 2894 of LNCS, pages 374-391. Springer-Verlag, 2003.

[AKI03b] N. Attrapadung, K. Kobara, and H. Imai. Broadcast encryption with short keys and transmissions. In *ACM Digital Rights Management Workshop — DRM 2003*, pages 55-66, 2003.

[AI05a] N. Attrapadung and H. Imai. Graph-decomposition-based frameworks for subset-cover broadcast encryption and efficient instantiations. In *Advances in Cryptology — Asiacrypt 2005*, volume 3788 of LNCS, pages 100-120. Springer-Verlag, 2005.

[AI05b] N. Attrapadung, H. Imai. Short Encrypted Broadcast with Short Keys. In *Proc. of SCIS 2005*, January 2005.

[AI05c] N. Attrapadung, H. Imai. Subset Incremental Chain Based Broadcast Encryption with Shorter Ciphertext. In *Proc. of SITA 2005*, November 2005.

[AI07] N. Attrapadung, H. Imai. Practical Broadcast Encryption from Graph-Theoretic Techniques and Subset-Incremental-Chain Structure. In *IEICE Transaction on Fundamental of Electronics, Communications and Computer Sciences — Special Section on Cryptography and Information Security*, Vol.E90-A No.1 pp.187-203, Jan. 2007.

[BP97] N. Bari, B. Pfitzmann. Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees. In *Proc. of Eurocrypt 1997*, volume 1233 of LNCS, pages 480-494.

[BC93] A. Beimel and B. Chor. Interaction in Key Distribution Schemes. In *Advances in Cryptology — Crypto 1993*, volume 773 of LNCS, pages 444-457.

[BDJR97] M. Bellare, A. Desai, E. Jokipii, P. Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *Proc. of FOCS 1997*, pages 394-403.

[BDPR98] M. Bellare, A. Desai, D. Pointcheval, P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes, In *Advances in Cryptology — Crypto 1998*, volume 1462 of LNCS, pages 26-45.

[BM99] M. Bellare and S. K. Miner. A forward-secure digital signature scheme. In *Advances in Cryptology — Crypto 1999*, volume 1666 of LNCS, pages 431-448. Springer-Verlag, 1999.

[BP02] M. Bellare, A. Palacio. Protecting against Key Exposure: Strongly Key-Insulated Encryption with Optimal Threshold. Available at `http://eprint.iacr.org/2002/064/`.

[BPR00] M. Bellare, D. Pointcheval, P. Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In *Advances in Cryptology — Eurocrypt 2000*, volume 1807 of LNCS, pages 139-155.

[BR93] M. Bellare, P. Rogaway. Random Oracles are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. ACM Conf. on Computer and Communications Security (CCS 1993)*, pages 62-73, 1993.

[BR94] M. Bellare, P. Rogaway. Optimal Asymmetric Encryption. In *Advances in Cryptology — Eurocrypt 1994*, volume 950 of LNCS, pages 92-111.

[B91] S. Berkovits. How To Broadcast A Secret. In *Advances in Cryptology — Eurocrypt 1991*, volume 547 of LNCS, pages 535-541.

[BSS05] I.F. Blake, G. Seroussi, N.P. Smart. Advances in Elliptic Curve Cryptography. London Mathematical Society Lecture Note Series 317, Cambridge University Press. (2005).

[B84] R. Blom. An Optimal Class of Symmetric Key Generation Systems. In *Advances in Cryptology — Eurocrypt 1984*, volume 209 of LNCS, pages 335-338.

[BSH+92] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro and M. Yung. Perfectly Secure Key Distribution for Dynamic Conferences. In *Advances in Cryptology — Crypto 1992*, volume 740 of LNCS, pages 471-486.

[BC94] C. Blundo and A. Cresti. Space Requirements for Broadcast Encryption. In *Advances in Cryptology — Eurocrypt 1994*, volume 950 of LNCS, pages 287-298.

[BMS96] C. Blundo, L. F. Mattos, D. R. Stinson. Trade-offs Between communication and Storage in Unconditionally Secure Schemes for Broadcast Encryption and Interactive Key Distribution. In *Advances in Cryptology — Crypto 1996*, volume 1109 of LNCS, pages 387-400.

[BF01] D. Boneh, M.K. Franklin. Identity-Based Encryption from the Weil Pairing. In *Advances in Cryptology — Crypto 2001*, volume 2139 of LNCS, pages 213-229.

[BF03] D. Boneh and M. Franklin. Identity Based Encryption from the Weil Pairing. *SIAM Journal of Computing*, 32(3), pp.586-615, 2003, full version of [BF01].

[BD94] J. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital signatures. In *Advances in Cryptology — EUROCRYPT 1993*, volume 765 of LNCS, pages 274-285.

[BMS96] C. Blundo, L. A. Frota Mattos, D. R. Stinson. Trade-offs between communication and storage in unconditionally secure schemes for broadcast encryption and interactive key distribution. In *Advances in Cryptology — Crypto 1996*, volume 1109 of LNCS, pages 387-400. Springer-Verlag, 1996.

[BB04a] D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *Advances in Cryptology — Eurocrypt 2004*, volume 3027 of LNCS, pages 223-238. Springer-Verlag, 2004.

[BB04b] D. Boneh, X. Boyen. Secure Identity Based Encryption Without Random Oracles. In *Advances in Cryptology —Crypto 2004*, volume 3152 of LNCS, pages 443-459. Springer-Verlag, 2004.

[BBG05] D. Boneh, X. Boyen, and E-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology — Eurocrypt 2005*, volume 3494 of LNCS, pages 440-456. Springer-Verlag, 2005. Full version available at `http://eprint.iacr.org/2005/015`.

[BBS04] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology — Crypto 2004*, volume 3152 of LNCS, pages 41-55. Springer-Verlag, 2004.

[BDOP04] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Advances in Cryptology — Eurocrypt 2004*, volume 3027 of LNCS, pages 506-522. Springer-Verlag, 2004.

[BGW05] D. Boneh, C. Gentry, and B. Waters. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In *Advances in Cryptology — Crypto 2005*, volume 3621 of LNCS, pages 258-275. Springer-Verlag, 2005.

[BF01] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology — Crypto 2001*, volume 2139 of LNCS, pages 213-229. Springer-Verlag, 2001.

[BK05] D. Boneh and J. Katz. Improved efficiency for CCA-secure cryptosystems built using identity based encryption. In *Proceedings of RSA-CT 2005*, volume 3376 of LNCS, pages 87-103. Springer-Verlag, 2005.

[BMW05] X. Boyen, Q. Mei, and B. Waters. Direct Chosen Ciphertext Security from Identity-Based Techniques. In *Proc. ACM Conf. on Computer and Communications Security (CCS 2005)*, pages 320-329, ACM Press, 2005.

[BW06] X. Boyen and B. Waters. Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). To appear in *Advances in Cryptology — Crypto 2006*, 2006.

[BS03] D. Boneh and A. Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71-90, 2003.

[Can01] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *Proc. of FOCS 2001* pages 136-145.

[CGI+99] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. In *Proc. IEEE INFOCOM '99*, volume 2, pages 708.716, New York, NY, March 1999. IEEE.

[CGH98] R. Canetti, O. Goldreich, S. Halevi. The Random Oracle Methodology, Revisited (Preliminary Version). In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing — STOC 1998*, pages 209-218.

[CHK03] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In *Advances in Cryptology — Eurocrypt 2003*, volume 2656 of LNCS, pages 255-271. Springer-Verlag, 2003.

[CHK04] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *Advances in Cryptology — Eurocrypt 2004*, volume 3027 of LNCS, pages 207-222. Springer-Verlag, 2004.

[CHK05] R. Canetti, S. Halevi, and J. Katz. Adaptively-Secure, Non-interactive Public-Key Encryption. In *Theory of Cryptography Conference — TCC 2005*, volume 3378 of LNCS, pages 150-168. Springer-Verlag, 2005.

[CKN03] R. Canetti, H. Krawczyk, J.B. Nielsen. Relaxing Chosen-Ciphertext Security. In *Advances in Cryptology — Crypto 2003*, volume 2729 of LNCS, pages 565-582.

[CMN99] R. Canetti, T. Malkin, and K. Nissim. Efficient communication-storage tradeoffs for multicast encryption. In *Advances in Cryptology — Eurocrypt 1999*, volume 1592 of LNCS, pages 459-474, Springer-Verlag, 1999.

[Che06] J. H. Cheon. Security analysis of the strong Diffie-Hellman problem. In *Advances in Cryptology — Eurocrypt 2006*, volume 4004 of LNCS, pages 1-11. Springer, 2006.

[CT89] G. C. Chick and S. E. Tavares. Flexible Access Control with Master Keys. In *Advances in Cryptology — CRYPTO 1989*, LNCS 435, pp. 316-322.

[CS98] R. Cramer, V. Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In *Advances in Cryptology — CRYPTO 1998*, volume 1492 of LNCS, pages 13-25. Springer, 1998.

[CS02] R. Cramer, V. Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In *Advances in Cryptology — Eurocrypt 2002*, volume 2332 of LNCS, pages 45-64. Springer, 2002.

[CS03] R. Cramer, V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal of Computing* 33, pages 167-226, 2003.

[DV98] Y. Desmedt, V. Viswanathan. Unconditionally Secure Dynamic Conference Key Distribution. In *Proc. of IEEE, ISIT 1998*.

[D00] R. Diestel. Graph theory. 2nd ed., Graduate texts in mathematics 173, (2000).

[DH76] W. Diffie, M.E. Hellman. New Directions in Cryptography. IEEE Trans. on Info. Theory, IT-22, pages 644-654, 1976.

[DOW92] W. Diffie, P. van Oorschot, and W. Wiener. Authentication and authenticated key exchanges. In *Designs, Codes and Cryptography*, volume 2, pages 107-125, 1992.

[DK05] Y. Dodis, J. Katz. Chosen-Ciphertext Security of Multiple Encryption. In *Theory of Cryptography Conference — TCC 2005*, volume 3378 of LNCS, pages 188-209. Springer-Verlag, 2005.

[DK02] Y. Dodis, J. Katz, S. Xu, M. Yun. Key-Insulated Public Key Cryptosystems. In *Advances in Cryptology — EUROCRYPT 2002*, volume 2332 of LNCS, pages 65-82. Springer-Verlag, 2002.

[DKXY03] Y. Dodis, J. Katz, S. Xu, M. Yung. Strong Key-Insulated Signature Schemes. In *Public Key Cryptography — PKC 2003*, volume 2567 of LNCS, pages 130-144. Springer-Verlag, 2003.

[DFK+03] Y. Dodis, M. K. Franklin, J. Katz, A. Miyaji, M. Yung. Intrusion-Resilient Public-Key Encryption. In In *Proceedings of CT-RSA 2003*, pages 19-32.

[DFK+04] Y. Dodis, M. K. Franklin, J. Katz, A. Miyaji, M. Yung. A Generic Construction for Intrusion-Resilient Public-Key Encryption. In *Proceedings of CT-RSA 2004*, pages 81-98.

[DF02] Y. Dodis and N. Fazio. Public-key broadcast encryption for stateless receivers. In *ACM Digital Rights Management — DRM 2002*, volume 2696 of LNCS, pages 61-80. Springer, 2002.

[DF03] Y. Dodis and N. Fazio. Public-key trace and revoke scheme secure against adaptive chosen ciphertext attack. In *Public Key Cryptography — PKC 2003*, volume 2567 of LNCS, pages 100-115. Springer-Verlag, 2003.

[DDN91] D. Dolev, C. Dwork, and M. Naor. Nonmalleable Cryptography. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing — STOC 1991*, ACM, 1991, pages 391-437, 1991.

[DDN00] D. Dolev, C. Dwork, and M. Naor. Nonmalleable Cryptography. In *SIAM Journal of Computing 30(2)*, pages 391-437. (2000).

[ElG85] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* 31(4), pages 469-472, 1985.

[FN93] A. Fiat and M. Naor. Broadcast encryption. In *Advances in Cryptology — Crypto 1993*, volume 773 of LNCS, pages 480-491. Springer-Verlag, 1993.

[FO99] E. Fujisaki, T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *Advances in Cryptology — Crypto 1999*, volume 1666 of LNCS, pages 537-554. Springer-Verlag, 1999.

[FOPS01] E. Fujisaki, T. Okamoto, D. Pointcheval, J. Stern. RSA-OAEP Is Secure under the RSA Assumption. In *Advances in Cryptology — Crypto 2001*, volume 2139 of LNCS, pages 260-274. Springer-Verlag, 2001.

[GS02] C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In *Advances in Cryptology — Asiacrypt 2002*, volume 2501 of LNCS, pages 548-566. Springer-Verlag, 2002.

[GR04] C. Gentry and Z. Ramzan. RSA Accumulator Based Broadcast Encryption. In *Proc. of Information Security Conference — ISC 2004*, LNCS 3225, pages 73-86. Springer-Verlag, 2004.

[Gen03] C. Gentry. Certificate-Based Encryption and the Certificate Revocation Problem. In *Advances in Cryptology — Eurocrypt 2003*, volume 2656 of LNCS, pages 272-293. Springer-Verlag, 2003.

[Goh03] E. Goh. Secure indexes. Available at `http://eprint.iacr.org/2003/216/`.

[Gol99] O. Goldreich. Modern Cryptography, Probabilistic Proofs and Pseudo-randomness. Springer-Verlag, Algorithms and Combinatorics, Vol 17, 1999.

[Gol01] O. Goldreich. Foundation of Cryptography (Basic Tools). Cambridge University Press, 2001.

[GMW87] O. Goldreich, S. Micali, A. Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing — STOC 1987*, ACM, 1987, pages 218-229.

[GM84] S. Goldwasser, S. Micali. Probabilistic Encryption. In *J. Comput. Syst. Sci.*, 28(2), pages 270-299 (1984).

[GMR85] S. Goldwasser, S. Micali, C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing — STOC 1985*, ACM, 1985.

[GMR89] S. Goldwasser, S. Micali, C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *SIAM Journal of Computing*, volume 18(1), pages 186-208. (1989).

[GMR88] S. Goldwasser, S. Micali, R. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. In *SIAM Journal on Computing 17(2)*, pages 281-308. (1988).

[GSW04] P. Golle, J. Staddon and B. Waters. Secure conjunctive keyword search over encrypted data. In *Proc. of Applied Cryptography and Network Security — ACNS 2004*, volume 3089 of LNCS, page 31-45. Springer-Verlag, 2004.

[GST04] M. T. Goodrich, J. Z. Sun, and R. Tamassia. Efficient tree-based revocation in groups of low-state devices. In *Advances in Cryptology — Crypto 2004*, volume 3152 of LNCS, pages 511-527. Springer-Verlag, 2004.

[GPSW06] V. Goyal, O. Pandey, A. Sahai, B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proc. ACM Conf. on Computer and Communications Security (CCS 2006)*, pages 89-98, 2006.

[Gün89] C. Günther. An identity-based key exchange protocol. In *Advances in Cryptology — Eurocrypt 1989*, volume 434 of LNCS, pages 29-37. Springer-Verlag, 1989.

[HS02] D. Halevy and A. Shamir. The LSD broadcast encryption scheme. In *Advances in Cryptology — Crypto 2002*, volume 2442 of LNCS, pages 47-60. Springer-Verlag, 2002.

[HHSI05] Y. Hanaoka, G. Hanaoka, J. Shikata, H. Imai. Identity-Based Hierarchical Strongly Key-Insulated Encryption and Its Application. In *Advances in Cryptology — Asiacrypt 2005*, volume 3788 of LNCS, pages 495-514.

[HHI06] Y. Hanaoka, G. Hanaoka, H. Imai. Parallel Key-Insulated Encryption. In *Public Key Cryptography — PKC 2006*, volume 3958 of LNCS, pages 105-122.

[HL02] J. Horwitz and B. Lynn. Toward hierarchical identity-based encryption. In *Advances in Cryptology — Eurocrypt 2002*, volume 2332 of LNCS, pages 466-481. Springer-Verlag, 2002.

[HLL05] J. Y. Hwang, D. H. Lee, and J. Lim. Generic transformation for scalable broadcast encryption Schemes. In *Advances in Cryptology — Crypto 2005*, volume 3621 of LNCS, pages 276-292. Springer-Verlag, 2005.

[HL06] Y. H. Hwang, P. J. Lee. Efficient Broadcast Encryption Scheme with Log-Key Storage. In *Proc. of Financial Cryptography 2006*, March 2006.

[ILL89] R. Impagliazzo, L.A. Levin, M. Luby. Pseudo-random Generation from one-way functions. In *Proc. of STOC 1989*, pages 12-24.

[JHC+05] N-S Jho, J. Y. Hwang, J. H. Cheon, M-H. Kim, D. H. Lee, E. S. Yoo. One-way chain based broadcast encryption schemes. In *Advances in Cryptology — Eurocrypt 2005*, volume 3494 of LNCS, pages 559-574. Springer-Verlag, 2005.

[Jou00] A. Joux. A one round protocol for tripartite Diffie-Hellman. In *Proc. of Algorithmic Number Theory Symposium IV*, volume 1838 of LNCS, pages 385-394. Springer-Verlag, 2000.

[Kat02] J. Katz. A Forward-Secure Public-Key Encryption Scheme. Available at `http://eprint.iacr.org/2002/060`.

[KRS99] R. Kumar, S. Rajagopalan, and A. Sahai. Coding constructions for blacklisting problems without computational assumptions. In *Advances in Cryptology — Crypto 1999*, volume 1666 of LNCS, pages 609-623. Springer-Verlag, 1999.

[KR03] R. Kumar, A. Russell. A Note on the Set Systems used for Broadcast Encryption. In *Proc. of SODA 2003*.

[KD04] K. Kurosawa, Y. Desmedt. A New Paradigm of Hybrid Encryption Scheme. In *Advances in Cryptology — Crypto 2004*, volume 3152 of LNCS, pages 426-442. Springer-Verlag, 2004.

[KYDB98] K. Kurosawa, T. Yoshida, Y. Desmedt, M. Burmester. Some Bounds and a Construction for Secure Broadcast Encryption. In *Advances in Cryptology — Asiacrypt 1998*, volume 1514 of LNCS, pages 420-433.

[Lam79] L. Lamport. Constructing Digital Signatures from a One-Way Function. Technical Report CSL-98, SRI International, Palo Alto, 1979.

[LS98] M. Luby and J. Staddon. Combinatorial bounds for broadcast encryption. In *Advances in Cryptology — Eurocrypt 1998*, volume 1403 of LNCS, pages 512-526. Springer-Verlag, 1998.

[MMM02] T. Malkin, D. Micciancio, and S. K. Miner. Efficient generic forward-secure signatures with an unbounded number of time periods. In *Advances in Cryptology — Eurocrypt 2002*, volume 2332 of LNCS, pages 400-417. Springer-Verlag, 2002.

[MI87] T. Matsumoto, H. Imai. On the Key Predistribution System: A Practical Solution to the Key Distribution Problem. In *Advances in Cryptology — Crypto 1987*, pages 185-193.

[MS98] D. McGrew, A. T. Sherman. Key Establishment in Large Dynamic Groups Using One-Way Function Trees. In *IEEE Trans. Software Eng*, 29(5) pages 444-458, (2003).

[Mer89] R.C. Merkle. A Certified Digital Signature. In *Advances in Cryptology — Crypto 1989*, volume 435 of LNCS, pages 218-238, Springer-Verlag, 1990.

[MH78] R.C. Merkle, M.E. Hellman. Hiding Information and Signatures in Trapdoor Knapsacks. IEEE Trans. Inform. Theory. Vol.24, pages 525-530, 1978.

[M03] M.J. Mihaljevic. Key Management Schemes for Stateless Receivers Based on Time Varying Heterogeneous Logical Key Hierarchy. In *Advances in Cryptology — Asiacrypt 2003*, volume 2894 of LNCS, pages 137-154. Springer-Verlag, 2003.

[MHS03] M.C. Mont, K. Harrison, M. Sadler. The HP time vault service: exploiting IBE for timed release of confidential information. In *ACM-WWW 2003*, pages 160-169.

[NNL01] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology — Crypto 2001*, volume 2139 of LNCS, pages 41-62. Springer-Verlag, 2001.

[NP00] M. Naor and B. Pinkas. Efficient trace and revoke schemes. In *Financial Cryptography — FC 2000*, volume 1962 of LNCS, pages 1-20. Springer-Verlag, 2000.

[NY89] M. Naor, Yung. Universal One-Way Hash Functions and Their Cryptographic Applications. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing — STOC 1989*, ACM, 1989, pages 33-43.

[NY90] M. Naor, M. Yung. Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In *Proc. of STOC 1990*, pages 427-437.

[Rab79] M.O. Rabin. Digitalized Signatures and Public Key Functions as Intractable as Factoring. MIT/LCS/TR-212, 1979.

[RS91] C. Rackoff, D.R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *Advances in Cryptology — Crypto 1991*, volume 576 of LNCS, pages 433-444.

[RSA78] R. Rivest, A. Shamir, L. M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. In *Commun. ACM 21(2)*, pages 120-126. (1978)

[Rom90] J. Rompel. One-Way Functions are Necessary and Sufficient for Secure Signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing — STOC 1990*, ACM, 1990, pages 387-394.

[SW05] A. Sahai, B. Waters. Fuzzy Identity-Based Encryption. In *Advances in Cryptology — Eurocrypt 2005*, volume 3494 of LNCS, pages 457-473.

[Sha79] A. Shamir. How to share a secret. In *Commun. ACM* 22(11), pages. 612-613. (1979)

[Sha84] A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In *Advances in Cryptology — CRYPTO 1984*, pages 47-53.

[Sha49] C.E. Shannon. Communication Theory of Secrecy Systems. *Bell Sys. Tech. Jour.*, Vol.28, pages 656-715, 1949.

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology — Eurocrypt 1997*, volume 1233 of LNCS, pages 256-266. Springer-Verlag, 1997.

[Sho01] V. Shoup. A proposal for an ISO standard for public key encryption (version 2.1).

[Sho04] V. Shoup. Sequences of Games: A Tool for Taming Complexity in Security Proofs. IACR ePrint Report 2004/332.

[S76] Z. Star. An Asymptotic Formula in the Theory of Composition. In *Aequationes Math*, Vol. 12, No. 1, page 113, 1976.

[Sti97] D. R. Stinson. On some methods for unconditionally secure key distribution and broadcast encryption. In *Designs, Codes and Cryptography*, 12(3):215-243, 1997.

[Sti96] D.R.Stinson. On Some Methods for Unconditionally Secure Key Distribution and Broadcast Encryption. In *Designs, Codes and Cryptography 12* (1997), pages 215-243.

[SW98] D. Stinson, R. Wei. Some New Results on Key Distribution Patterns and Broadcast Encryption. In *Designs, Codes and Cryptography 14* (1998), pages 261-279.

[SWP00] D. X. Song, D. Wagner and A. Perrig. Practical techniques for searches on encrypted data. In *Proc. of IEEE Symposium on Security and Privacy*, page 44-55, 2000.

[WHA97] D.M. Wallner, E.J. Harder, and R.C. Agee. Key management for multicast: Issues and architectures. IETF draft wallner-key, 1997.

[WNR04] P. Wang, P. Ning, and D.S. Reeves. Storage-Efficient Stateless Group Key Revocation. In *Proc. of Information Security Conference — ISC 2004*, volume 3225 of LNCS, pages 25-38. Springer-Verlag, 2004.

[Wat05] B. Waters. Efficient Identity-Based Encryption Without Random Oracles. In *Advances in Cryptology — Eurocrypt 2005*, volume 3494 of LNCS, pages 114-127. Springer-Verlag, 2005.

[WBDS04] B. Waters, D. Balfanz, G. Durfee and D. Smetters. Buliding an encrypted and searchable audit log. In *Proc. of Network and Distributed System Security Symposium — NDSS 2004*, page 205-214, 2004.

[WGL98] C. Wong, M. Gouda, and S. Lam. Secure group communications using key graphs. In *Proceedings of the ACM SIGCOMM '98*, pages 68-79. ACM Press, 1998.

[YFDL04] D. Yao, N. Fazio, Y. Dodis, and A. Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *Proc. ACM Conf. on Computer and Communications Security (CCS 2004)*, pages 354-363, ACM, 2004.

[ZHSI04] R. Zhang, G. Hanaoka, J. Shikata, H. Imai. On the Security of Multiple Encryption or CCA-security+CCA-security=CCA-security?. In *Public Key Cryptography — PKC 2004*, volume 2947 of LNCS, pages 360-374.

Appendix A

List of Publications

Selected Journal and International Conference Papers

[1] N. Attrapadung, K. Kobara, and H. Imai. Broadcast encryption with short keys and transmissions. In *ACM Digital Rights Management Workshop — DRM 2003*, pages 55-66, 2003.

[2] N. Attrapadung, K. Kobara, and H. Imai. Sequential key derivation patterns for broadcast encryption and key predistribution schemes. In *Advances in Cryptology — Asiacrypt 2003*, volume 2894 of LNCS, pages 374-391. Springer-Verlag, 2003.

[3] N. Attrapadung and H. Imai. Graph-decomposition-based frameworks for subset-cover broadcast encryption and efficient instantiations. In *Advances in Cryptology — Asiacrypt 2005*, volume 3788 of LNCS, pages 100-120. Springer-Verlag, 2005.

[4] N. Attrapadung, J. Furukawa, and H. Imai. Forward-secure and searchable broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology — Asiacrypt 2006*, volume 4284 of LNCS, pages 161-177. Springer-Verlag, 2006.

[5] N. Attrapadung, J. Furukawa, T. Gomi, G. Hanaoka, H. Imai, R. Zhang. Efficient Identity-Based Encryption with Tight Security Reduction. In *Cryptology and Network Security — CANS 2006*, volume 4301 of LNCS, pages 19-36. Springer-Verlag, 2006.

[6] N. Attrapadung, H. Imai. Practical Broadcast Encryption from Graph-Theoretic Techniques and Subset-Incremental-Chain Structure. In *IEICE Transaction on Fundamental of Electronics, Communications and Computer Sciences — Special Section on Cryptography and Information Security*, Vol.E90-A No.1 pp.187-203, Jan. 2007.

Other Reviewed International Conference Papers

[1] N. Attrapadung, K. Kobara, H. Imai. Refining SD and LSD Broadcast Encryption Schemes. *International Symposium on Information Theory and its Applications — ISITA 2004*, Italy, October 2004.

[2] N. Attrapadung, K. Kobara, H. Imai. Broadcast Encryption Schemes Designed for Low-Bandwidth Wireless Communication. *Wireless Personal Media Communications — WPMC 2005*, Denmark, September 2005.

[3] N. Attrapadung, Y. Cui, D. Galindo, G. Hanaoka, I. Hasuo, H. Imai, K. Matsuura, P. Yang, R. Zhang. Relations Among Notions of Security for Identity Based Encryption Schemes. Extended abstract version in *Latin American Theoretical Informatics — LATIN 2006*, volume 3887 of LNCS, pages 130-141, Springer-Verlag, March 2006. Full version in *Journal of Information Processing Society of Japan (IPSJ Journal)*, Vol.47, No. 8, June 2006.

[4] N. Attrapadung, K. Kobara, H. Imai. Subset Incremental Chain Based Broadcast Encryption with Shorter Ciphertext. *International Symposium on Information Theory and its Applications — ISITA 2006*, South Korea, October 2006.

Invited Papers

[1] N. Attrapadung, H. Imai. A Survey on Recent Advances in Broadcast Encryption. Invited Tutorial at the *Annual IEICE General Conference 2006*, Tokyo, March 2006 .

Non-Reviewed Domestic Conference Papers

[1] N. Attrapadung, K. Kobara, H. Imai. Key Predistribution Scheme based on Pseudo-Random Generator secure Against Any Size of Collusion. In *Proc. of SITA*[1] *2002*, Gunma, December 2002.

[2] N. Attrapadung, K. Kobara, H. Imai. Broadcast Encryption with One Storage Key at Each Receiver in One Transmission Message. In *Proc. of SCIS*[2] *2003*, Shizuoka, January 2003.

[3] N. Attrapadung, K. Kobara, H. Imai. Optimally Mastering Keys in Various Broadcast Encryption Schemes. In *Proc. of SITA 2003*, Hyogo, December 2003.

[4] N. Attrapadung, K. Kobara, H. Imai. Efficient Broadcast Encryption from Trapdoor One-way Accumulators. In *Proc. of SCIS 2004*, Sendai, January 2004.

[5] N. Attrapadung, G. Hanaoka, K. Kobara, H. Imai. ID-based Encryption for Directed Acyclic Graph Hierarchies and Application to Key-evolving Encryption Primitives. *Technical report of IEICE*, ISEC2004-77, September 2004.

[6] N. Attrapadung, G. Hanaoka, K. Kobara, H. Imai. ID-based Encryption for Directed Acyclic Graph Hierarchies: Unification of Key-evolving Encryption Primitives. In *Proc. of SITA 2004*, Gifu, December 2004.

[7] N. Attrapadung, K. Kobara, H. Imai. Short Encrypted Broadcast with Short Key. In *Proc. of SCIS 2005*, Kobe, January 2005.

[8] N. Attrapadung, K. Kobara, H. Imai. Subset Incremental Chain Based Broadcast Encryption with Shorter Ciphertext (Extended Abstract). In *Proc. of SITA 2005*, Okinawa, November 2005. ($\star$ Received the best paper award of SITA 2005).

[9] N. Attrapadung, G. Hanaoka, H. Imai. Directed Acyclic Graph Encryption (Extended Abstract). In *Proc. of SCIS 2006*, Hiroshima, January 2006. ($\star$ Received the best paper award of SCIS 2006).

[1] SITA = Symposium on Information Theory and its Applications
[2] SCIS = Symposium on Cryptography and Information Security

[10] N. Attrapadung, J. Furukawa, H. Imai, K. Matsuura. Searchable public-key broadcast encryption. In *Proc. of SITA 2006*, Hokkaido, November 2006.

[11] N. Attrapadung, J. Furukawa, H. Imai, K. Matsuura. Forward-secure broadcast encryption with short ciphertexts and private keys. In *Proc. of SCIS 2007*, Nagasaki, January 2007.