uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Deholo Nali

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Ph.D. (Computer Science)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Identity-based Cryptographic Schemes for Large User Communities

TITRE DE LA THÈSE / TITLE OF THESIS

Ali Miri

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

Carlisle Adams

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Liam Peyton

Paul van Oorschot

Amr Youssef

Robert Zuccherato

Gary W. Slater

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES /
DEAN OF THE FACULTY OF GRADUATE AND POSTDOCORAL STUDIES

IDENTITY-BASED CRYPTOGRAPHIC SCHEMES FOR LARGE USER COMMUNITIES

By
Deholo Nali

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
AT
UNIVERSITY OF OTTAWA
OTTAWA, ONTARIO

Your file Votre référence
ISBN: 0-494-11001-5
Our file Notre référence
ISBN: 0-494-11001-5

To my father,
Dr. M. N. Nali.

Table of Contents

List of Tables

List of Figures

List of Abreviations

Abreviation	Definition
AM	Access Manager
AOS	Append-Only Signature
AS	Autonomous System
BBAC	Biometric-based Access Control
BGP	Border Gateway Protocol
CBPKC	Certificate-based Public-Key Cryptography
CLPKC	Certificateless Public-Key Cryptography
EMA	Encrypted Message Authentication
FIBE	Fuzzy Identity-base Encryption
FS	Forward Security
fsHIBE	Forward-Secure Hierarchical Identity-based Encryption
fsPKE	Forward-Secure Public-Key Encryption
fsPKS	Forward-Secure Public-Key Signature
GS	Group Signature
HAA	Hierarchical Anonymous Authentication
HGS	Hierarchical Group Signature
HIBE	Hierarchical Identity-based Encryption
HIBGS	Hierarchical Identity-based Group Signature
HIBS	Hierarchical Identity-based Signature
HIBSC	Hierarchical Identity-based Signcryption
HIBKG	Hierarchical Identity-based Key Generation
HTIR	Hierarchical Time-based Information Release
IBE	Identity-based Encryption
IBPKC	Identity-based Public-Key Cryptography
IBS	Identity-based Signature
IBSC	Identity-based Signcryption
IBGS	Identity-based Group Signature
ID	Identity

Abreviation	Definition
KRA	Key Release Agent
PCA	Public Ciphertext Authenticity
PKC	Public-Key Cryptography
PKG	Private-Key Generator
PKI	Public-Key Infrastructure
PSI	Public Subgroup Identification
RBAC	Role-based Access Control
rPKG	root Private-Key Generator
RL	Revocation List
rMan	root Manager
SCPKC	Self-certified Public-Key Cryptography
SEM	Security Mediator
SGM	Subgroup Manager
TA	Trusted Authority
TIR	Time-based Information Release
thABE	Threshold Atribute-based Encryption
TPS	Trusted Privilege Server
TTP	Trusted Third Party
VLR	Verifier Local Verification

Abstract

Most currently deployed public-key cryptosystems are certificate-based (CBPK) cryptosystems. In large and distributed user communities, CBPK cryptosystems typically require encryptors to obtain the public-key certificate of each intended recipient, and signature verifiers to obtain the public-key certificate of each claimed signer. However, the management and distribution of public-key certificates are cumbersome in these communities. To address this issue, identity-based (ID-based) cryptosystems have been suggested as a possible alternative, because they enable the derivation of public keys from arbitrary string identifiers (such as email addresses, IP addresses, or privilege identifiers).

This dissertation investigates the use of ID-based cryptography for large user communities. In particular, we present efficient and provably-secure (hierarchical) ID-based cryptographic schemes to support the following applications: (1) generation of decryption keys from non-repeatable (e.g. biometric) information; (2) authentication of the sender of encrypted information; (3) verification that a user belongs to a publicly known subset of a group, while the user remains anonymous to all but a designated party.

Acknowledgements

It is with deep gratitude that I submit this dissertation.

I started this project thinking that I could make important contributions to the field of cryptography. In retrospective, I have learned that cryptography is just one aspect of computer security, and that my contributions are significantly less impressive than the advances made by my mentors.

My deep thanks to A. Miri, C. Adams, R. Zuccherato, and P. van Oorschot. A. Miri found research funds for me, guided me through critical decisions, and helped me achieve my potential. C. Adams inspired me greatly, helped me write meaningfully, and provided funds that enabled me to attend international conferences. R. Zuccherato's paper on hyperelliptic curves and their applications to cryptography remains the most influential paper of my academic career, because it convinced me to choose cryptography as a research topic (instead of pure algebraic number theory). P. van Oorschot opened my eyes to the world of computer security, and taught me to think critically, in practical terms.

My dearest thanks to Madeleine, my sweet wife, who shared my joys and tears in the difficult times. Thanks to Behzad, Maggie, and the other team mates of the Computational Lab in Coding and Cryptography, at the University of Ottawa.

Humbly and honestly, I confess that the encouragements and inspiration of my mentors and friends did not suffice, in the difficult and loneliest times. In Christ Jesus, I have found: a friend to talk to; a model for integrity; inspiration to use my talents in order help others achieve their potential; and the way to be reconciled to God. While the Ph.D. programme was sometimes very difficult for me, I have been comforted by the knowledge that God shapes whom He loves, and that, while He sometimes bends me, He never breaks me.

I take full responsibility for any mistakes contained in this dissertation.

Ottawa, Ontario
October 25, 2005

Deholo Nali

Chapter 1

Introduction

This dissertation explores the following theme:

Efficient Identity-Based Cryptography (IBC) for Large User Communities.

More specifically, the following three fundamental questions will be examined:

- *Is it possible to generate efficiently-usable private keys from non-repeatable (e.g. biometric) information ?*
- *Is it possible to authenticate the sender of encrypted information ?*
- *Is it possible to determine whether a user belongs to a publicly known subset of a group, while preserving the anonymity of this user from all but a designated party ?*

In this introductory chapter, the relevance of these questions is briefly explained, and the dissertation's main contributions are outlined.

1.1 Dissertation's Relevance

Broadly defined, *information security* research aims at building systems which authenticate users, preserve the verifiability of these users' commitments, preserve the *availability* and *integrity* of sensitive information, and guarantee the confidentiality of such information. This dissertation is concerned with the design of mechanisms which support some of the aforementioned services, in large user communities (e.g. sets of more than 10,000 users). After defining fundamental (security related) terms, we present a practical motivation for addressing the dissertation's three fundamental questions, summarize the shortcomings of existing technical solutions, and outline our achievements.

1.1.1 Fundamental Terminology

Cryptography is the subset of information security which deals with the design and analysis of mathematical transformations whose goal is to preserve the confidentiality (i.e. secrecy), integrity (i.e. unaltered state), and authenticity (i.e. validity of origin) of sensitive information. Many types of *cryptographic schemes* (i.e. mechanisms composed of sets of algorithms) exist, including the most popular *encryption* and *signature* schemes. Encryption schemes transform *plaintext* (i.e. bit sequences representing plain information) into *ciphertext* (i.e. bit sequences representing information that is unintelligible to a third party), in order to provide information confidentiality. Associated with each encryption scheme is a decryption scheme which performs the inverse operation of the encryption scheme. Signature schemes produce cryptographic digital signatures (i.e. bit sequences which attest to the authenticity and integrity of the data that has been signed). Associated with each signature scheme is a verification scheme whose task is to determine the validity of a given cryptographic digital signature. In order to accomplish their intended purposes, encryption/decryption and signature/verification schemes respectively use encryption/decryption and signature/verification *keys*. Each such *key* is a bit string.

Cryptographic schemes can be combined to make up *cryptosystems*, which thereby benefit from the aggregated features of their contained cryptographic schemes. One can distinguish two major types of cryptosystems. *Symmetric* (also known as *secret-key*) cryptosystems use a common (secret) key to perform encryption and decryption procedures, while *asymmetric* (also known as *public-key*) cryptosystems use two distinct types of keys (namely *private* and *public* keys) to perform these two types of procedures. For a public-key cryptosystem to be secure, it must be computationally infeasible to derive a private key from any combination of the corresponding public key and publicly disclosed parameters of the cryptosystem. This is not however a sufficient condition for security.

1.1.2 Practical Motivation

Since the topic of efficient cryptography for large and distributed user communities is large, this dissertation focuses on three specific problems whose relevance is discussed below. The first problem is more related to efficient cryptography than cryptography in large user communities. However, the solution provided for this problem is scalable to large user communities. The second and third problems both deal with efficient cryptography in large user communities.

Generation of Efficiently-Usable Keys From Non-Repeatable Information

One use of decryption keys generated from non-repeatable information can be found in the context of biometric-based access control. Indeed, biometric readings are non-repeatable (two readings of the same entity are typically different [Dau04]).

Moreover, access to restricted resources or services is typically controlled by three means: what people know, what people have, and what people are. Though cost-effective, human-memorized secrets (i.e. what people know) are often easy to guess [FK89, Kle90, Wu99]. Hence, secret information stored on transportable tokens (i.e. what people have) is often used as a basis for access control. However, transportable tokens are prone to theft, loss, and damage. Consequently, the use of human-biometric identity (i.e. what people are) has been suggested as means to enforce access control (see, e.g., `www.biometrics.org`). Suppose, for instance, that a user $\mathcal{A}$ goes to a hospital $\mathcal{H}$ and identifies herself via an iris scan, *under the ongoing surveillance of a trained agent.* $\mathcal{H}$ could use this iris scan to encrypt $\mathcal{A}$'s medical information when the information needs to be securely sent to $\mathcal{A}$ (e.g. via the Web, or via storage of ciphertexts on a smart card presented by (a representative of) $\mathcal{A}$). In order to obtain her biometric decryption key, $\mathcal{A}$ could be required to go in person to a trusted third party (e.g. a state agency) which would deliver biometric decryption keys via the same identification procedure as that used by $\mathcal{H}$. $\mathcal{A}$ could then decrypt ciphertexts addressed to her, using a decryption scheme suitable (i.e. specifically designed) for the above biometric-based access control process.

It is important to note that, when biometrics are read under the surveillance of trained agents, the odds of user impersonation in the process of obtaining biometric-based decryption keys is significantly reduced. Moreover, the aforementioned identification method is simple[1], does not require prior relationship with the authenticating party, and does not require corroborating credentials (which may have been illegally forged). Furthermore, each user decides to whom her biometric identity is revealed, and the use of biometrics to generate cryptographic keys provides an efficient method to assign unique identifiers to human users. Hence, the ability to generate cryptographic keys from non-repeatable information is useful for biometric-based access control.

Another use of private keys based on non-repeatable information arises in the context of biometric-based authentication. Using the above notation, suppose that $\mathcal{A}$ cryptographically signs a document using her biometric-based private key, and that $\mathcal{H}$ wants to verify that $\mathcal{A}$ is really the signer of this document. Then, using a copy of $\mathcal{A}$'s last biometric reading, $\mathcal{H}$ can verify whether the given signature is associated

[1] It consists of a biometric reading only.

with $\mathcal{A}^2$.

However, the generation of cryptographic keys obtained from non-repeatable information is non-trivial. Moreover, even if one constructed a method to generate cryptographic keys from non-repeatable information, these keys should be *efficiently-usable.* In other words, decryption and signature schemes using keys generated from non-repeatable information should be computationally and space efficient. This, as shall see in Chapter 3, is not a trivial task either.

Encrypted Message Authentication

The second problem which this dissertation deals with is *encrypted message authentication* (EMA).

EMA consists of authenticating the issuer of encrypted information. Suppose that encrypted messages are sent via untrusted networks. If these networks are unable to distinguish information encrypted by trusted users from both random data and information encrypted by untrusted users, then malicious attackers can clog these networks (e.g. by overflowing them with random data), and thereby deny service to the associated users. An instance of this problem arises when encrypted electronic mail (email) is sent via the Internet. In order to preserve the confidentiality of email from any sender to her email recipients, large corporations offering email services must find a way to allow email routers to block encrypted undesired email (spam). The same issue arises in the case of encrypted IP packet routing, since the ability of highly-used public network routers to recognize and block untrusted encrypted packets can significantly reduce the threats of denial of service attacks. The problem also appears when encrypted data is made available on public servers. If these servers are unable to identify untrusted data or encrypted random data, then significant delays can be experienced in the retrieval of trusted information. Therefore, efficient cryptographic schemes supporting encrypted message authentication are useful.

Anonymous Authentication

The third issue which this dissertation deals with is *anonymous authentication.* More precisely, suppose that a large user group G is composed of publicly-known non-overlapping subgroups organized in a tree-shaped hierarchy. Suppose furthermore that the members of each subgroup wish to remain anonymous to all but a designated member of their subgroup (henceforth called the *subgroup leader*). Then the third challenge tackled by this dissertation is to design a mechanism which enables such

[2]Note also that, as in the case of password-based authentication, biometric-based authentication gains from its combination with another independent authenticating mechanism (e.g. the proof of knowledge of a passphrase).

type of anonymous authentication (henceforth referred to as hierarchical anonymous authentication or HAA).

One application of HAA is *traffic filtering and routing in partially-hidden networks.* Suppose, for instance, that a company $\mathcal{O}$ wants to achieve the following goals: (G1) $\mathcal{O}$ wants to reveal only part of its network map to third parties; (G2) $\mathcal{O}$ wishes to stop unwanted traffic passing through its network; and, (G3) when legitimate network traffic is intended for hidden parts of its network, $\mathcal{O}$ wants to route this traffic to the right destination nodes.

G1 may be motivated by the desire to achieve both increased privacy and improved resistance against network mapping attacks. (These attacks precede most harmful network-based attacks.) G2 is useful to combat denial of service attacks, and this goal could be achieved by stopping all network messages passing through $\mathcal{O}$'s network, except those which come from $\mathcal{O}$'s subnets and those which respond to recent messages coming from $\mathcal{O}$'s subnets. (Remark that, if $\mathcal{O}$'s network is virtual, then network messages coming from $\mathcal{O}$'s subnets may either enter or exit $\mathcal{O}$'s physical network. In particular, network messages coming from a mobile device belonging to $\mathcal{O}$ can send messages which virtually come from $\mathcal{O}$'s network, but physically come from another network.) Finally, G3 is essential to the use of networks. In particular, $\mathcal{O}$ could use a designated node of each one of its subnets in order to route messages intended for specific nodes of this subnet.

Consequently, $\mathcal{O}$ could organize its network nodes into hierarchically-structured and non-overlapping subnets, and use HAA in order to achieve G1, G2 and G3. HAA is therefore useful for traffic filtering and routing in partially-hidden and hierarchically-structured networks.

Another application of HAA is *privacy-enhanced access control.* Suppose, for instance, that an organization $\widetilde{\mathcal{O}}$ wants to offer a large number of private services to its customers. These services could be hierarchically structured, in such a way that eligibility to certain services guarantees access to other (hierarchically lower) services. Thus, each service of $\widetilde{\mathcal{O}}$ could be associated with a subgroup of a large group, where the subgroups would be hierarchically structured. In order to protect the anonymity of customers, a trusted third party (e.g. another organization) could provide each one of $\widetilde{\mathcal{O}}$'s customers with special keys associated with various services. Thus, $\mathcal{O}$ could determine whether a user is eligible to a specific service, given a cryptographic signature issued by this user with one of her special keys.

Imagine, for instance, that $\widetilde{\mathcal{O}}$ wants to provide hierarchically-structured private online consultation services. $\widetilde{\mathcal{O}}$ could partner with a trusted third party (e.g. a government or bank) who would provide eligible (e.g. paying) clients with a set of special keys giving them access to multiple private online services. Note that $\widetilde{\mathcal{O}}$'s clients could also provide anonymous feedback on the services for which they

are eligible. Another similar application (suggested by Boneh and Franklin [BF99]) is that of a hotel that wants to offer privacy-enhanced access control on selected sections of its facilities. These sections could be hierarchically structured, and hotel residents could be given privacy-enhancing keys giving access to selected sections of the hotel (based on the room location of key holders). Moreover, in case of law-mandated investigation, the privacy of customers could be revoked. (Note that such a privacy-enhancing service could attract customers and thereby increase revenue.)

It should be noted that one attempt to support HAA is to proceed as follows: assign a unique secret key to each subgroup, give each subgroup key to all members of the corresponding subgroup, and ask each system user to cryptographically sign statements with their given key in order to prove their association with a specific subgroup. However, the following two problems arise. First, the anonymity of users is not revocable. Second, the revocation of a user demands the generation and distribution of a new subgroup key for all unrevoked members of the revoked user's subgroup.

Another attempt to support HAA is to use a cryptographic primitive called group signature [CvH91]. This primitive enables all the members of a group to sign on behalf of the group, in such a way that their identity is kept secret from all but a designated group leader. Thus, one can associate a group signature scheme with each subgroup of a large group, and use these schemes to support HAA, when the subgroups are hierarchically structured. This scheme, however, has the following two limitations. First, a central trusted party must generate a distinct set of public parameters for each subgroup. Ideally, there would be one small set of public parameters which could be used to verify signatures issued on behalf of all subgroups. Second, one central trusted party (associated with the large group) must both generate each one of the subgroup parameters, and secretly send secret parameters to each subgroup leader. When the hierarchy is large (or becomes large as subgroups are added), it is more efficient to have a scheme whereby a central party dynamically delegates to other entities (e.g. first-level subgroup leaders) the task of generating and distributing lower-level secret subgroup parameters. Hence, this second attempt to support HAA is neither space efficient nor computationally efficient in the case of large hierarchies. Consequently, for increased scalability, a novel cryptographic primitive supporting HAA needs to be designed.

1.1.3 Limitations of Current Solutions

Most cryptosystems currently used to handle large user communities are *certificate-based* public-key (CBPK) cryptosystems. Known alternatives include *certificate-less* public-key (CLPK) cryptosystems, *self-certified* public-key (SCPK) cryptosystems,

and *identity-based* public-key (IBPK) cryptosystems[3]. This section compares these options, selects IBPK cryptography (IBPKC) as the paradigm of choice for this dissertation, and explains how IBPKC currently tackles the three specific challenges which this dissertation deals with.

Comparison Criteria

When public-key cryptography (PKC) is used in large and distributed user communities, the following questions must be answered:

Q1 How are users distinguished ?

Q2 What public information is required for encryption and signature verification ?

Q3 What information is required for decryption and signing?

Q4 Which mechanisms are used to enforce revocation rules?

Q5 Which mechanisms are used to detect the compromise of private keys ?

Q6 Is key escrow[4] required ?

Q1 refers to the so-called *naming problem* of PKC. Q2 includes the issue of *trust*, since the public information required for encryption and signature verification needs to be trusted by its users. Q3 can be used to measure the amount of interaction between users and authoritative entities of public-key cryptosystems. Q4 deals with the issue of *revocation*, and Q5 with the issue of *stolen-key detection*. Finally, Q6 aims at identifying the level of trust placed on authoritative entities of public-key cryptosystems.

Overview of Cryptographic Paradigms

In the context of certificate-based public-key cryptography (CBPKC) [DH76, AL02], each user selects a private-public key pair, and asks a trusted authority (TA) to certify her (i.e. the user's) public key. The TA assigns an identifier to the user and issues a public-key certificate which binds this identifier with the user's public key, for a given time period. However, it is possible that a user's public key is revoked before the

[3]SCPK and IBPK cryptosystems are two classes of *implicitly-certified* public-key cryptosystems. (Cf. [MvOV97] pp.562–566.)

[4]Key escrow refers to the fact that a trusted authority maintains a copy of each user's private key.

aforementioned time period ends. Hence, the authenticated and up-to-date[5] public-key certificate of an entity is needed in order to either encrypt information for that user or verify signatures from that entity[6].

In the framework of certificate-less public-key cryptography (CLPKC) [ARP03, Gen03], the obtainment of up-to-date public certificates is not required. Instead, each user selects a user-based partial private key, derives a public key (from this partial private key and a chosen identifier), requests an up-to-date server-based partial private key from a TA, and uses her two partial private keys to generate a (full) private key. (The TA needs to know the identifier of a user in order to generate this user's server-based partial private key.) In order to encrypt a message to a user (or verify a signature issued by this user), the encryptor (respectively the signature verifier) needs to obtain this user's public key (along with authentic system-level public parameters). Assuming that both the user-based partial private key and the identifier of a user do not change regularly, it follows that, in CLPK, decryptors and signers have to send their public key once (to encryptors and signature verifiers), but must regularly obtain their up-to-date server-based partial private keys.

It should be noted also that Gentry [Gen03] used the moniker *certificate-based cryptography* to refer to a variant of CLPK cryptography (CLPKC). However, to avoid confusion, we use the term CLPKC to refer to both Gentry's variant and the paradigm described in the above paragraph.

Another known paradigm is self-certified public-key cryptography (SCPKC) [Gir91]. In SCPK cryptosystems, users generate a private-public key pair and give the public key to a TA who produces an up-to-date partial public key (using the user's identifier). This up-to-date partial public key is needed (along with authentic system-level public parameters) in order to encrypt messages for this user and verify signatures associated with this user. In particular, encryptors and signature verifiers recover this user's public key from the user's identifier and the user's partial public key.

Finally, identity-based (ID-based) public-key cryptosystems [Sha84, BF01] allows public keys to be derived from user identifiers. Each user obtains, from a TA, her private key, and this key is generated based on the user's identifier. Moreover, encryptors and signature verifiers can perform their respective task without obtaining any up-to-date information from a TA. However, both of these tasks require the use of authentic system-level public parameters, and each decryptor/signer must choose one of the following options (if revocation is to be supported): either she binds her identifier with a short time period, and must then regularly obtain an up-to-date

[5]The phrase *up-to-date* refers to the fact that public key certificates must be obtained along with associated recent and detailed revocation information.

[6]This certificate is cryptographically signed by a TA whose public key is a system-level parameter that must be obtained by encryptors and signature verifiers.

private key from a TA; or, she must cooperate with an online agent (known as a *SEcurity Mediator* or SEM) to complete each decryption and signature process. The first option requires encryptors and signature verifiers to concatenate user identifiers with a short time period, and the second option [BDT02, BDTW01, LQ03a] is transparent to encryptors and signature verifiers. Moreover, SEMs cannot (on their own) decrypt or sign on behalf of users. (For this reason, SEMs are said to be *semi-trusted* agents.)

Comparison of Cryptographic Paradigms

None of the compared classes of cryptosystems provide a practical answer to Q1. In other words, no general practical solution has been proposed for the Naming problem of PKC. However, various solutions have been suggested for specific contexts (e.g. states and companies), including the use of full civil names, email addresses, social security numbers, IP addresses, etc.

With respect to Q2, there are two aspects to consider, namely the need of authentic system-level public parameters, and the need of up-to-date user-specific information for encryption and signature verification.

If all parties asking for authentic system-level public parameters can be quickly reached by central entities, then the inquiring parties can be alerted when a change occurs in system-wide parameters. For instance, encryptors and signature verifiers could be required to give their email addresses to central entities the first time they acquire system-level parameters. This information could then be used, when the aforementioned parameters change, in order to alert the parameters-acquiring parties (e.g. via email messages cryptographically signed by a trusted third party). The alert messages could also be formatted according to predefined rules, so that email applications automatically interpret them, verify their validity, and replace revoked system-level parameter with new ones (whenever these are provided). If, however, the aforementioned contact information cannot be easily gathered, then cryptographic paradigms requiring authentic system-level parameters to be distributed are, in general, not preferable to paradigms that do not feature such a requirement. This suggests that, with respect to the distribution of authentic system-level parameters, CBPKC is, in general, preferable to the other aforementioned paradigms. However, it can be argued that, when public-key cryptosystems are used for secure intra-*system* communications, encryptors and signature verifiers can be efficiently notified of changes in system-level public parameters. Hence, all the paradigms herein studied feature similar constraints (with regard to the system-level aspect of Q2), when they are used to support secure communications in either one of the following scenarios: (1)

intra-organizational communications[7]; and (2) *inter-organizational* communications between members of organizations that trust one another[8].

The other aspect of Q2 concerns the need of up-to-date user-specific information for encryption and signature verification. With regards to this aspect, ID-based cryptosystems are the most convenient to use since encryptors and signature verifiers only need to obtain the identifier of each intended decryptor or claimed signer. CLPK cryptosystems are also convenient, since up-to-date information is not required for encryption and signature verification. Moreover, note that the management (creation, storage, deployment, revocation, updating) of public-key certificates is cumbersome for systems supporting large distributed user populations (cf. §2.1).

The issue of trust in public information available for encryption and signature verification remains a challenge for all compared cryptographic frameworks. One solution proposed for this problem is to use a mechanism whereby a TA representing a group of users certifies that public-key certificates and publicly available system parameters issued by another TA are reliable. (This process is called inter-domain certification [Tur00].)

With respect to Q3 and Q4, it should be noted that SEMs can be used with any of the compared paradigms. Moreover, we note that, when access to online agents is possible at any time, SEMs are more space-efficient than solutions based on private keys that are associated with (short) time periods. Indeed, the latter class of solutions force users to safeguard *all* time-bound keys, in order to be able to decrypt ciphertexts associated with past time periods. In fact, solutions based on short time periods (i.e. solutions providing fast revocation of users'security privileges) generally assume that users are able to obtain fresh (partial or full) private keys from online entities. Certain classes of cryptosystems – called *forward-secure* cryptosystems – assume that time is divided into time periods, bind each private key to one time period, and allow any user having a private key associated with a given time period to generate a private key associated with any subsequent time period. However, forward secure schemes do not address the issue of revocation of user privileges. Moreover, the issuing of time-bound private keys by central trusted entities is computationally intensive in large user communities. The use of multiple key-issuing TAs could reduce this computational burden, but this poses a design challenge because the compromise of one TA should not affect the confidentiality of private keys issued by other TAs. Hence, the use of SEMs appears to be the most effective solution for fast revocation of security privileges, when access to online services is guaranteed. (This assumption is generally realistic, since even planes offer access to the Internet.)

[7]e.g. email communications between employees of country's government and cell phone communications between parties having a common cell phone service provide.

[8]e.g. students of two universities that trust each other.

Concerning Q5, known techniques for the detection of compromised keys [JvO99] are applicable to all compared classes of cryptosystems. These techniques are based on the concepts of synchronization and independence of secret keys. In order to issue signatures and decrypt documents, users must interact with a trusted party, and have pieces of information which are coherent with information kept by the trusted party. Moreover, the synchronized (public and secret) information should not be derivable from users'signing or decryption keys.

Finally, Q6 draws an important distinction between the compared paradigms. Indeed, CBPK, CLPK, and SCPK cryptosystems do not require key escrow, while the ID-based public-key cryptography (IBPKC) assumes the existence of a TA that generates each user's decryption key. It should be noted that, since ID-based signatures can be sent with accompanying information, it is possible (cf. [CZK03]) to design an ID-based signature scheme whose key generation algorithm allows users to keep their signing key secret from their TA, and yet obtain, from this TA, a proof that they have the right to sign documents with specifiable public keys. Hence, IBPK cryptosystems only require ID-based *decryption* keys to be escrowed (not ID-based signing keys).

Comparison Summary

This dissertation assumes the use of either one of the above revocation techniques to support ID-based key revocation. Moreover, concerning Q5, known techniques for the detection of compromised keys [JvO99] are applicable to CBPK, CLPK, and IBPK cryptosystems.

As a summary (cf. Table 1.1), let us recall that ID-based cryptography provides a convenient solution to one of the important challenges related to the use of cryptography in large and distributed user communities. Moreover, it should be noted that existing solutions related to the issues of *trust*, *revocation*, and *detection of compromised keys* can be used for ID-based cryptosystems, as well as for the other classes of cryptosystems. Furthermore, we note that any cryptographic signature can be sent with accompanying information (such as the public-key certificate or the public key of a signer). Hence, the benefits of ID-based signature are, in general, less compelling than those of ID-based encryption. Noteworthy is the fact that ID-based encryption requires decryption keys to be escrowed by a trusted authority. Therefore, a decisive factor in the choice of public-key cryptosystem is the effect of decryption-key escrow in specific applications. If decryption-key escrow is more important than convenience of encryption and efficiency in the management of users'public-key certificates, then, in large user communities, ID-based public-key encryption is not the best option (certificate-less encryption is). If, on the other hand, the fact that decryption keys are escrowed by a trusted entity is less important than the significant convenience and efficiency gained by the use ID-based public-key encryption, then IBPKE is

	CBPKC	CLPKC	SCPKC	IBPKC
Naming	Not Solved	Not Solved	Not Solved	Not Solved
Encryption and Signature Verification	Fresh User's Certificate and ID Required	User's Pub. Key and ID Required	Fresh User's Partial Pub. Key and ID Required	User's ID Required
Trust	Inter-Domain Certification	Inter-Domain Certification	Inter-Domain Certification	Inter-Domain Certification
Decryption and Signing	Decryption and Signing Keys Required	Partial Decryption/Signing Keys Required	Decryption and Signing Keys Required	Decryption and Signing Keys Required
Revocation Enforcement	Online Mediation of Decryption and Signature	Online Mediation of Decryption and Signature	Online Mediation of Decryption and Signature	Online Mediation of Decryption and Signature
Detection of Compromised Key	Online Mediation of Decryption and Signature	Online Mediation of Decryption and Signature	Online Mediation of Decryption and Signature	Online Mediation of Decryption and Signature
Key Escrow	No Key Escrow	No Key Escrow	No Key Escrow	Escrow of Decryption Keys Only

Table 1.1: Comparison of Known Cryptographic Paradigms.

the best option for applications targeting either intra-organizational electronic communications or inter-organizational electronic communications between members of organizations that trust one another.

It should be noted, however, that certain classes of ID-based signature schemes might be significantly more interesting than specific classes of other (e.g. certificate-based) signature schemes. This is the case for existing forward-secure signature schemes [CHYC04]. Indeed, the known examples of these schemes can only be derived from ID-based signature schemes.

Consequently, *due to the benefits of ID-based cryptography concerning the creation, management, and distribution of public keys and public-key certificates in large user communities, this dissertation will investigate the use of IBPK cryptosystems.* The fact that decryption keys are escrowed by trusted authorities will therefore not be considered more important than the aforementioned benefits of ID-based cryptography.

Next, we review current ID-based cryptographic solutions to the following problems: key generation from non-repeatable information; encrypted message authentication; and hierarchical anonymous authentication.

ID-Based Key Generation from Non-Repeatable Information

Sahai and Waters [SW05] recently described the first ID-based encryption scheme in which identities are viewed as sets of public features, these features are extracted from non-repeatable information (e.g. biometric readings), and the extracted features are used to generate private keys that support decryption. Sahai and Waters' scheme is a *threshold attribute-based encryption* (thABE) scheme, because each ciphertext c encrypted for an identity ω is associated with a threshold value d. Moreover, decryptors must have keys associated with at least d elements of ω, in order to decipher c.

However, two limitations of Sahai and Waters'scheme are: (1) its high computational decryption cost (each decryption requires $2d$ expensive operations, where d is the threshold value associated with a given ciphertext); and (2) its lack of flexibility (no mechanism is provided whereby threshold values can be specified, by encryptors, at encryption time, for an ad-hoc set of biometric features[9]).

Since the number of features used to identify human users is relatively large (e.g. 249 in the case of Daugman's iris scanning algorithm [Dau04]) and evolves with both time (i.e. research discoveries) and the type of devices used for biometric-based authentication, it follows that a more efficient and flexible thABE scheme must be designed for biometric-based authentication and access control applications.

IBPKC Support of Encrypted Message Authentication

Before presenting the state-of-the-art in ID-based cryptographic support of EMA, we recall that this dissertation seeks solutions which are scalable to large user populations. In the case of biometric-based authentication and access control, the number of independent biometric features whose combinations are used to identify human users remains below 700 (15 in the case of Monrose et al.'s key-stroke-based algorithm [MRLW01], 46 in the case of Monrose et al.'s voice-based algorithm [MRW99], 249 in the case of Daugman's iris-based algorithm [Dau04], and 648 in the case of Jain et al.'s fingerprint-based method [JRP01]). However, when ID-based private keys are associated with users or nodes of networks (instead of biometric features), then one is interested in designing schemes which scale to more than 10,000 users (for a large university/corporation) or 10,000 IP addresses (in the case of a reasonably large network). ID-based encryption (IBE) schemes which are suitable for this class of populations are termed hierarchical IBE (HIBE) schemes, because they enable a root private key generator (PKG) to delegate, to other (hierarchically-lower) PKGs,

[9]Threshold values and sets of biometric features could be automatically specified, at encryption time, depending on the accuracy of the biometric reader and the associated environmental condition (including light/noise exposure and humidity level in the reader's room).

the task of verifying users'identity, of generating a key for each of these users, and of secretly sending these users their decryption/signing keys.

Cryptographic mechanisms which support encrypted message authentication form a subclass of *signcryption* schemes. Signcryption schemes efficiently combine encryption and signature schemes. A signcryption scheme is essentially composed of a *Key Generation* method, a *Signcryption* procedure (which combines signature and encryption), and an *Unsigncryption* algorithm (which combines decryption and signature verification). Signcryption schemes can have various properties. One of these properties – termed *public ciphertext authenticity* (PCA) – enables third parties to verify, without the help of intended recipients, whether: (1) any given signcrypted message was unsigncrypted and re-signcrypted; (2) any given signcrypted message was issued by a claimed user, without compromising the confidentiality of the signcrypted message.

No hierarchical ID-based signcryption (HIBSC) scheme featuring public ciphertext authenticity is known. Moreover, the only known HIBSC scheme has very high computational requirements. For instance, its unsigncryption procedures requires $O(t)$ expensive operations, where t is the hierarchical depth of an intended recipient. Consequently, the design of an efficient HIBSC scheme with PCA is needed to support EMA in large and distributed user communities.

IBPKC Support of Hierarchical Anonymous Authentication

Recall that cryptographic schemes which have been specifically designed to support authentication with revocable anonymity are called *group signature* schemes. A group signature scheme allows each member of a given group to sign on behalf of the group in such a way that only a designated group leader is able to identify the issuer of a valid group signature.

Recall also that, for improved scalability to large hierarchies, and for increased space and computational efficiency, a new group-signature-like primitive needs to be defined in order to support HAA. Such a primitive would allow each leader of subgroup (in a hierarchy of subgroups) to generate dynamically the secret parameters of her hierarchical descendant subgroups. Moreover, such a primitive would allow third parties to verify whether a given signature is associated with a claimed subgroup.

However, no such scheme is known (neither in the certificate-based cryptographic framework nor in the context of ID-based cryptography).

1.2 Dissertation's Contributions

In order to answer our three *fundamental questions* and to address the aforementioned limitations of current solutions, we describe, in this dissertation, a set of mechanisms which use ID-based cryptography and scale to large user communities. Our contributions are outlined in the following sections.

1.2.1 Threshold Attribute-Based Encryption

The first main contribution [NAM05d, NAM05a] of this dissertation is the design of a provably secure[10] threshold attribute-based encryption (thABE) scheme featuring both constant-time decryption and a mechanism enabling encryptors to specify, at encryption time, multiple attribute sets with their corresponding threshold values.

This encryption scheme includes a key generation algorithm which derives keys from attributes that can be extracted from non-repeatable information. Thus, the proposed thABE scheme can be used to support biometric-based access control, when an algorithm is used to extract features from biometric readings (see, e.g., [MRLW01, MRW99, Dau04, JRP01]). Moreover, it should be noted that the proposed thABE scheme has a fast decryption procedure. (In comparison with Sahai and Waters'thABE scheme [SW05], our thABE scheme reduces from 498 to 2 the number of bilinear pairings [Jou00] used to decrypt a ciphertext that was encrypted with the 249 biometric features of Daugman's iris recognition algorithm.)

1.2.2 Hierarchical ID-based Signcryption with Public Ciphertext Authenticity

This dissertation's second main contribution [NAM04, NAM05c, NMA04] is the design of a provably secure hierarchical ID-based signcryption (HIBSC) scheme featuring both public ciphertext authenticity (PCA) and forward-security (FS). FS is the property whereby the compromise of a sender's signing key does not compromise the confidentiality of messages signcrypted by this sender.

In order to design an efficient HIBSC with PCA and FS, the following components are needed: first, one needs to describe an efficient hierarchical ID-based *Key Generation* (HIBKG) procedure; second, one must devise an encryption/decryption scheme and a signing/verifying scheme, from the HIBKG procedure; third, one needs

[10]The qualifier *provably-secure* refers to the fact that a given procedure is resistant to specific attacks (typically presented in the form of a *threat model*). Therefore, the value of the term *provably secure* depends on the empirical suitability of the associated threat model.

to design a mechanism enabling authentication of signcrypted messages by third parties; fourth, one must describe a mechanism providing forward security; fifth, one needs to combine the above mechanisms in an efficient way.

Boneh et al. [BBG05] recently proposed an efficient HIBE scheme, which was subsequently used to design an efficient hierarchical ID-based signature (HIBS) scheme [KMPR05]. However, this dissertation presents a HIBE scheme and a HIBS scheme which are derived from a proposed HIBKG algorithm. This algorithm was found independently from Boneh and al.'s HIBKG algorithm (henceforth called BBG-HIBKG).

Benefits of Proposed HIBKG Algorithm

Benefits of the proposed HIBKG include its efficiency, and the fact that it yields both an efficient forward-secure HIBE scheme and an efficient append-only signature scheme.

- **Computational Efficiency**:

  The proposed HIBKG algorithm is more efficient than BBG-HIBKG. More specifically, it runs in constant time, and issues constant-size keys, while BBG-HIBKG runs in $O(\ell)$ steps and issues keys of size $O(\ell - t)$, where ℓ and t respectively denote the maximal hierarchical depth of a user and the hierarchical depth of an arbitrary user.

- **Efficient Forward-Secure HIBE Scheme**:

  The computational efficiency of the proposed HIBKG algorithm yields an efficient forward-secure HIBE scheme.

  In a forward-secure encryption scheme, each decryption key is associated with a time period, and keys must be *evolved* at the beginning of each time period. This evolution process consists in using the key associated with the time period in order to obtain the key associated with the following time period. When a document is encrypted, the resulting ciphertext c is associated with a time period t_p, and decryptors must have a decryption key associated with a time period preceding t_p, in order to decipher c. The only known forward-secure HIBE scheme (fsHIBE) scheme was suggested by Yao et al. [YFDL04]. When the proposed HIBKG algorithm is used to obtain an encryption scheme (as the proposed HIBE scheme), this HIBE scheme can be used to instantiate Yao et al.'s fsHIBE scheme. The resulting fsHIBE scheme is then more efficient than a similar instantiation with Boneh et al.'s HIBE (BBG-HIBE) scheme, especially when time periods are short. Indeed, one of the most frequent operations performed by decryptors, in a fsHIBE scheme, is key evolution. The cost

of key evolution therefore has a direct impact both on the computational requirements and the power consumption of decrypting engines. For instance, the computational cost of key evolution in Yao et al.'s scheme is about twice that of the underlying HIBE key generation algorithm. Since the computational cost of BBG-HIBE key generation algorithm is linear in the *maximal* hierarchical depth of users, while the cost of the proposed HIBKG algorithm is constant, it follows that the proposed HIBKG algorithm allows to build portable fsHIBE decrypting devices whose power-autonomy is independent from user hierarchy of the associated fsHIBE scheme. Moreover, the space available to store fsHIBE keys may be constrained for some decrypting devices. With regards to this aspect, recall that the proposed HIBKG algorithm issues constant-size keys. Thus, an instantiation of Yao et al.'s scheme with the proposed HIBE scheme yields an efficient scheme, suitable for resource-constrained devices (such as smartcards).

As demonstrated by Yao et al. [YFDL04], fsHIBE schemes can be used for role-based access control applications [FSG+01], especially when the number of hierarchically-structured roles is large. Moreover, fsHIBE schemes can be used to build mechanisms which make restricted information quickly accessible to large and distributed user communities, after a predetermined time [NAM05c].

- **Efficient Append-Only Signature Scheme**:

  The computational efficiency of the proposed HIBKG algorithm also yields an efficient append-only signature (AOS) scheme.

  The concept of AOS was recently introduced by Kiltz et al. [KMPR05]. Given an AOS signature on a message m_1, any recipient of the OAS signature is only able to compute an AOS signature on the concatenation $m_1||m_2$ of m_1 with any message m_2. As pointed out by Kiltz et al. [KMPR05], HIBS schemes can be efficiently transformed into AOS schemes. The efficiency of the resulting AOS scheme heavily depends on the (space and computational) efficiency of the underlying HIBS's *Key Generation* algorithm. Moreover, the computational cost of signature verification in the resulting AOS scheme is essentially the sum of the costs of *Signature* and *Verification* in the underlying HIBS scheme. When our proposed HIBS scheme is used to build an AOS scheme (henceforth referred to as NewAOS), the resulting scheme issues *constant-size* AOS signatures, and features both a *constant time Append*[11] procedure and a *constant-time* verification algorithm. This was impossible to achieve with an AOS scheme (henceforth called BBG-OAS) derived from Boneh et al.'s HIBKG algorithm. Indeed, if d is the number of appended messages associated with a given BBG-AOS signature,

[11]The *Append* procedure of an AOS scheme enables any user who receives an AOS signature on message, to produce an AOS signature on the concatenation of this message with another one.

then this AOS signature takes $2\sqrt{d}+1$ space units (see Table 5.2). Moreover, BBG-AOS's *Append* procedure runs in $\sqrt{d}+3$ steps, and BBG-AOS's *Verification* method takes a total of $d+2\sqrt{d}+1$ steps including $\sqrt{d}+1$ expensive steps.

AOS schemes can be used to provide both path authentication in Internet routing [KMPR05], and resource delegation in privilege management systems [KMPR05]. Both applications would benefit from an AOS scheme with constant-size signatures, constant-time *Append* procedure, and fast verification method [KMPR05]. However, the possibility of designing such a scheme has been an open question [KMPR05]. Consequently, our HIBKG algorithm contributes to research in Internet routing and privilege management.

To summarize this section, let us recall that the second main contribution of this dissertation is the design of a provably secure hierarchical ID-based signcryption scheme with public ciphertext authenticity and forward security. This scheme is based on a proposed efficient hierarchical ID-based key generation algorithm which yields (in addition to the aforementioned signcryption scheme) both an efficient forward-secure HIBE scheme and an efficient append-only signature scheme.

1.2.3 Hierarchical ID-based Group Signature with Public Subgroup Identification

The third main contribution of this dissertation [NMA06] is the design of a provably secure hierarchical ID-based group signature (HIBGS) scheme with public subgroup identification (PSI). PSI allows users of a HIBGS system to determine, given publicly available information only, whether the presumed issuer of a given HIBGS signature belongs to a particular subgroup (in a hierarchy of subgroups). Hence, HIBGS with PSI offers a solution to the HAA problem.

Our proposed HIBGS scheme provides several security and efficiency benefits. First, the scheme prevents subgroup members from being able to be impersonated by their subgroup managers (unlike Boneh and Sacham's group signature scheme [BS04]). Second, the proposed scheme enables signers to reuse the same private key to generate multiple group signatures. This improves on Chen et al.'s ID-based GS scheme [CZK03], which requires the use of a different signing key for each signature that is issued. Third, the HIBGS scheme features a signature verification procedure whose computational cost is only logarithmic with respect to the number $|GM|$ of subgroups in a given hierarchy. This stands in contrast with Boneh and Franklin's anonymous authentication scheme with subset queries [BF99], whose signing and signature verification costs grow linearly with respect to $|GM|$.

Dissertation's Outline

In this chapter, we have presented a motivation for our work, and have outlined the contributions of this dissertation. Chapter 2 reviews past research on threshold attribute-based encryption, hierarchical ID-based signcryption, and hierarchical ID-based group signature. Chapter 3 presents our proposed thABE scheme. Chapter 4 describes our proposed HIBE and HIBS schemes, and two of its applications are investigated in Chapter 5. Chapter 6 presents our proposed HIBSC scheme, and Chapter 7 describes our HIBGS scheme. Finally, Chapter 8 concludes the dissertation.

Chapter 2

Background Research

The aim of this chapter is to review past research related to cryptography for large user communities, threshold attribute-based encryption, hierarchical signcryption, and hierarchical group signature.

2.1 Certificate-Based Cryptography

Most widely deployed public-key cryptosystems are *certificate-based* cryptosystems[1]. Certificate-based public-key infrastructures (CPKIs) [AL02] typically require that the authenticated and up-to-date public-key certificate of an entity be obtained in order to encrypt information for that entity. However, maintaining high availability of up-to-date public-key certificates in large and distributed user communities implies high computational, bandwidth, storage, and possibly financial cost requirements[2]. These certificates need to be generated and sent to many users. Moreover, the certificates need to be frequently verified (since their validity depends on the revocation status of the corresponding users.) When users are revoked, a mechanism needs to be set in place for encryptors to be notified accordingly (this is typically done by having encryptors inspect revocation data structures). Since certificates typically expire after a given time, a procedure also needs to be established for old certificates to be void[3], and new certificates to be made available to encryptors, in a timely fashion. Even the trust of public-key certificates' validity is a challenging problem – especially when encryptors do not have prior trust relationships with parties certifying the validity of recipients'certificates. Consequently, the management of public-key certificates in

[1]Very few implementations of certificate-less and self-certified public-key cryptosystems are known. These classes of cryptosystems shall therefore not be discussed further. See §1.1.3.

[2]Most reliable public-key certificates need to be bought.

[3]e.g. by checking expiry dates embedded in public-key certificates.

CPKIs is cumbersome when these certificates are used to support secure communications in large and distributed user communities [Gut02].

2.2 Identity-Based Cryptography

Identity-based (ID-based) cryptography was originally suggested by Shamir [Sha84], in 1984, as a method to avoid the exchange of public keys and the use of public-key certificates in public key infrastructures. The idea is to derive public keys directly from user identifiers, such as email addresses, telephone numbers, IP addresses, or social insurance numbers. On the other hand, each identity-based private key is generated as a combination of a user public key and a system-level secret key that is kept private by a central trusted authority. Thus, public keys can be derived from any string and private keys must be securely generated and delivered to users by a central authority (also known as the *Private Key Generator* or *PKG*). Since public keys can be derived from random strings, one may embed the expiration time of a public key in the corresponding string. Assuming that PKGs do not issue private keys corresponding to expiration time which has passed, ID-based encryption scheme can therefore easily support the *automatic* time-based revocation of private keys[4]. It should also be noted that ID-based private keys can be generated just before decryption time. When used, this features therefore allows to reduce the time exposure of ID-based private keys. Such a benefit stands in contrast with certificated-based private keys which must be generated before encryption time (in conjunction with their associated public keys). An important area of research in ID-based cryptography is *private key revocation.* In particular, one class of schemes enabling the revocation of decryption/signing privileges involves the use of online cryptographic agents called SEMs. (This technique was introduced in Chapter 1 and will be discussed later, in greater detail.) Another important issue is that of system-level parameter distributions: for ID-based cryptosystems to be used, encryptors and signature verifiers must obtain *certified* public parameters associated with their intended decryptors and claimed signers. Since these parameters can be reused for all users associated with the PKGs of aforementioned decryptors/signers, IBPKC requires reusable certified system-level public parameters. These parameters can be certified using mechanisms used in existing certificated-based public key infrastructures [Tur00]. However, the certification of system-level parameters may impede wide adoption of ID-based cryptosystems for applications involving user groups having no trust relationships (such as companies

[4]Identifiers could also include other pieces of information such as large random numbers (also known as *nonces*) and user privilege identifiers (e.g. the string "system Admin"). When a nonce is used, users must obtain the corresponding decryption key after reception of the associated ciphertext, and this key is only useful for this ciphertext. When a privilege identifier is used, the user must have the corresponding privilege at the time of decryption.

whose email-related ID-based system-level parameters are not (indirectly) certified by a common trusted party).

Pairing-Based IBE

From 1984 to 2001, many identity-based cryptographic schemes were proposed (see, e.g., [DQ86, Tan87, MY91, Coc01]). However, it was only in 2001 that Boneh and Franklin [BF01] proposed the first efficient identity-based encryption scheme, based on bilinear pairings over Gap-Diffie-Hellman groups, which we define below. (Note, nevertheless, that efficient ID-based signature schemes were described before 2001, including, for instance, [GQ88] and [SOK00].)

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two Abelian groups [Lan93] of prime order q, where $\mathbb{G}_1$ is written additively and $\mathbb{G}_2$ is written multiplicatively. Let $P_0^{(1)} \in \mathbb{G}_1^*$ be a generator of $\mathbb{G}_1$. A *bilinear pairing* $\hat{e}$ is a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ such that $\hat{e}(aP_0^{(1)}, bP_0^{(1)}) = \hat{e}(P_0^{(1)}, P_0^{(1)})^{ab}$ for all $a, b \in \mathbb{Z}_q^*$. (bilinear pairings can be constructed using *Weil pairings* (cf. section 5 of [BF01]) and – more efficiently – using *Tate pairings* on elliptic curves [BKLS02, BLS02, BLS01, Gal01, GHS02].) The map $\hat{e}$ is said to be an *admissible pairing* if it is a *non-degenerate* (i.e. $\hat{e}$ does not send all pairs of points in $\mathbb{G}_1 \times \mathbb{G}_1$ to the identity in $\mathbb{G}_2$), computable (i.e. $\hat{e}$ efficiently computes the image of any pair of points in $\mathbb{G}_1 \times \mathbb{G}_1$) *bilinear pairing*. Let $\mathcal{A}$ be an attacker modelled as a probabilistic Turing machine [Tur37]. The *computational Diffie-Hellman* (*CDH*) problem [BF01] is that in which $\mathcal{A}$ is to compute $abP_0^{(1)}$, given $(\mathbb{G}_1, q, P_0^{(1)}, aP_0^{(1)}, bP_0^{(1)})$ and a security parameter k, where $a, b \in \mathbb{Z}_q^*$ are unknown. The success (or *advantage*) of $\mathcal{A}$ is then defined as $Pr[\mathcal{A}$ computes $abP_0^{(1)}]$. The *decisional Diffie-Hellman* (*DDH*) problem [BF01] is that in which $\mathcal{A}$ is to guess whether $cP_0^{(1)} = abP_0^{(1)}$, given $(\mathbb{G}_1, q, P_0^{(1)}, aP_0^{(1)}, bP_0^{(1)}, cP_0^{(1)})$ and a security parameter k, where $a, b, c \in \mathbb{Z}_q^*$ are unknown. The success (or *advantage*) of $\mathcal{A}$ is then defined as $Pr[\mathcal{A}$ makes a right guess that $cP_0^{(1)} = abP_0^{(1)}]$. $\mathbb{G}_1$ is called a *Gap-Diffie-Hellman group* if the CDH is intractable in $\mathbb{G}_1$, but the the DDH can be solved in polynomial time in $\mathbb{G}_1$. The *bilinear Diffie-Hellman* (*BDH*) problem [BF01] is that in which $\mathcal{A}$ is to compute $\hat{e}(P_0^{(1)}, P_0^{(1)})^{abc}$ given a security parameter k, the tuple $(\mathbb{G}_1, q, P_0^{(1)}, aP_0^{(1)}, bP_0^{(1)}, cP_0^{(1)})$ where $a, b, c \in \mathbb{Z}_q^*$ are unknown, and given the fact that the *CDH* problem cannot be solved in polynomial time with non-negligible advantage in both $\mathbb{G}_1$ and $\mathbb{G}_2$. The success (or *advantage*) of $\mathcal{A}$ is then defined as $Pr[\mathcal{A}$ outputs $\hat{e}(P_0^{(1)}, P_0^{(1)})^{abc}]$.

Boneh and Franklin's encryption scheme [BF01] has been summarized as follows [ARP03]:

- **Setup**: given a security parameter k, the PKG chooses groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order $q > 2k$, a generator $P_0^{(1)}$ of $\mathbb{G}_1$, a master key $s \in \mathbb{Z}_q^*$ and the associated public key $P_{pub} = sP_0^{(1)}$. The plaintext space is $\mathcal{M} = \{0,1\}^n$ for a fixed n, while $\mathcal{C} = \mathbb{G}_1^* \times \{0,1\}^n$ is the ciphertext space. Hash functions $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{G}_1^*$ and $\mathcal{H}_2 : \mathbb{G}_2 \to \{0,1\}^n$ are also chosen. The systems public parameters are $params = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P_0^{(1)}, P_{pub}, \mathcal{H}_1, \mathcal{H}_2)$.

- **Keygen**: given an identity $ID \in \{0,1\}^*$, the PKG computes $Q_{ID} = \mathcal{H}_1(ID) \in \mathbb{G}_1$ and $d_{ID} = sQ_{ID} \in \mathbb{G}_1$ that is given to the user as a private key.

- **Encrypt**: to encrypt a message $m \in \mathcal{M}$, the sender uses the recipients public identifier $Q_{ID} = \mathcal{H}_1(ID) \in \mathbb{G}_1$ as follows:

  1. she picks a random $r \in \mathbb{Z}_q^*$
  2. she computes $g_{ID} = \hat{e}(Q_{ID}, P_{pub}) \in \mathbb{G}_2$.

  The ciphertext $c = \langle rP_0^{(1)}, m \oplus \mathcal{H}_2(g_{ID}^r) \rangle$ is sent to the receiver.

- **Decrypt**: given a ciphertext $c = \langle U, V \rangle$, the receiver uses her private key d_{ID} to decrypt by computing $m = V \oplus \mathcal{H}_2(\hat{e}(d_{ID}, U))$.

The correctness of the scheme (i.e. the fact the decryption algorithm is inverse of the encryption algorithm) is achieved since $U = rP_0^{(1)}$ and

$$\hat{e}(d_{ID}, U) = \hat{e}(sQ_{ID}, rP_0^{(1)}) = \hat{e}(Q_{ID}, P_{pub})^r = g_{ID}^r.$$

Since its inception, much research has been conducted to investigate applications and extensions of the first efficient identity-based cryptographic scheme due to Boneh and Franklin [BF01]. (See for instance [SOK00, Sma03b, CC03, Hes02, CHM+02, CHSS02, Sma03a, Boy03, LQ03a, LQ04, BZ04, GS02, Y. 02, CZK03] for specific schemes, and [DBS04, Gag03, Pat02, PP03] for surveys on ID-based and pairing-based cryptography.)

Mediated IBE

One extension of Boneh and Franklin's schemes is the mediated ID-based encryption scheme of Libert and Quisquater [LQ03a]. This scheme has a related signature scheme. Introduced by Boneh et al. [BDTW01], mediated cryptography is predicated on the idea that each user's private key can be split into two random shares, one of which is given to the user and the other to an online entity called a *security mediator* (*SEM*). Thus, any decryption or signature must be performed as a cooperation

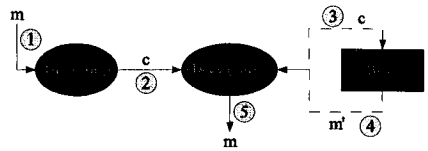

Figure 2.1: The SEM Model. The decryption of a ciphertext c derived from a message m is performed as follows: the decryptor sends c to her SEM; the SEM partially decrypts c into m' and sends m' back to the decryptor; the decryptor completes the decryption of m' and obtains m.

between a user and her associated *SEM* (cf. Figure 2.1). Moreover, *SEMs* are typically associated with a small number of local users and can be instantly instructed to revoke any of their associated users'decryption or signature privileges. Consequently, the *SEM* architecture allows fine-grained and fast[5] revocation of user security capabilities. Moreover, this architecture allows a system's *PKG* to delegate its decryption- and signature-related duties to the *SEMs*. Furthermore, the schemes of Libert and Quisquater allow *SEMs* to be semi-trusted entities, in the sense that their compromise only affects the users associated with them[6]. Recently, Baek and Zheng [BZ04] have improved the encryption scheme of Libert and Quisquater [LQ03a] by making it secure against insider chosen ciphertext attacks. More precisely, the newer scheme ensures that the compromise of a user's private key share does not compromise the confidentiality of plaintexts encrypted for that user, provided the associated *SEM*'s private key share is not compromised. However, one limitation of the above mediated schemes is their inefficiency at handling hierarchically structured user communities. Indeed, the schemes do not (and are not meant to) provide mechanisms for *SEMs* to generate the private key shares of other (children) *SEMs*. Consequently, the *PKG* of a *SEM* architecture must generate the private key shares of all *SEMs*. This necessity imposes both a high computational burden on the *PKG* and a high communication cost between the *PKG* and the *SEMs*. Therefore, it would be desirable to have a mediated encryption scheme which enables a root *PKG* to generate the private key shares of a limited number of *SEMs* who, in turn, generate the private key shares of other *SEMs*, and so on. Such a scheme would efficiently handle hierarchically

[5]Note that, by *fast revocation*, we mean *revocation without delay* after an authoritative entity becomes aware that privileges must be revoked. By way of comparison, note that technologies such as *Certificate Revocation Lists* and their variants (e.g. delta-CRLs) [Koc98, MAM+99] do not provide such an *fast* revocation of user security capabilities.

[6]This is in contrast with the *RSA*-based identity-based mediated scheme of Boneh et al. [BDT04], whose security completely breaks down when a *SEM* is compromised. The break down is due to the fact that a *SEM* compromise reveals a complete user public-private *RSA* key pair, which, in conjunction with a system-wide *RSA* modulus, allows to compute the factors of the system-wide *RSA* modulus, and therefore any other user's private key, from the other user's public key.

structured user communities.

Threshold IBE

The concept of threshold secret sharing was introduced by Shamir [Sha79, Bla79], in 1979. Secret sharing refers to the distribution of secret information shares to a group of entities, in such a way that a pre-determined number of secret shares is required for the corresponding distributed secret to be reconstructed. Secret sharing has given birth to many cryptographic schemes which use this concept to distribute decryption or signing keys among cryptographic servers, in such a way that a quorum of servers must cooperate in order to complete decryption or signing procedures. This allows one to increase the reliability and security of electronic systems which process confidential information [Ped92, Des94, Des97, CG99, LP01, Sim88, SM95].

Shamir's secret sharing scheme works as follows[7]. Let $\mathbb{G}_1$ be an Abelian additive group and $s \in \mathbb{G}_1$ be some secret element. In order to share s among a group $\{SER_i\}_{i=1}^n$ of $n \geq 2$ servers in such a ways that $t \geq 2$ of them is required for s to be recovered, define first the polynomial $F_s(u) = s + \sum_{i=1}^{t-1} r_i u^i$, where each r_i is picked uniformly at random in $\mathbb{G}_1$. Then, give $S_i = F_s(i)$ to SER_i, as a secret share $(i = 1, \cdots, n)$. Now, let $\Phi \subseteq \{1, \cdots, n\}$ such that $|\Phi| \geq t$. Then $\{SER_i\}_{i \in \Phi}$ forms a quorum of servers. Thus, to recover s this quorum can jointly compute $s = \sum_{i \in \Phi} c_{0i}^{\Phi} S_i$, where $c_{ui}^{\Phi} = \prod_{j \in \Phi, j \neq i} \frac{u-j}{i-j}$ is the called the general Lagrange coefficient of the set Φ.

It is interesting to note that mediated cryptography is in fact a special case of threshold cryptography. In a (γ, η)-threshold decryption scheme, any decryptor must interact with $\gamma - 1$ out of $\eta - 1$ other decryption entities, instead of only one *security mediator*. Hence, a mediated scheme is a $(2, 2)$-threshold scheme. Baek and Zheng [BZ04] recently described the first pairing-based identity-based threshold encryption/decryption scheme offering semantic security against adaptive chosen ciphertext attacks, in the random oracle model, assuming the difficulty of the bilinear Diffie-Hellman problem. However, the scheme is not (designed to be) suitable for the hierarchical setting (as the above mediated scheme, which is in fact derived from it). More precisely, no procedure is described for decryption servers to compute the private keys of other (children) decryption servers.

[7]Note that in Shamir's original secret sharing scheme, the coefficients of $F_s(u)$ belong to the reals. Hence, the version which is presented here is a *variant* of Shamir's original scheme.

2.3 Cryptography for Large User Communities

As mentioned earlier, certificate-based PKIs are cumbersome when used to support user-based[8] encryption and signature in large distributed user communities (such as university or corporate networks consisting of more than 10,000 users). However, two main classes of certificate-based cryptographic solutions have been suggested to support encryption in large user communities. These two classes of solutions support control on *objects* which can be accessed by *multiple* users having sufficient security privileges. Consequently, these solutions are not suitable for user-based encryption since they compromise user-privacy. The first class of solutions consists of using trusted cryptographic agents known as key-release agents (KRAs), in order to support object-based access controlled decryption. The second approach consists of associating users with hierarchically-structured sets, and of describing mechanisms enabling users to decrypt ciphertexts intended for selected sets, based on the security privileges of these users. Both classes of solution are reviewed next, followed by a review of *ID-based* cryptographic solutions facilitating *user-based* encryption and signature in large user communities.

Key Release Agents

The KRA scheme [FW94] works as follows. Let $\mathcal{A}$ be an encryptor and $\mathcal{B}$ be an intended decryptor associated with a KRA denoted by $K_{\mathcal{B}}$. To encrypt a message m, $\mathcal{A}$ generates a random symmetric key k, encrypts m with k (thereby obtaining c_m), encrypts k with the public key of $K_{\mathcal{B}}$ (thereby obtaining c_k), and sends a data structure $c = (c_m, c_k, r)$ to a public repository or a recipient $\mathcal{B}$ (where r is the identifier list of privileges required to access m). Then, $\mathcal{B}$ picks c, contacts $K_{\mathcal{B}}$, $K_{\mathcal{B}}$ verifies that $\mathcal{B}$ currently has r, and, if so, decrypts c_k, and sends k to $\mathcal{B}$ who uses the random key to decrypt c_m. The KRA scheme is flexible with respect to access control models. In particular, this scheme works for time-based and group-based access control (depending on the types of privileges included in the encryption data structures). Moreover, the KRA scheme can be used with certificate-based public-key cryptography as well as ID-based cryptography. Note, however, that the KRA scheme assumes that key release agents be fully trusted entities which never divulge (or lose control of) their private keys, and which can decrypt all ciphertexts intended for their associated users. Hence, the KRA scheme compromises user privacy, and encourages system designers to limit the number of users associated with each KRA (in order to limit the effect of KRA compromise). Thus, when utilized for user-based encryption-based access

[8] By *user-based encryption*, it is hereby meant encryption to individual users (as opposed, e.g., to encryption to *sets* of users). Likewise, *user-based signature* refers to signature on behalf of individual users.

control, the KRA scheme is not scalable to large and distributed user communities. However, the KRA scheme is suitable (and was actually suggested) for object-based access controlled decryption.

Hierarchical Certificate-based Schemes

The second class of certificate-based public-key encryption schemes proposed for object-based access control, in large user communities, are meant to support (variants of) *role-based access control* (RBAC).

Role-Based Access Control [FSG+01] is a methodology to manage the access to resources of a system. Efficiency in the management of user security privileges is achieved, in RBAC, by grouping users who have identical privileges in classes called *roles*. Roles can be organized in a hierarchy, and may be activated and deactivated. Thus, access privileges can be given only when required, and it is possible to require that users deactivate certain security privileges before exercising others.

A detailed review of these schemes are presented in Appendix A. As a summary, we simply note that some of these schemes use algebraic techniques, while others (significantly more efficient than the former ones) use one-way hash functions.

Hierarchical ID-based Cryptography

Within the framework of ID-based cryptography, the class of schemes which have been suggested to support user-based access control, in large and distributed user populations, are called hierarchical ID-based cryptographic schemes.

The concept of hierarchical ID-based cryptography (HIBC) was suggested by Horwitz and Lynn [HL02], shortly after the description of Boneh and Franklin's IBE scheme. HIBC schemes provide a method for a central trusted authority to delegate the computation and delivery of user decryption keys to lower-level authorities. More precisely, a *root PKG* generates a unique secret key for each of its *children PKGs*, who in turn follow the same procedure to generate a unique secret key for each of their *children PKGs*, and so on. Such a procedure allows any *PKG* to decrypt ciphertexts addressed to the users associated with any of its *descendant PKGs*. Moreover, only the top-level (system) parameters of a hierarchy are needed for encryption. Consequently, the scheme is scalable to large and distributed structured communities of users.

Horwitz and Lynn [HL02] described the first hierarchical ID-based encryption scheme: a 2-level scheme allowing a sufficient number of 2^{nd}-level users to collude in order to compute the secret key of their 1^{st}-level PKG. (Recall that, in HIBE, PKGs should be able to generate the keys of their descendants only). The first HIBE

scheme (henceforth called GS-HIBE) with total collusion resistance on a polynomially bounded number of levels was introduced by Gentry and Silverberg [GS02], in 2002. GS-HIBE's security was proven in the random oracle model. Boneh and Boyen [BB04] recently presented a HIBE scheme (henceforth called BB-HIBE) whose security is shown in the standard model [BR93]. Both GS-HIBE and BB-HIBE exhibit ciphertext lengths and decryption times which grow linearly in the depth of ciphertext recipients in the underlying hierarchies. Moreover, these two schemes feature key generation times which grow linearly in the hierarchical depth of users. In a very recent independent work, Boneh et al. [BBG05] presented a HIBE scheme (henceforth called BBG-HIBE) with constant size ciphertexts and constant decryption time. One feature of BBG-HIBE is that its key generation computational cost is linear with respect to the hierarchical depth of users. Moreover, the length of BBG-HIBE's keys is linear in the depth of the underlying hierarchy[9]. Boneh et al. [BBG05] also presented a scheme whose key length is sublinear in the hierarchical depth of its user, but this scheme does not yield constant size ciphertexts. Gentry and Silverberg [GS02] and Chow et al. [CHYC04] described HIBS schemes, but the signature length and the computational cost of signature verification in these schemes grow linearly with respect to the hierarchical depth of signers. Hence these HIBS schemes are not computationally efficient.

In this section, we have presented an overview of past research on the use of (certificate-based and ID-based) public-key cryptography in large and distributed user communities. §2.4 reviews background research on threshold attribute-based encryption.

2.4 Threshold Attribute-Based Encryption

Since the first threshold attribute-based encryption scheme was only recently suggested [SW05], our literature review on thABE will be short.

Threshold Attribute-Based Encryption

Sahai and Waters [SW05] recently introduced the first threshold attribute-based encryption scheme. thABE schemes view identities as sets of descriptive attributes. Let ω be such a set, d be a threshold value, and c be a ciphertext intended for ω. Let also ω' be the identity of a user $\mathcal{U}$. Then, $\mathcal{U}$ must have decryption keys associated with a d-element subset of ω in order decrypt c. (In other words, $|\omega \cap \omega'| \geq d$ must

[9]More precisely, BBG-HIBE's key length *decreases* linearly in the depth of the underlying hierarchy.

be satisfied in order for $\mathcal{U}$ to decrypt c.) Thus, if attributes represent biometric features, thABE can be used to support biometric-based access control (as explained in Chapter 1).

Fuzzy IBE

thABE schemes form a subclass of fuzzy ID-based encryption (FIBE) schemes [SW05]. As thABE schemes, FIBE schemes view identities as sets of descriptive attributes, and use attributes to encrypt messages. More precisely, a user $\mathcal{U}$ with identity ω' is able to decrypt a ciphertext encrypted with identity ω if and only if ω and ω' are within a certain distance from each other, as stipulated by a specifiable metric. One such metric is the *set-overlap* distance, which forces each decryptor of any given ciphertext c to have keys corresponding to a sufficiently large subset of the identity used to obtain c. When a FIBE scheme uses the *set-overlap* distance, then this scheme is called a *threshold attributed-based encryption* (thABE) scheme.

Comparison with Threshold IBE

thABE schemes distinguish themselves from *threshold IBE* (thIBE) schemes [BZ04, BDT02, LQ03a], by the following two features: first, thIBE schemes view each identity as one string (instead of a set of attribute strings); second, thIBE schemes require one decryptor to interact with a threshold number of servers in order to complete decryption procedures (such a network interaction is not needed in the case of thABE).

Collusion Resistance

Noteworthy is the fact that thABE schemes should resist collusion attacks whereby users having various attribute keys attempt to pool their keys in order to decrypt ciphertexts which none of them could decrypt on her own. This means that private attribute keys should be "personalized" while public attribute keys should not be tied with users, so that thABE schemes be scalable to large user populations.

Yao et al. [YFDL04] proposed a collusion-resistant ID-based encryption scheme which encrypts to multiple hierarchical identities, but this scheme is computationally expensive (the number of pairings in the decryption procedure grows linearly with the number of hierarchical identities used to encrypt a message.)

Other Work on Biometric-based Access Control

Much research dealing with the use of biometrics to derive cryptographic secret keys has also been conducted [Boy04, DRS04, JW99, MRL+02, MRLW01, MRW99]. In these systems, as pointed out by Sahai and Waters [SW05], the capture of one's

biometric reading enables full impersonation of the corresponding person. On the contrary, FIBE allow biometric measurements to be public, without compromising the confidentiality of ciphertexts associated with the corresponding biometric features. In fact, identities used to encrypt messages are also made public to chosen encryptors.

2.5 Encrypted Message Authentication

This section reviews past research related to encrypted message authentication.

Signcryption with Public Ciphertext Authenticity

Recall that *Signcryption* [Zhe97] (see also [RBBK01]) is the process whereby a given message is both encrypted and digitally signed in one logical step, in order to provide message confidentiality, signer authentication, and signature non-repudiation[10] at the same time. Signcryption schemes should produce ciphertexts (henceforth referred to as *signcryptexts*) whose length is comparable to the ciphertext size of the underlying encryption algorithms. Moreover, the computational cost of signcryption should be lower than the added costs of the underlying signature and encryption procedures. In terms of security [Boy03], signcryption schemes should be *semantically secure* (for confidentiality) and *existentially unforgeable* (for authentication and non-repudiation).

Another desirable property of signcryption schemes is *public ciphertext authenticity* [CYHC03, GLZ99]. This feature enables third parties to ascertain, without the help of any intended recipient, both the identity of any given signcryptext sender and the validity of this signcryptext (i.e. the fact that the signcrypted message was validly signed and ciphered by the same party.) Thus, public ciphertext authenticity (PCA) enables public verification of signatures without compromise of the confidentiality of signcrypted messages. Such a feature is useful for applications in which valid signcrypted messages must be either routed from one point to another, stopped (e.g. by a firewall), or distinguished from fake signcryptexts. For instance, PCA is useful when electronic mail (email) needs to be confidentiality-protected and communication systems need to combat denial-of-service (DoS) attacks whereby mail routers are flooded with fake signcryptexts. Likewise, PCA is useful in applications which store signcryptexts on public file servers. Indeed, these servers could be flooded with fake signcryptexts, and this could cause significant delays (and confusion) in the retrieval of protected (i.e. signcrypted) valid information.

[10]Signature non-repudiation is the property whereby the identity of a signer cannot be denied, given a valid signature and publicly available cryptosystem parameters.

Signcryption with Forward Security

Another desirable property of signcryption schemes is *forward-security* (FS). This feature prevents the theft of any user's private key from compromising the confidentiality of signcrypted messages issued by this user.

Benefits of Hierarchical ID-based Signcryption

Chapter 6 is concerned with *hierarchical ID-based* signcryption, because of the convenience of use which ID-based schemes provide, and because of the scalability that hierarchical (ID-based) schemes offer.

Certificate-Based and ID-based Signcryption

The concept of public-key signcryption was introduced by Zheng [Zhe97], in 1997. The motivation is to perform both signature and encryption in a single logical procedure, in order to obtain message confidentiality, signer authentication and signature non-repudiation, more efficiently than by first signing a message, and by then encrypting the resulting signature. Various efficient signcryption schemes were first proposed [Zhe98, ZI98, SZ00, Zhe01, YL01], none of which were ID-based ones.

Combining a variant of Hess's ID-based signature scheme [Hes02], and a simplified version of Boneh and Franklin IBE scheme [BF01], Malone-Lee [ML02] described the first ID-based signcryption scheme. The resulting scheme is not, however, semantically secure, since signcrypted messages do not hide the signature of the original message. Nalla and Reddy [NR03] then suggested a signcryption scheme which does not allow parties other than an intended recipient to verify the validity of the signature associated with each signcrypted message. In other words, Nalla and Reddy's signcryption scheme does not have the so called public ciphertext authenticity property [CYHC03]. Recently, Libert and Quisquater authored various signcryption schemes [LQ03b] which do not simultaneously provide public ciphertext authenticity and a property called *forward security*. Chow et al. [CYHC03] presented an ID-based signcryption scheme featuring both forward-security and public ciphertext authenticity (see also [MB04]).

Currently, the most security-feature-rich ID-based signcryption scheme is due to Boyen [Boy03][11]. This scheme provably and simultaneously provides message confidentiality (semantic security with respect to adaptive chosen ciphertext attacks), signature non-repudiation (existential unforgeability with respect to adaptive chosen message attacks), ciphertext anonymity, and public verifiability (i.e. the ability for a

[11] More precisely this scheme can be *directly derived* from Boyen's multipurpose ID-based signature-encryption scheme [Boy03].

third party to determine, with the help of signcryptext intended recipient, whether a given signcrypted message provides a valid signature of a given message.) However, Boyen's scheme [Boy03] does not provide public ciphertext authenticity.

Note also that all the above (ID-based) signcryption schemes are not suitable for hierarchical settings or for very large user communities. The only known *hierarchical* ID-based signcryption scheme is due to Chow et al. [CYHY04]. However, this scheme does not provide PCA. Moreover, Chow et al.'s scheme [CYHY04] inherits efficiency limitations of Gentry and Silverberg's hierarchical ID-based encryption and signature schemes [GS02]. In particular, the length of signcrypted messages is the sum of the hierarchical depths of its signer and its intended recipient. Libert and Quisquater [LQ04] also suggested a signcryption scheme which combines ID-based and certificate-based cryptography in order to handle large communities of users forming multiple groups (domains). However, this hybrid scheme has the following limitations: first, it requires each signcryptext to include a public-key certificate associated with its sender's PKG – thereby inducing higher space requirements and forcing signcryptext recipients to verify the validity of each sender PKG's public-key certificate; second, it does not handle hierarchical key generation (i.e. the ability of PKGs to delegate, to other (hierarchically lower) PKGs, the task of user key generation). Consequently, the hybrid scheme is not suitable for hierarchically-structured or large distributed user communities. As a summary, we have seen that no efficient hierarchical ID-based signcryption (HIBSC) is known and the only known HIBSC does not provide public ciphertext authenticity.

2.6 Hierarchical Group Signature

The aim of this section is to review past research related to hierarchical group signature.

Group Signature

Group signature (GS) schemes are cryptographic primitives enabling each member of a given group to sign on behalf of the group, in such a way that only a designated group leader is able to identify the issuer of any given valid group signature.

Since the introduction of group signatures by Chaum and van Heijst [CvH91], many group signature schemes have been suggested [CP94, CS97, Pet97, Cam97, TJ98, AT99, Pop00]. Most of these schemes, however, are inefficient for large groups, due to the fact that their group public key size and group signature length linearly increase with the size of groups. Camenish and Stadler [CS97] introduced the first efficient group signature scheme whose group public key size and group signature

length are constant. The state-of-the-art group signature scheme is the provably coalition-resistant secure scheme described by Ateniese et al. [ACJT00]. Recently, Bellare et al. [BMW03] established two properties which imply all known reasonable security features that a group signature scheme can be expected to have. These properties (full-anonymity and full-traceability) were then used to prove the security of subsequent schemes, including the VLR scheme of Boneh and Shacham [BS04].

ID-based Group Signature

Park et al. [PKW97] proposed the first ID-based group signature scheme. This scheme – whose insecurity was shown by Mao and Lim [ML98] – was inefficient for large groups (due to the same reasons as those described above for non ID-based group signature schemes). Tseng and Jan [TJ98] presented an alternative ID-based group signature scheme, but its signatures were shown to be forgeable [JKL99], and the scheme was proved not to be coalition-resistant [Joy99]. It was also suggested, by Castellucia [Cas02], to use random identifiers to generate group member public keys in ID-based group signature schemes. This is however not appropriate, since group leaders (which are instantiated by *PKG*s) can then misattribute valid group signatures in order to frame honest group members. In fact, apart from [CZK03], all the proposed ID-based group signature schemes suffer from this weakness.

Hierarchical Group Signature

Hierarchical group signature (HGS) schemes generalize their GS counterparts by dealing with hierarchically structured groups (i.e. groups whose subgroups are hierarchically structured). The precise security requirements of HGS schemes are being defined, and this is therefore a focus of Chapter 7. However, HGS schemes always assume the existence of an underlying rooted hierarchy of subgroup leaders, each of which manages a set of signers. One class of HGS schemes (henceforth called *HGS-Class-1*) deals with cases in which group signatures completely hide the internal hierarchical structure of a given group. In these cases, the challenge is to enable signers to issue signatures on behalf of the whole group, and to provide a mechanism whereby, given a valid group signature σ, each subgroup leader is only able to identify either which of its subgroup members issued σ, or which of its subgroup members (indirectly) leads the subgroup of σ's issuer. A second class of HGS schemes (henceforth called HGS schemes with public subgroup identification (PSI)) deals with cases in which the group structure of a given group is public[12], but the identity of each

[12]By *public group structure*, it is hereby meant that both the identity of all subgroup leaders and the structural relationship between these leaders are available to signature verifiers. The identity of signers is not known by verifiers.

member of each subgroup remains undisclosed. In those cases, the design goal is to enable signers to sign on behalf of their subgroup, and to provide a mechanism whereby the leader of any given subgroup is the only party able to identify the issuer of any valid group signature issued on behalf of the given subgroup. In other words, HGS schemes with PSI support HAA.

HGS and Related Primitives

Trolin and Wickstrōm [TW05] recently presented the first formal treatment of hierarchical group signatures. Their work focuses on what we refer to as *HGS-Class-1* group signature schemes. Trolin and Wickstrōm describe a provably secure *HGS-Class-1* scheme, but the scheme is suboptimal (for reasonable security parameters, producing group signature requires a few hundred exponentiations). Group signatures for hierarchical multigroups had already been considered by Kim et al. [KPW97], but the scalability of Kim et al.'s scheme is hampered by the fact that both group managers and signers must register in one central authoritative party before signers are able to issue group signatures. Moreover, Kim et al.'s scheme does not feature a hierarchical identification procedure for group managers, which makes the scheme non adequate for HAA applications. Anonymous authentication schemes were presented by Kilian and Petrank [KP98], featuring identity escrow (i.e. a designated party is able, under special circumstances, to identity the issuer of anonymous authenticating tokens). However, these schemes do not provide hierarchical delegation of secret key generation (as HGS schemes do). Anonymous authentication schemes with subset queries were also proposed (cf. [CDS94, SCPY94, FFS88]). These schemes enable an authenticating party to determine whether the issuer of a given authenticating token belongs to a subset of the system users. Such a subset may be the subset of another one – thereby yielding a hierarchical setting. However, [CDS94], [SCPY94], and [FFS88] do not provide identity escrow. Boneh and Franklin [BF99] presented the first anonymous authentication scheme with subset queries, identity escrow, and signer revocation capability. However, the work required by provers (e.g. signers) and verifiers, in this scheme, is linear with respect to the number of subsets. This is suboptimal. For instance, when subsets are structured as a rooted tree, the work required by provers may be expected to be constant, and the work required by verifiers to be linear with respect to the depth of a specific subset (i.e. logarithmic in the total number of subsets). This is what our proposed HIBGS scheme achieves.

Chapter's Summary

In this chapter, we have reviewed past research related to the use of efficient ID-based cryptography in large user communities. In particular, we have reviewed work on threshold attribute-based encryption, hierarchical ID-based encryption and signature, hierarchical ID-based signcryption, and hierarchical group signature. In the next chapter, we present our first main contribution, namely the design of a provably secure threshold attribute-based encryption scheme, as a solution to the problem of generating efficiently-usable decryption keys from non-repeatable information.

Chapter 3

Threshold Attribute-Based Encryption

This chapter explains how to generate efficiently-usable[1] decryption keys based on selected attributes that are extracted from non-repeatable information. In order to do so, we describe an efficient threshold attribute-based encryption (thABE) scheme.

thABE was recently proposed to support biometric-based access control (BBAC). However, the key generation algorithm of any thABE scheme can be used to build signature schemes which support biometric-based authentication (cf. Chapter 1). Practical biometric-based applications impose the following constraints on the design of thABE schemes: first, the private keys of a suitable thABE scheme must be efficiently usable for decryption; second, this scheme must prevent colluding users from being able to decrypt ciphertexts which none of them could decrypt; third, the designed scheme must provide a mechanism whereby encryptors can, *at encryption time*, specify multiple sets of attributes with their corresponding threshold values.

Note that similar constraints could be formulated for signature schemes associated with the key generation algorithm of any thABE scheme. However, the topic of threshold attribute-based signature schemes is left for future research.

To the best of our knowledge, no thABE scheme is known that simultaneously satisfies the aforementioned requirements. This chapter describes an efficient and collusion-resistant thABE scheme featuring dynamically-specifiable threshold values. The proposed scheme is proven secure in the random oracle model, and its efficiency and flexibility are compared with Sahai and Waters' thABE scheme.

[1]Recall that *efficiently-usable* decryption keys refer to keys which can be used to decrypt efficiently.

3.1 Introduction

Cryptography has long been suggested as a method to support information access control. The idea is to encrypt confidential documents according to sets of attributes, and to give the corresponding attribute decryption keys to users who have the corresponding attributes. Thus, it is expected that only authorized users can cryptographically access (decrypt) protected documents. In the following section, we present issues which must be addressed when cryptography is used to support biometric-based access control (BBAC).

Practical BBAC Constraints

In BBAC, one's biometric identity is often composed of multiple sets of attributes. This is true for iris scans [Dau04] and fingerprints [JRP01]. Moreover, the total number of attributes used to represent biometric identities is often relatively large. For instance, Daugman's iris recognition algorithm features 249 degrees of freedom (i.e. independent characteristics) [Dau04]. Likewise, Jain et al.'s fingerprint scanning algorithm [JRP01] extracts 648 features. Consequently, practical cryptographic BBAC applications call for the use of algorithms which enable users to encrypt confidential documents with respect to multiple and varying sets of attributes.

Moreover, the devices used to read biometric features have varying degrees of accuracy (cf. §4.3 of [SW05]). Hence, encryption based on biometric features is associated with a degree of accuracy which is determined at the time of biometric reading. Consequently, practical cryptographic BBAC applications call for the use of mechanisms which enable encryptors to specify a degree of accuracy at encryption time.

Threshold Attribute-Based Encryption

Recently introduced by Sahai and Waters [SW05], threshold attribute-based encryption (thABE) has been suggested as a cryptographic primitive suitable for both BBAC and RBAC. More precisely, let d be a threshold value, and c be a ciphertext intended for an identity ω (where ω is a set of descriptive attributes). Let also ω' be the identity of a user $\mathcal{U}$. Then, $\mathcal{U}$ must have decryption keys associated with a d-element subset of ω in order decrypt c. (In other words, $|\omega \cap \omega'| \geq d$ must be satisfied in order for $\mathcal{U}$ to decrypt c.) Thus, if attributes represent roles or biometric features, thABE can be used to support RBAC and BBAC.

Concrete Example of thABE Application to BBAC

In BBAC, each user is identified by sets of attributes representing features of her biometric identity (such as her iris or her thumb). Since biometric measurements are noisy, the use of thABE enables the use of biometrics to authenticate human users[2], and thereby control access to information addressed to these users. For instance, a user $\mathcal{A}$ could go to a Driver Licensing Agency $\mathcal{D}$, and identify herself via an iris scan, *under the ongoing surveillance of a trained agent.* $\mathcal{D}$ could then use this scan to encrypt $\mathcal{A}$'s information (e.g. an annual driver's license), when this information needs to be securely sent to $\mathcal{A}$ (e.g. via the Web, or via storage of ciphertexts on a smart card presented by (a representative of) $\mathcal{A}$). In order to obtain her biometric private keys, $\mathcal{A}$ would have to go in person to a trusted third party (e.g. a state agency) which would deliver keys via the same authenticating procedure as that used by $\mathcal{D}$. (Note that such a procedure *significantly reduces the possibility of user impersonation* in the process of obtaining attribute keys.) $\mathcal{A}$ could then decrypt ciphertexts addressed to her, using a thABE scheme.

RBAC Applications

thABE can also be used for role-based access control (RBAC) [FSG+01] (cf. §2.3). Suppose for instance that the CEO of an organization $\mathcal{O}$ wants to send a strategic plan to all the vice presidents of $\mathcal{O}$. The CEO could use a thABE scheme and encrypt the strategic plan under the attributes "*employee-of-*$\mathcal{O}$", "*vice-president*", and "*not-on-vacation*". If the threshold parameter of the thABE scheme is set to 3, then only the vice presidents of $\mathcal{O}$ who are not on vacation will be able to decrypt the ciphered strategic plan. We emphasize that, in practice, role-based access control often requires the possession of multiple specific roles in order to access confidential documents. Therefore, thABE with dynamically-specifiable threshold values offers a better (i.e. space and time efficient) alternative to the sequential encryption of documents under the attributes required to access these documents. Indeed, ciphertexts issued by the thABE scheme presented by Sahai and Waters are shorter than ciphertexts obtained by sequentially applying Boneh and Franklin's identity-based (ID-based) encryption (IBE) scheme [BF01].

Design Challenges

Designing efficient and flexible thABE schemes is not trivial. Indeed, if New-thABE is such a scheme, then the following challenges must be simultaneously addressed.

[2]While biometric measurements are noisy, we assume that it is possible to determine, from a biometric reading, whether a biometric feature has been recognized.

C1 New-thABE should be resistant to collusion attacks whereby users having various attribute keys attempt to pool their keys in order to decrypt ciphertexts which none of them could decrypt on their own. This means that private attribute keys should be "personalized" while public attribute keys should not be tied with users, so that New-thABE be scalable to large user populations.

C2 Encryption for an identity ω should target each of the possible sufficiently large subsets of ω. If ω has ℓ elements and d is a given threshold value, then there are $\frac{\ell!}{d!(\ell-d)!}$ possible combinations. However, the cost of encryption should remain low (i.e. ideally constant, but at most linear with respect to d).

C3 If d is a threshold value associated with a ciphertext c, then a user with identity ω should be given private keys in such a way that at least d of them must be jointly used in order to decrypt c. This is challenging to achieve because private keys are given before encryption time. Hence, the use of a mere secret sharing scheme at decryption-key granting time is not adequate.

C4 Encryptors should be able to select attributes from multiple sets whose threshold values are different.

With respect to C3, remark that the decryption procedure of Sahai and Waters's second scheme [SW05] requires $2d$ pairing (i.e. expensive) operations, where d is a system-wide threshold parameters. (Note that Sahai and Waters presented two schemes in [SW05]. However, only the second scheme is scalable when the number of attributes grows. Moreover, only the second scheme features a small set of certified public parameters. Consequently, the phrase *Sahai and Waters'scheme* will henceforth refer to Sahai and Waters'*second* scheme.) One is interested in a scheme whose decryption algorithm involves a small constant number of pairing computations. Furthermore, remark that Sahai and Waters suggested two methods to address C4. First, it was suggested to use different thABE systems with different threshold values. Second, it was proposed to use a maximal threshold value for all ciphertexts, and to use dummy attributes (whose secret keys are given by default to all decryptors) in order to decrease the effective threshold value associated with a given ciphertext. The first proposed option is not necessarily convenient, but most importantly, it increases the length of ciphertexts associated with multiple sets of attributes (by a factor equal to the number of sets). The second proposed option is not necessarily adequate either, as illustrated by the following example. Suppose that $d_{max} = 4$, and that one wants to encrypt a message m for an identity $\omega = \{\alpha_1, \alpha_2\}$ with threshold value $d = 2$. Let δ_1 and δ_2 be two dummy attributes. If m is encrypted for ω with δ_1 and δ_2, then the recipient can decrypt the ciphertext without having the secret keys associated with α_1 and α_2. Moreover, if m is encrypted for with ω with δ_1, then the recipient only needs to have the secret key associated with α_1 or α_2.

Thus, while Sahai and Waters addressed C1 and C2 simultaneously, designing a scheme which efficiently addresses C1 through C4 remains an open question. The goal of this chapter is to address each of the aforementioned challenges.

Contributions

The main contribution of this chapter is to present a flexible and collusion-resistant thABE scheme featuring significantly lower computational requirements than Sahai and Waters'scheme (henceforth referred to as SW-thABE). More precisely, let d be the predefined threshold value associated with SW-thABE. Then the proposed scheme's decryption procedure requires only 2 pairing computations, compared with $2d$ pairings in the case of SW-thABE's decryption algorithm. Moreover, the proposed thABE scheme (henceforth referred to as New-thABE) is flexible because it allows to specify, at encryption time, various sets of attributes with their associated threshold values. Furthermore, if t denotes the number of attributes used to encrypt a message, an optimization of New-thABE is proposed which drastically reduces its ciphertext length, from $O(t)$ to $O(1)$, when decryptors must have *all* the attributes of a given set. Moreover, we note that, when a ciphertext's threshold value is maximal, New-thABE provides the first efficient collusion-resistant IBE scheme.

We examine the security of New-thABE in the random oracle model [BR93], and show that this scheme can be extended (via Fujisaki-Okamoto padding [FO99]) into a scheme that provides semantic security against adaptive chosen ciphertext attacks, in the random oracle model [BR93], if the bilinear Diffie-Hellman problem is intractable. For comparison, note that SW-thABE achieves a weaker security result (i.e. semantic security against selective-ID attacks[3]), in the standard model [BR93].

Outline

The sequel is organized as follows: §3.2 reviews standard terminology concerning thABE, its security, and related number theoretic assumptions. §3.3 describes our proposed thABE scheme, and §3.4 discusses its computational requirements, in comparison with SW-thABE. §3.5 summarizes the security guarantees of our scheme, and §3.6 concludes the chapter.

[3]Selective-ID attacks model attackers which choose, in advance, the target identity they wish to attack.

3.2 Preliminaries

In this section, we present fundamental definitions related to thABE schemes. Readers familiar with [SW05] may go to §3.3.

Identities

For the description of thABE schemes, each *identity* is viewed as a set ω of *attributes*. Each user with identity ω is given a set of private *attribute keys* each of which corresponds to an element of ω. These attributes keys are secretly granted by an entity known as the private key generator (PKG), upon careful inspection of users' identities (e.g. via surveillance of on-site biometric measurement).

Threshold Attribute-Based Encryption Scheme

Each thABE scheme is composed of four algorithms whose functions are described below:

1. `Setup`: Given a security parameter k, this algorithm is used by the PKG to return a tuple *params* of system parameters. These parameters include a description of the message space $\mathcal{M}$ and the ciphertext space $\mathcal{C}$, along with a secret piece of data s_0 called the *master secret key*. Other parameters are allowed, such as *attribute* parameters. Some parameters may be public (including those describing $\mathcal{M}$ and $\mathcal{C}$), while others remain secret (including the master secret key).

2. `Key Generation`: Given the scheme's public and private parameters, and an arbitrary identity ω, this algorithm returns a set D_ω of private keys corresponding to ω.

3. `Encrypt`: Given a threshold parameter d, a message $m \in \mathcal{M}$, the identity ω of an intended recipient, and the scheme's public parameters, this algorithm returns a ciphertext $c \in \mathcal{C}$ corresponding to m, ω and d.

4. `Decrypt`: Given a ciphertext $c \in \mathcal{C}$ (including a threshold parameter d and a target identity ω), the private key set $D_{\omega'}$ of a recipient, and the scheme's public parameters, this algorithm returns the message $m \in \mathcal{M}$ associated with c if $|\omega \cap \omega'| \geq d$.

Encryption and decryption must satisfy the following consistency constraint:

$$\forall\, m \in \mathcal{M}\ \text{Decrypt}(d,\, c,\, pubParams,\, D_{\omega'}) = m,$$
$$\text{if } |\omega \cap \omega'| \geq d \text{ and } c = \text{Encrypt}(d,\, m,\, \omega,\, pubParams).$$

3.3 Proposed thABE Scheme

The thABE scheme presented below (i.e. New-thABE) shows how to encrypt for two attribute sets whose threshold values can be dynamically specified by the encryptor. Encryption to polynomially many attribute sets could also be handled, using a similar methodology. However, for simplicity, we only describe the case in which there are two attribute sets.

1. `Instance Generator` (k). This procedure, denoted by IG, is a randomized algorithm which takes a security parameter $k > 0$, runs in $O(k)$, and outputs $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are two Abelian groups of prime order $q \geq 2^k$, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is an admissible pairing with respect to which $\mathbb{G}_1$ and $\mathbb{G}_2$ are Gap-Diffie-Hellman groups.

2. `Setup` (k): Given a security parameter $k > 0$, the *PKG*:

   (a) runs IG with input k and obtains $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$.

   (b) sets $\delta = 2$, and computes both $n = poly_1(k)$ and $\ell = poly_2(k)$, where $poly_1$ and $poly_2$ are polynomials over the positive integers; (δ is the number of attribute sets used for encryption, n is the message length, and ℓ is the total number of attributes;)

   (c) picks, randomly and uniformly[4], $P_0^{(1)}, P_0^{(2)} \in \mathbb{G}_1$ and $s_0 \in \mathbb{Z}_q^*$;

   (d) computes $P_{pub} = s_0 P_0^{(1)}$, $g = \hat{e}(P_0^{(2)}, P_{pub})$;

   (e) chooses two cryptographic hash functions $\mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $\mathcal{H}_2 : \mathbb{G}_2 \rightarrow \{0,1\}^n$.

   Let N_ℓ denote the set $\{1, \cdots, \ell\}$. The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = N_\ell^\delta \times \{0,1\}^{\delta \cdot \ell} \times \mathbb{G}_1^{\ell+1} \times \{0,1\}^n$. The system's public parameters (which must be certified) are $pubParams = (\delta, q, n, \hat{e}, P_0^{(1)}, g, P_{pub}, \mathcal{H}_1, \mathcal{H}_2)$, and the *PKG* keeps s_0 and $P_0^{(2)}$ secret. $params = (pubParams, s_0, P_0^{(2)})$.

3. `Key Generation` $(\omega, params)$: Let $\omega = \{i_j\}_{j=1}^t$ be the identity of a user $\mathcal{U}$, where each i_j is the index of an attribute. Let also ID_{i_j} be the string identifier of the i_j^{th} attribute, and $Q_{i_j} = \mathcal{H}_1(ID_{i_j})$ for $1 \leq j \leq t$. Then, the PKG picks $\mu \in_R \mathbb{Z}_q^*$, and secretly gives $\mathcal{U}$ her private key $D_\omega = (S_\omega, T_\mu, U_\mu)$, where $U_\mu = \mu P_0^{(1)}$, $T_\mu = s_0 P_0^{(2)} - \mu s_0 P_0^{(1)}$, and $S_\omega = \{D_{i_j}\}_{j=1}^t$ with $D_{i_j} = \mu \cdot Q_{i_j}$ for $1 \leq j \leq t$.

[4] In the sequel, we shall use the notation $x \in_R X$ to indicate that the element x is chosen uniformly at random from the set X.

4. `Encrypt` $(d_1, d_2, m, \omega_1, \omega_2, pubParams)$: Given two threshold parameters d_1 and d_2, a message $m \in \mathcal{M}$, $pubParams$, and the identities $\omega_1 = \{i_j\}_{j=1}^{t_1}$ and $\omega_2 = \{u_v\}_{v=1}^{t_2}$ of an intended recipient, the encryptor:

   (a) selects 2 random polynomials F_1, F_2, where
   $F_p(x) = P_{pub} + (\sum_{z=1}^{d_p-1} r_{(p,z)} x^z) P_0^{(1)}$
   (with $p = 1, 2$ and $r_{(p,z)} \in_R \mathbb{Z}_q^*$ for $1 \leq z \leq d_p$);

   (b) picks $r \in_R \mathbb{Z}_q^*$;

   (c) computes X, $(Y_{i_j}^{(1)})_{j=1}^{t_1}$, and $(Y_{u_v}^{(2)})_{v=1}^{t_2}$ as follows:
   $X = rP_0^{(1)}$, $Q_j = \mathcal{H}_1(ID_j)$, and $Y_j^{(p)} = r(F_p(j) + Q_j)\ \forall\ j \in \omega_p$ for $p = 1, 2$;

   (d) computes $\sigma_1, \sigma_2 \in \{0,1\}^\ell$ such that the e^{th} bit of σ_p is 1 if and only if $e \in \omega_p$ (with $p = 1, 2$);

   (e) computes $Z = m \oplus \mathcal{H}_2(g^r)$ and outputs
   $c = (d_1, d_2, \sigma_1, \sigma_2, X, (Y_{i_j}^{(1)})_{j=1}^{t_p}, (Y_{u_v}^{(2)})_{v=1}^{t_2}, Z)$.

5. `Decrypt` $(c, D_{\omega'}, pubParams)$: Let $c = (d_1, d_2, \sigma_1, \sigma_2, X, (Y_{i_j}^{(1)})_{j=1}^{t_1}, (Y_{u_v}^{(2)})_{v=1}^{t_2}, Z)$ be a given ciphertext computed for the identities ω_1 and ω_2. Let also $D_{\omega'} = (S_{\omega'}, T_\mu, U_\mu)$ be a recipient's private key, where $S_{\omega'} = \{D_j\}_{j \in \omega'}$. For $p = 1, 2$, let Φ_p be a d_p-element subset of $\omega_p \cap \omega'$. Then, the recipient outputs $m = Z \oplus \mathcal{H}_2(W)$, where $W = \hat{e}(\frac{1}{\delta}(\sum_{j \in \Phi_1} \phi_j^{\omega_1} D_j + \sum_{j \in \Phi_2} \phi_j^{\omega_2} D_j) - T_\mu, X)^{-1} \cdot \hat{e}(U_\mu, \frac{1}{\delta}(\sum_{j \in \Phi_1} \phi_j^{\omega_1} Y_j + \sum_{j \in \Phi_2} \phi_j^{\omega_2} Y_j))$ and each $\phi_j^{\omega_p} = \prod_{z \in \omega_p \setminus \{j\}} \frac{-z}{j-z}$ is the zero evaluation of the Lagrange coefficient of j with respect to ω_p.

For correctness, assume that $\delta = 1$ (since the case $\delta = 2$ follows similarly), and remark that:

- $\hat{e}(\sum_{j \in \Phi_1} \phi_j^{\omega_1} D_j - T_\mu, X)^{-1} = \hat{e}(\mu \sum_{j \in \Phi_1} \phi_j^{\omega_1} Q_j, rP_0^{(1)})^{-1} \cdot \hat{e}(s_0 P_0^{(2)}, rP_0^{(1)}) \cdot \hat{e}(\mu s_0 P_0^{(1)}, rP_0^{(1)})^{-1}$;
- $\hat{e}(U_\mu, \sum_{j \in \Phi_1} \phi_j^{\omega_1} Y_j) = \hat{e}(\mu P_0^{(1)}, r \sum_{j \in \Phi_1} \phi_j^{\omega_1} F(j)) \cdot \hat{e}(\mu P_0^{(1)}, r \sum_{j \in \Phi_1} \phi_j^{\omega_1} Q_j)$
  $= \hat{e}(\mu P_0^{(1)}, rs_0 P_0^{(1)}) \cdot \hat{e}(\mu \sum_{j \in \Phi_1} \phi_j^{\omega_1} Q_j, rP_0^{(1)})$;
- $\hat{e}(\sum_{j \in \Phi_1} \phi_j^{\omega_1} D_j - T_\mu, X)^{-1} \cdot \hat{e}(U_\mu, \sum_{j \in \Phi_1} \phi_j^{\omega_1} Y_j) = \hat{e}(s_0 P_0^{(2)}, rP_0^{(1)}) = g^r$.

Optimization when Threshold Value is Maximal

In some access control scenarios, users are required to have *all* the attributes of a given attribute set in order to access a given resource. For instance, some attributes of a person may appear in all her biometric readings. These attributes may therefore be grouped in a special set, and used when encrypting documents for this person. Likewise, many RBAC applications require users to have *all* the roles of a given role set in order to access (decrypt) a given resource. When thABE is used to enforce BBAC or RBAC, the aforementioned scenarios can be modelled as follows: let ω be one set of t attributes; then decryptors must have keys associated with the t elements of ω. The scheme presented below shows how to obtain a thABE scheme with short and constant-size ciphertexts, when a user must have all the attributes of a given set in order to decrypt a particular ciphertext.

1. `Instance Generator` (k). As in New-thABE.
2. `Setup` (k). As in New-thABE with $\delta = 1$ and $\mathcal{C} = \{0,1\}^{\ell} \times \mathbb{G}_1^2 \times \{0,1\}^n$.
3. `Encrypt` $(d, m, \omega, pubParams)$: Given a message $m \in \mathcal{M}$, $pubParams$, and the identity $\omega = \{i_j\}_{j=1}^{t}$ of an intended recipient, the encryptor:
   (a) picks $r \in_R \mathbb{Z}_q^*$;
   (b) computes $X = rP_0^{(1)}$ and $Y = r(P_{pub} + \sum_{j=1}^{t} Q_{i_j})$ (where $Q_{i_j} = \mathcal{H}_1(ID_{i_j})\ \forall j = 1, \cdots, t$);
   (c) computes $\sigma \in \{0,1\}^{\ell}$ such that the e^{th} bit of σ is 1 if and only if $e \in \omega$;
   (d) computes $Z = m \oplus \mathcal{H}_2(g^r)$ and outputs $c = (\sigma, X, Y, Z)$.
4. `Decrypt` $(c, D_{\omega'}, pubParams)$: Let $c = (\sigma, X, Y, Z)$ be a given ciphertext computed for the identity ω (where $|\omega| = t$). Let also $D_{\omega'} = (S_{\omega'}, T_{\mu}, U_{\mu})$ be a recipient's private key, where $S_{\omega'} = \{D_j\}_{j \in \omega'}$. Let $\Phi = \omega' \cap \omega$. Then, the recipient outputs

$$m = Z \oplus \mathcal{H}_2\left(\hat{e}(\sum_{j \in \Phi} D_j - T_{\mu}, X)^{-1} \cdot \hat{e}(U_{\mu}, Y)\right).$$

3.4 Efficiency

Table 3.1 compares the computational requirements of New-thABE with those of SW-thABE, and Table 3.2 presents corresponding timing estimates, based information obtained from [HPS02]. d denotes the (sum of all) threshold value(s), while t denotes

Schemes		Computational Requirements	Features
SW-thABE	Public Parameters	$M_{\mathbb{G}_1} + 2R_{\mathbb{G}_1} + R_{\mathbb{Z}_q^*}$	1 Attribute Set
	Key Generation	$(t^2+t)A_{\mathbb{G}_1} + t(d-1)Ex_{\mathbb{Z}_q^*} + (4t+t^2)M_{\mathbb{G}_1}$ $+ t(d-1)M_{\mathbb{Z}_q^*} + (d+t-1)R_{Zqs}$	1 Threshold Value
	Encryption	$tA_{\mathbb{G}_1} + Ex_{\mathbb{G}_2} + (2t+2)M_{\mathbb{G}_1} + \mathbf{P} + R_{\mathbb{Z}_q^*}$	
	Decryption	$d \cdot Ex_{\mathbb{G}_2} + d \cdot Inv_{\mathbb{G}_2} + (1+d)M_{\mathbb{G}_2} + 2d \cdot \mathbf{P}$	
New-thABE	Public Parameters	$M_{\mathbb{G}_1} + \mathbf{P} + 2R_{\mathbb{G}_1} + R_{\mathbb{Z}_q^*}$	Many Attribute Sets
	Key Generation	$A_{\mathbb{Z}_q^*} + (3+t) \cdot M_{\mathbb{G}_1}$	
	Encryption	$(2t)A_{\mathbb{G}_1} + Ex_{\mathbb{G}_2} + t(d-1)E_{\mathbb{Z}_q^*}$ $+ (1+2t) \cdot M_{\mathbb{G}_1} + t(d-1)M_{\mathbb{Z}_q^*} + d \cdot R_{\mathbb{Z}_q^*}$	Flexible Threshold Values
	Decryption	$(2d-1)A_{\mathbb{G}_1} + Inv_{\mathbb{G}_2}$ $+ (2d+2)M_{\mathbb{G}_1} + M_{\mathbb{G}_2} + 2\mathbf{P}$	

Table 3.1: Efficiency Comparison of New-thABE with SW-thABE.

both number of attributes used for encryption and the number of attributes which identify an arbitrary user. M_X and A_X respectively denote computational costs of scalar multiplication and addition in the Abelian group X. R_X denotes the computational cost of uniformly selecting a random element in the set X. The computational cost of exponentiation in the group X is denoted by Ex_X, and $\mathbf{P}$ denotes the computational cost of a bilinear pairing operation. Note that pairings computations are, by far, the most expensive of the operations considered above. For instance, [HPS02] indicates that, for parameters providing practical security assurance (i.e. equivalent to 1024-bit RSA [RSA78]), $\mathbf{P} \approx 4M_{\mathbb{G}_1}$, $M_{\mathbb{G}_1} \approx 100A_{\mathbb{G}_1}$, $A_{\mathbb{G}_1} \approx 14M_{\mathbb{Z}_q^*}$, $Inv_{\mathbb{Z}_q^*} \approx 16M_{\mathbb{Z}_q^*}$, and $M_{\mathbb{Z}_q^*}$ takes about 15 micro seconds in software. In this case, $Ex_{M_{\mathbb{Z}_q^*}} \approx 3.3M_{\mathbb{Z}_q^*}$ can be assumed if the exponentiation of b-long ternary representations of elements in $\mathbb{Z}_q^*$ is implemented in such a way that it requires $\frac{b}{3}$ cubing operations. Furthermore, the cost of an operation in $\mathbb{G}_2$ can be assumed to be comparable to the cost of the same operation in $\mathbb{Z}_q^*$. Note also that neither the computational cost of hash functions, nor that of additions in $\mathbb{Z}_q^*$ and exclusive OR operations are taken into account.

Table 3.1 shows that the *Setup* procedure of SW-thABE and New-thABE have similar computational requirements. In particular, both feature a short constant set of public and private parameters. However, New-thABE's *Key Generation* algorithm is significantly less computationally expensive than SW-thABE's *Key Generation* procedure. This computational difference is partially shifted to the cost requirement of

Schemes		Computational Requirements					
		t=10 d=8	t=50 d=40	t=100 d=80	t=250 d=200	t=500 d=400	t=700 d=560
SW-thABE	Public Parameters	21×10^3	21×10^3	21×10^3	21×10^3	21×10^3	21×10^3
	Key Generation	3×10^6	57×10^6	221×10^6	1350×10^6	5357×10^6	10477×10^6
	Encryption	548×10^3	2×10^6	4×10^6	11×10^6	21×10^6	30×10^6
	Decryption	1×10^6	7×10^6	13×10^6	34×10^6	67×10^6	94×10^6
New-thABE	Public Parameters	105×10^3	105×10^3	105×10^3	105×10^3	105×10^3	105×10^3
	Key Generation	273×10^3	1×10^6	2×10^6	5×10^6	11×10^6	15×10^6
	Encryption	450×10^3	2×10^6	5×10^6	14×10^6	34×10^6	55×10^6
	Decryption	549×10^3	2×10^6	3.6×10^6	9×10^6	17×10^6	24×10^6

Table 3.2: Timing Estimates of New-thABE and SW-thABE in Microseconds.

New-thABE's encryption algorithm. Indeed, SW-thABE's *Encryption* procedure requires much more computation than New-thABE's *Encryption* algorithm. Note however that the cost of New-thABE's *Encryption* algorithm is lower than SW-thABE's *Key Generation* algorithm, especially when the number of attributes used for encryption is high. The main advantage of New-thABE comes from its *Decryption* procedure which involves only 2 pairings, compared with $2d$ pairings in the case of SW-thABE's decryption algorithm. This gain is achieved by replacing pairing computations with scalar multiplications in $\mathbb{G}_1$. (The latter class of operations are significantly less expensive than the former one.) Furthermore, Table 3.1 emphasizes that New-thABE handles multiple attribute sets whose threshold values can be specified *at encryption time.*

3.5 Security

This section presents the security guarantees of our proposed thABE scheme. §3.5.1 describes our threat model and §3.5.2 presents our main security result.

3.5.1 Threat Model

To examine the security of thABE schemes, the related notions of adaptive chosen ciphertext security and chosen plaintext security are first defined. These notions are

then used to define the security of thABE schemes.

Chosen Ciphertext Security

Let Ψ be a thABE scheme. Note that Ψ's ability to encrypt for multiple sets of attributes is not intrinsically related to Ψ's security[5]. Hence, we shall assume (for simplicity) that Ψ encrypts for only one set of attributes. Thus, the following game, initiated by a challenger $\mathcal{C}h$ against an attacker $\mathcal{A}$, may be considered:

Setup: From a security parameter k, $\mathcal{C}h$ uses Ψ's *Setup* algorithm to generate the cryptosystem's public and private parameters - keeping the private parameters secret while giving the public ones to $\mathcal{A}$.

Phase 1: $\mathcal{A}$ issues to $\mathcal{C}h$ a polynomially bounded number of queries of the following types:

- *Key Extraction query*: Given an identity ω_i, $\mathcal{C}h$ must return the private key set D_{ω_i} associated with ω_i.
- *Decryption query*: Given an identity ω_i', and a ciphertext c_i encrypted for an identity ω_i such that $|\omega_i \cap \omega_i'| \geq d_i$ (where d_i is the threshold parameter associated with c_i), $\mathcal{C}h$ must return a message m_i associated with c_i.

The above queries may be issued adaptively: each request may depend on its predecessors.

Challenge: Once *Phase 1* is over, $\mathcal{A}$ issues a threshold parameter d^*, an identity ω^* of its choice, and a pair (m_0, m_1) of equal-length plaintexts, such that, in *Phase 1*, no *Key Extraction* queries were issued on an identity ω such that $|\omega \cap \omega^*| \geq d^*$. $\mathcal{C}h$ then picks a random bit $\beta \in \{0, 1\}$, computes the encryption c^* of m_β for ω^*, and sends c^* to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ issues a polynomially bounded number of *Phase 1* types of queries, under the following restrictions:

- No *Key Extraction* queries are issued on an identity ω_i such that $|\omega_i \cap \omega^*| \geq d^*$.
- No *Decryption* query is issued with c^* as an argument.

The queries may be issued adaptively: each request may depend on its predecessors.

[5]More precisely our proof of Theorem 3.5.1 could be extended to include the case in which an attacker encrypts message with multiple attribute sets. Since such an extension would add length but not clarity, we have decided to omit it.

Guess: Once *Phase 2* is over, $\mathcal{A}$ submits a guess bit β' and wins the game if $\beta' = \beta$.

Definition 3.5.1.1. *The above game is called the* IND-thABE-CCA *attack game. The quantity* $|Pr[\beta' = \beta] - \frac{1}{2}|$ *- representing the advantage of* $\mathcal{A}$ *over any challenger* $\mathcal{C}h$ *in the game - is denoted by* $Adv_{\mathcal{A},\mathcal{C}h}^{thABE-CCA}$. *A thABE scheme is said to be secure against adaptive chosen ciphertext attacks (*IND-thABE-CCA *secure, in short) if no polynomially bounded attacker can be found that has non-negligible advantage in the above* IND-thABE-CCA *game.*

Chosen Plaintext Security

The notion of chosen plaintext security is similar to (and weaker than) the notion of *chosen ciphertext security.* To define semantic security for thABE schemes, one may consider a game (called the *IND-thABE-CPA* game), which is identical to the *IND-thABE-CCA* game, except that the attacker $\mathcal{A}$ is unable to issue *Decryption* queries.

Definition 3.5.1.2. *The value* $|Pr[\beta' = \beta] - \frac{1}{2}|$ *- representing the advantage of* $\mathcal{A}$ *over any challenger* $\mathcal{C}h$ *in the* IND-thABE-CPA *game - is denoted by* $Adv_{\mathcal{A},\mathcal{C}h}^{thABE-CPA}$. *A thABE scheme is said to be secure against adaptive chosen plaintext attacks (*IND-thABE-CPA *secure, in short) if no polynomially bounded attacker can be found that has non-negligible advantage in a* IND-thABE-CPA *game.*

3.5.2 Security Theorem

Theorem 3.5.1. *Let* k *be a security parameter. Assume that the hash functions of New-thABE are random oracles. Suppose also that there exists an attacker* $\mathcal{A}$ *which has non-negligible advantage* $\varepsilon(k)$, *in time* τ, *against any challenger* $\mathcal{C}h$, *in the* IND-thABE-CPA *game. Then, there exists an algorithm* $\mathcal{B}$ *which solves the BDH problem, in time* $O(\tau)$, *with non-negligible advantage at least* $\frac{\varepsilon(k)}{q_{\mathcal{H}_2}}$, *where* $q_{\mathcal{H}_2}$ *is the number of* $\mathcal{H}_2$ *queries issued by* $\mathcal{A}$ *in the attack game.*

Practical Meaning of Theorem 3.5.1: In practice, this theorem indicates that, when a message is encrypted for a user who has at least d of a number of specified attributes, the corresponding ciphertext can only be decrypted by those who meet this requirement.

Proof: In this proof, we show how to construct $\mathcal{B}$ using $\mathcal{A}$. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two Abelian groups of prime order q, where $\mathbb{G}_1$ is additive and $\mathbb{G}_2$ is multiplicative. Let $P_0^{(1)} \in \mathbb{G}_1^*$ be a generator of $\mathbb{G}_1$, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be a bilinear pairing on $\mathbb{G}_1$, such that the BDH is assumed to be hard with respect to $\hat{e}$, and let

$(\mathbb{G}_1, q, P_0^{(1)}, aP_0^{(1)}, bP_0^{(1)}, cP_0^{(1)})$ be a tuple for which a, b, c are unknown to $\mathcal{B}$. $\mathcal{B}$'s goal is to solve the BDH Problem by computing $\hat{e}(P_0^{(1)}, P_0^{(1)})^{abc}$ in polynomial time. To achieve this goal, $\mathcal{B}$ initiates an *IND-thABE-CPA* game with $\mathcal{A}$, using an arbitrary security parameter k. For simplicity, we shall assume that New-thABE is only used to encrypt for one set of attributes at a time (instead of two). The proof we construct could be easily be adapted to deal with the case $\delta = 2$.

$\mathcal{B}$ first sets $n = poly_1(k)$ and $\ell = poly_2(k)$, using polynomials $poly_1$ and $poly_2$. Then $\mathcal{B}$ picks $s_0 \in_R \mathbb{Z}_q^*$, $\eta \in_R \mathbb{Z}_q^*$, and sets $P_0^{(3)} = \eta P_0^{(1)}$, $P_{pub} = aP_0^{(1)}$, $X^* = bP_0^{(1)}$, and $P_0^{(2)} = cP_0^{(1)}$. $\mathcal{B}$ also sets $g = \hat{e}(P_{pub}, P_0^{(2)})$ and defines two hash functions $\mathcal{H}_1^{sim} : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $\mathcal{H}_2^{sim} : \mathbb{G}_2 \rightarrow \{0,1\}^n$ over which it has full control. Then, $\mathcal{B}$ sets $\mathcal{M} = \{0,1\}^n$, $\mathcal{C} = \mathbb{N}_\ell \times \{0,1\}^\ell \times \mathbb{G}_1^{\ell+1} \times \{0,1\}^n$, $pubParams = (q, n, \hat{e}, P_0^{(1)}, g, \mathcal{H}_1, \mathcal{H}_2)$, $params = (pubParams, a, P_0^{(2)}, P_0^{(3)})$, where $s_0 = a$ is not known by $\mathcal{B}$. $\mathcal{B}$ also picks $\mu \in_R \mathbb{Z}_q^*$. Throughout the attack game, μ will be used to "personalize" the keys sent to $\mathcal{A}$. (Since $\mathcal{A}$ is the only users requesting keys, there is no need to personalize the keys for many users.)

The rest of the proof consists of three sections. The *Hash Simulation* section shows how $\mathcal{B}$ simulates $\mathcal{H}_1^{sim}$ and $\mathcal{H}_2^{sim}$. The *Attack Game* section explains how $\mathcal{B}$ handles the queries of the attack game. Finally, the *Complexity and Probability* section derives the complexity and probability results of the theorem.

Hash Simulation:

For $\mathcal{H}_1^{sim}$-queries, $\mathcal{B}$ maintains a list $L_{\mathcal{H}_1^{sim}}$ whose entries have the form $(\alpha, h_{1,\alpha})$, where $\alpha \in \{0,1\}^*$ and $h_{1,\alpha} \in \mathbb{G}_1$ is the simulated value of $\mathcal{H}_1(\alpha)$. Thus, on input $\alpha \in \{0,1\}^*$, $\mathcal{B}$ proceeds as follows:

- If $L_{\mathcal{H}_1^{sim}}$ already has an entry $(\alpha, h_{1,\alpha})$, then $\mathcal{B}$ returns $h_{1,\alpha}$ as an answer to the $\mathcal{H}_1^{sim}$-query;
- Otherwise, $\mathcal{B}$ picks $h_{1,\alpha}$ uniformly at random in $\mathbb{G}_1$, adds $(\alpha, h_{1,\alpha})$ to $L_{\mathcal{H}_1^{sim}}$, and returns $h_{1,\alpha}$ as an answer to the $\mathcal{H}_1^{sim}$ query.

$\mathcal{B}$ also maintains a list $L_{\mathcal{H}_2^{sim}}$ for $\mathcal{H}_2^{sim}$-queries, where the entries of $L_{\mathcal{H}_2^{sim}}$ have the form $(\gamma, h_{2,\gamma})$, where $\gamma \in \mathbb{G}_2$ and $h_{2,\gamma} \in \{0,1\}^n$ is the simulated value of $\mathcal{H}_2(\gamma)$. Thus, on input $\gamma \in \mathbb{G}_2$, $\mathcal{B}$ proceeds as follows:

- If $L_{\mathcal{H}_1^{sim}}$ already has an entry $(\gamma, h_{1,\gamma})$, then $\mathcal{B}$ returns $h_{1,\gamma}$ as an answer to the $\mathcal{H}_1^{sim}$-query;
- Otherwise, two cases are distinguished:

1. If, for some existing entry $(\gamma', h_{2,\gamma'})$ of $L_{\mathcal{H}_2^{sim}}$, the discrete logarithm of γ' with respect to $\hat{e}(P_0^{(1)}, P_0^{(2)})$ is equal to the discrete logarithm of γ with respect to $\hat{e}(P_0^{(1)}, P_0^{(3)})$, then $\mathcal{B}$ returns $h_{2,\gamma'}$. (Note that the equality of discrete logarithm can be tested using standard techniques [ACJT00]. Moreover, note that the search of a potential entry with equal discrete logarithm takes at most $q_{\mathcal{H}_2}$ trials.)
2. Otherwise, $\mathcal{B}$ picks $h_{1,\alpha}$ uniformly at random in $\{0,1\}^n$, adds $(\alpha, h_{1,\alpha})$ to $L_{\mathcal{H}_1^{sim}}$, and returns $h_{1,\alpha}$ as an answer to the $\mathcal{H}_1^{sim}$ query.

Attack Game:

$\mathcal{B}$ handles $\mathcal{A}$'s queries as follows:

- *Phase 1*: Given an identity $\omega_i = \{i_j\}_{j=1}^{t_i}$, $\mathcal{B}$ computes $D_{\omega_i} = (S_{\omega_i}, T_\mu, U_\mu)$, where $U_\mu = \mu P_0^{(1)}$, $T_\mu = \eta P_{pub} - \mu P_{pub} = s_0 P_0^{(3)} - \mu P_{pub}$, and $S_{\omega_i} = \{D_{i_j}\}_{j=1}^{t_i}$ with $D_{i_j} = \mu \cdot Q_{i_j}$ and $Q_{i_j} = \mathcal{H}_1(ID_{i_j})$ for $1 \leq j \leq t_i$. $\mathcal{B}$ then sends D_{ω_i} to $\mathcal{A}$.
- *Challenge Phase*: $\mathcal{A}$ issues a threshold parameter d^*, an identity $\omega^* = \{i_j^*\}_{j=1}^{t^*}$ of its choice, and a pair (m_0, m_1) of equal-length plaintexts, such that, in *Phase 1*, no *Key Extraction* queries were issued on an identity ω such that $|\omega \cap \omega^*| \geq d$. $\mathcal{B}$ then picks a random bit $\beta \in \{0,1\}$, $Z^* \in_R \{0,1\}^n$, and defines a polynomial $F(x) = P_{pub} + (\sum_{z=1}^{d^*-1} r_z x^z) P_0^{(1)}$, where $r_z \in_R \mathbb{Z}_q^*$ for $z = 1, \cdots, d^*$. $\mathcal{B}$ also computes $Y_{i_j^*} = F(i_j^*) + h_{1,ID_{i_j^*}} X^*$ for $j = 1, \cdots, t^*$, where $(ID_{i_j^*}, h_{1,ID_{i_j^*}})$ is the entry of $L_{\mathcal{H}_1^{sim}}$ used to simulate the output of $\mathcal{H}_1(ID_{i_j^*})$. Then $\mathcal{B}$ computes σ^* according to the method described in New-thABE's *Encryption* procedure, and returns $c^* = (\sigma^*, X^*, (Y_{i_j^*})_{j=1}^{t^*}, Z^*)$ to $\mathcal{A}$.
- *Phase 2*: $\mathcal{B}$ proceeds as in *Phase 1*.
- *Guess*: $\mathcal{A}$ sends $\mathcal{B}$ his guess β' and wins if $\beta' = \beta$. Regardless of $\mathcal{A}$'s success, $\mathcal{B}$ picks (uniformly at random) an entry $(\gamma, h_{2,\gamma})$ of $L_{\mathcal{H}_2^{sim}}$, and submits $\gamma \cdot \hat{e}(\mu P_{pub}, X^*) \cdot \hat{e}(\mu P_0^{(1)}, P_{pub})^{-1}$ as a solution of the BDH problem.

Complexity and Probability:

Let $c = (d, \sigma, X, (Y_{u_v})_{v=1}^{t}, Z)$ be a valid ciphertext computed for an identity ω. Then $L_{\mathcal{H}_2^{sim}}$ contains an entry of the form $(\gamma, h_{2,\gamma})$, where $\gamma = g^r = \hat{e}(P_{pub}, P_0^{(2)})^r = \hat{e}(P_0^{(1)}, P_0^{(2)})^{s_0 r}$ for some $r \in \mathbb{Z}_q^*$ such that $X = rP_0^{(1)}$. Suppose that $\omega_i = \{i_j\}_{j=1}^{t_i}$ is some identity for which $\mathcal{A}$ receives $D_{\omega_i} = (S_{\omega_i}, T_\mu, U_\mu)$ from $\mathcal{A}$. Then $U_\mu = \mu P_0^{(1)}$, $T_\mu = \alpha P_{pub} - \mu P_{pub} = s_0 P_0^{(3)} - \mu P_{pub}$, and $S_{\omega_i} = \{D_{i_j}\}_{j=1}^{t_i}$ with $D_{i_j} = \mu \cdot Q_{i_j}$ and

$Q_{i_j} = \mathcal{H}_1(ID_{i_j})$ for $1 \leq j \leq t_i$. Suppose that Φ is a d-element subset of $\omega \cap \omega_i$. Then $\lambda = \hat{e}(\sum_{j\in\Phi} \phi_j^{\Phi} D_j - T_\mu, X)^{-1} \cdot \hat{e}(U_\mu, \sum_{j\in\Phi} \phi_j^{\Phi} Y_j) = \hat{e}(s_0 P_0^{(3)} - \mu P_{pub}, r P_0^{(1)}) \cdot \hat{e}(U_\mu, r P_{pub}) = \hat{e}(P_0^{(3)}, P_0^{(1)})^{s_0 r}$. So the discrete logarithm (DL) of λ with respect to $\hat{e}(P_0^{(3)}, P_0^{(1)})$ is equal to the DL of g^r with respect to $\hat{e}(P_0^{(1)}, P_0^{(2)})$. Thus, if $\mathcal{A}$ encrypts a message m, and decrypts the corresponding ciphertext with the keys it receives from $\mathcal{B}$, then $\mathcal{A}$ recovers m.

Moreover, remark that the only way for $\mathcal{A}$ to win the attack game with non-negligible probability is to, somehow, obtain attribute keys associated with a d-element subset Φ^* of ω^*, and to use c^* to decrypt Z^*. Note that the attribute keys provided by $\mathcal{B}$ do not enable $\mathcal{A}$ to perform such a decryption. Thus, $\mathcal{A}$ must somehow obtain a d-element set of *valid* attribute keys. Moreover, note that the decryption of Z^* involves the computation of $\mathcal{H}_2(\zeta)$, where $\zeta = \hat{e}(\sum_{j\in\Phi^*} \phi_j^{\Phi^*} D_j - T_\mu, X^*)^{-1} \cdot \hat{e}(U_\mu, \sum_{j\in\Phi^*} \phi_j^{\Phi^*} Y_j^*) = \hat{e}(s_0 P_0^{(2)} - \mu P_{pub}, b P_0^{(1)}) \cdot \hat{e}(U_\mu, P_{pub}) = g^b \cdot \hat{e}(\mu P_{pub}, r P_0^{(1)})^{-1} \cdot \hat{e}(U_\mu, P_{pub})$. Since $g^b = \hat{e}(P_0^{(1)}, P_0^{(1)})^{abc}$, the solution of the BDH can be obtained by computing $\zeta \cdot \hat{e}(\mu P_{pub}, X^*) \cdot \hat{e}(\mu P_0^{(1)}, P_{pub})^{-1}$. Thus, if $\mathcal{B}$ picks a random entry $(\gamma, h_{2,\gamma})$ of $L_{\mathcal{H}_2^{sim}}$ and if $\gamma = \zeta$, then $\mathcal{B}$ solves the BDH problem.

Let $\varepsilon(k)$ be $\mathcal{A}$'s advantage in the *IND-thABE-CPA* game. Then, $\mathcal{B}$ solves the BDH problem with probability at least $\frac{\varepsilon(k)}{q_{\mathcal{H}_2}}$, where $q_{\mathcal{H}_2}$ is the number of $\mathcal{H}_2$ queries issued by $\mathcal{A}$ in the attack game. Moreover, since each query of the *IND-thABE-CPA* game requires $\mathcal{B}$ to make a polynomial number of search operations and a polynomial number of operations in $\mathbb{Z}_q^*$, $\mathbb{G}_2$ and $\mathbb{G}_1$, it follows that $\mathcal{B}$ solves the BDH problem in time $O(\tau)$ where τ is the running time of $\mathcal{A}$ in the *IND-thABE-CPA* attack game.
QED.

A Note on Chosen Ciphertext Security

In [BF01], it is shown how to prove that the FullIndent scheme is IND-ID-CCA secure using the fact that the BasicIdent scheme is IND-ID-CPA secure, where FullIdent is obtained from BasicIdent by applying the so-called Fujisaki-Okamoto padding [FO99]. In the same way, New-thABE can be transformed into a scheme which is IND-thABE-CCA secure. This can be done as follows: (1) transform New-thABE into a scheme Basic-New-thABE-Pub (using the method described in [BF01] to transform BasicIdent into BasicPub); (2) show that Basic-New-thABE-Pub is IND-CPA if the BDH is intractable (using the method described in the proof of Theorem 3.5.1); (3) show that New-thABE is IND-ID-CPA if Basic-New-thABE-Pub is IND-CPA (using the method described in the proof of Lemma 4.2 of [BF01]); (4) transform Basic-New-thABE-Pub into a scheme Full-New-thABE-Pub (by applying Fujisaki-Okamoto padding [FO99]); transform Full-New-thABE-Pub into a scheme Full-New-thABE (as FullIdentPub is

transformed into FullIdent in [BF01]); show that Full-New-thABE is IND-ID-CCA if Full-New-thABE-Pub is IND-CCA (using the method described in the proof of Lemma 4.6 of [BF01]); apply Lemma 4.5 of [BF01] to prove that Full-New-thABE-Pub is IND-CCA if Basic-New-thABE-Pub is IND-CPA; conclude that Full-New-thABE is IND-ID-CCA secure if the BDH problem is intractable. (Note that the above line of reasoning has been considered standard in the literature [GS02, SW05].)

3.6 Conclusion

The aim of this chapter was to explain how to construct efficiently-usable decryption keys from non-repeatable information. To do so, we first assumed that a set of attributes were extracted from from non-repeatable information (e.g. readings of biometric information). Then, we assumed that these attributes are mapped to unique values (using a one-way hash function). Next, we described a provably-secure efficient collusion-resistant threshold attribute-based encryption (thABE) scheme which handles multiple attribute sets with dynamically-specifiable threshold values. This thABE scheme includes a key generation algorithm which constructs keys from the aforementioned (biometric) attributes.

Applications of our thABE scheme include *biometric-based* and *role-based* cryptographic access control. To the best of our knowledge, the proposed scheme is the most efficient one of its class. In particular, the new scheme is significantly more efficient and flexible than Sahai and Waters' scheme. Moreover, the proposed scheme was proven secure in the random oracle model, under a standard number theoretic assumption. An optimization of our scheme allows to produce constant-size ciphertexts, when intended decryptors must have all the attributes of given attribute sets. However, it remains an open question whether it is possible to design a thABE scheme featuring constant-size ciphertexts regardless of both the number of target attributes and the specified threshold values. Moreover, the topic of threshold attribute-based signature could be investigated.

Chapter 4

Efficient Hierarchical Identity-Based Cryptography

The second main contribution of this dissertation is the design of a provably secure hierarchical ID-based signcryption scheme with public ciphertext authenticity and forward security. Each signcryption scheme relies on an encryption scheme and a related signature scheme. The aim of this chapter is to present *the first part* of the aforementioned contribution by describing novel and efficient hierarchical ID-based encryption (HIBE) and signature (HIBS) schemes.

HIBE and HIBS enable convenient encrypted and authenticated transmissions in large and distributed user communities. In very recent independent work, Boneh et al. [BBG05] described a HIBE scheme with constant-size ciphertexts and constant-time decryption. In that scheme, the length and the generation time of decryption keys are both linear with respect to the depth of users in the underlying hierarchy. This chapter introduces a novel hierarchical ID-based key generation (HIBKG) algorithm running in constant time and producing constant-size keys. The proposed HIBKG algorithm is first used to describe a HIBE scheme with constant-time decryption and constant-size ciphertexts. Then it is used to describe a HIBS scheme with constant-time signature and constant-size signatures. Other applications of the proposed HIBKG algorithm include an efficient ID-based forward-secure HIBE scheme and an efficient append-only signature scheme.

§4.1 reviews our motivation for the design of HIBE and HIBS schemes. §4.2 presents standard terminology related to HIBE and HIBS. §4.3 describes our proposed HIBE scheme, and §4.4 presents the proposed HIBS scheme. §4.5 discusses the computational and space requirements of our HIBE and HIBS schemes, in comparison with related schemes. §4.6 summarizes the security guarantees of the novel HIBE scheme, and §4.7 concludes the chapter.

4.1 Introduction

Two primary motivations for the design of *hierarchical* ID-based cryptographic schemes are *scalability* and *damage control in case of PKG compromise.*

Scalability

In non-hierarchical ID-based cryptographic schemes, all users of a domain deal with only one private key generator (PKG). This trusted entity must authenticate each user of the domain, generate a private key for each user, and securely give this key to the associated user. Consequently, when the number of users in the domain increases, the computational and communicational load placed on this PKG for the generation and distribution of user keys becomes very heavy.

One way to address this issue is to use a mechanism whereby a root PKG can delegate to lower PKGs the task of identifying users, generating user private keys, and distributing these keys in a secure fashion. This delegation could consist of generating a sub-domain private key, and of secretly giving this key to the appropriate lower PKG. The mechanism should be recursive, so that PKGs placed directly under the root PKG can also delegate (part of) their responsibilities to lower PKGs. In this paradigm, PKGs form a hierarchy, and the generation of private keys is hierarchical. Hence, ID-based schemes providing hierarchical key generation are named *hierarchical* ID-based (HIB) schemes.

An important feature of HIB schemes is the fact that there is only one set of public parameters for all the PKGs forming a hierarchy. This means that, if a user $\mathcal{A}$ belongs to a hierarchy H, then encryptors only need to obtain one set of public parameters associated with H in order to cipher documents for $\mathcal{A}$. Likewise, the verification of HIB signatures issued by $\mathcal{A}$ only requires the obtention of H's public key parameters.

Consequently, HIB schemes provide scalability to large and distributed user communities.

Damage Control in Case of PKG Compromise

In addition to scalability, HIB schemes provide tighter control on the effects of PKG compromise.

Suppose, for instance, that the private key of a PKG is compromised. Then, only the PKGs and the users falling under the compromised PKG are affected. Assuming that high-level PKGs (i.e. the root PKG and all PKGs of levels 1 through a small integer) are more difficult to compromise than lower-level PKGs, it follows that HIB schemes provide a mechanism to limit the effect of PKG compromise.

Other Motivating Factors

HIB schemes appear to be the only known mechanisms which can be used to build both forward secure public-key encryption (fsPKE) schemes and forward secure public-key signature (fsPKS) schemes. fsPKE and fsPKS schemes assume that the life time of a system is divided into periods. Moreover, each user is given a private time-bound key when she joins the system. Each user must also evolve her time-bound key at the beginning of each time period. Each ciphertext c is associated with a time period, and a user must possess a decryption key associated with this or a preceding time period in order to decrypt c. Thus, the compromise of a time-bound key at a given time period only compromises the confidentiality of ciphertexts associated with this or subsequent time periods. Likewise, signatures are associated with a time period, and the compromise of a private signing key at a given time period only allows to issue signatures associated with this or subsequent time periods. fsPKE and fsPKS schemes are therefore useful to mitigate the effect of private key compromise. Consequently, the design of HIB schemes is also motivated by the fact that one can thereby build cryptographic schemes which reduce the impact of private key compromise.

Another motivation for the design of HIB schemes is the fact that HIBS schemes can be used to derive append-only signature (AOS) schemes. Given an AOS signature on a message m_1, a user is only able to issue an AOS signature on the concatenation of m_1 with any message m_2 (i.e. "m_2 appended to m_1"). This is useful for both privilege delegation in access control applications, and path authentication in network routing. In most access control applications, users are given privileges, and these privileges can sometimes be delegated to other users. If this is the case, a user can only delegate privileges which have been given to her. AOS signatures can be used to support this process in the following way: (1) each user (say $ID_{\mathcal{A}}$) receives, from a trusted entity, an AOS signature on a set of privileges; (for instance, assume that m_1 is a privilege, and that $ID_{\mathcal{A}}$ is given an AOS signature on $ID_{\mathcal{A}}||m_1$ from a trusted entity;) (2) in order to delegate m_1 to another user (say $ID_{\mathcal{B}}$), $ID_{\mathcal{A}}$ gives $ID_{\mathcal{B}}$ an OAS signature on $ID_{\mathcal{A}}||m_1||ID_{\mathcal{B}}$; (3) suppose now that $\mathcal{B}$ needs m_1 to access restricted information; then $\mathcal{B}$ shows the OAS signature on $ID_{\mathcal{A}}||m_1||ID_{\mathcal{B}}$, and obtains the restricted information if the AOS signature is valid. Another application of AOS schemes can be found in network routing. In this context, network nodes receive network paths, and are only able to add their network information to received paths. Hence AOS signatures can be used to support this requirement. Further details on the use of AOS signatures for privilege delegation and network routing are provided in Chapter 5. However, our goal here is to note that the design of HIB schemes is also motivated by the possibility to derive AOS schemes from HIBS schemes. §4.2 reviews important terminology related to HIBE and HIBS.

4.2 Preliminaries

In this section, we recall fundamental definitions related to HIBE and HIBS. Readers familiar with [GS02] may go to §4.3.

ID-tuples

For the description of HIBE schemes, PKGs are assumed to be hierarchically organized. We only consider *tree*-shaped hierarchies, and denote by *user* any PKG except the *root PKG*. This root PKG is the root of the tree-shaped hierarchy, and is denoted by $rPKG$. Every user is identified with a tuple $\overline{ID}_t = (ID_1, ID_2, \cdots, ID_t)$, which is called an *ID-tuple*. $\overline{ID}_t$'s ancestors consists of $rPKG$ and the $\overline{ID}_i$'s such that $1 \leq i < t$.

Hierarchical ID-based Encryption Scheme

Each HIBE scheme is composed of four algorithms, each of whose function is described below:

1. `Root Setup`: Given a security parameter k, this algorithm is used by the root PKG to return a tuple *params* of system parameters. These parameters include a description of the message space $\mathcal{M}$ and the ciphertext space $\mathcal{C}$, along with a secret piece of data s_0 called the *root secret key*. Other parameters are allowed, *provided they be not unique to any user*. Ideally, these other parameters should not be unique to a strict subset of users (*including users who all belong the same hierarchical level.*). However, this latter optimization is not mandatory. Some parameters may be public (including those describing $\mathcal{M}$ and $\mathcal{C}$), while others remain secret (including the root secret key).

2. `Extract`: Given the scheme's public parameters, an arbitrary identifier ID_{t+1}, and the private key $d_{\overline{ID}_t}$ of a PKG identified by $\overline{ID}_t$, this algorithm returns the private key $d_{\overline{ID}_{t+1}}$ of the *child* $\overline{ID}_{t+1}$ of $\overline{ID}_t$.

3. `Encrypt`: Given a message $m \in \mathcal{M}$, the ID-tuple $\overline{ID}_t$ of an intended recipient, and given the scheme's public parameters, this algorithm returns a ciphertext $c \in \mathcal{C}$ corresponding to m.

4. `Decrypt`: Given a ciphertext $c \in \mathcal{C}$, the ID-tuple $\overline{ID}_t$ of the intended recipient, and given the scheme's public parameters along with the private key $d_{\overline{ID}_t}$ associated with $\overline{ID}_t$, this algorithm returns the message $m \in \mathcal{M}$ associated with c.

Encryption and decryption must satisfy the following consistency constraint:

$\forall\ m \in \mathcal{M}$ Decrypt(c, $\overline{ID}_t$, *params*, $d_{\overline{ID}_t}$) = m, where c = Encrypt(m, $\overline{ID}_t$, *params*).

Hierarchical ID-based Signature Scheme

Each HIBS scheme is composed of four algorithms, whose functions are described below:

1. `Root Setup`: Given a security parameter k, this algorithm is used by the root PKG to return a tuple *params* of system parameters. These parameters include a description of the message space $\mathcal{M}$ and the signature space $\mathcal{S}$, along with a secret piece of data s_0 called the *root secret key*. Other parameters are allowed, *provided they be not unique to any user*. Ideally, these other parameters should not be unique to a strict subset of users (*including users who all belong the same hierarchical level.*). However, this latter optimization is not mandatory. Some parameters may be public (including those describing $\mathcal{M}$ and $\mathcal{C}$), while others remain secret (including the root secret key).

2. `Key Extraction`: Given the scheme's public parameters, an arbitrary identifier ID_{t+1}, and the private key $d_{\overline{ID}_t}$ of a PKG identified by $\overline{ID}_t$, this algorithm returns the private key $d_{\overline{ID}_{t+1}}$ of the *child* $\overline{ID}_{t+1}$ of $\overline{ID}_t$.

3. `Signing`: Given a message $m \in \mathcal{M}$, the ID-tuple $\overline{ID}_{t_A}$ of a sender, the private key $d_{\overline{ID}_{t_A}}$ of $\overline{ID}_{t_A}$, and the scheme's public parameters, this algorithm returns a signature $\sigma \in \mathcal{S}$ corresponding to m.

4. `Verification`: Given a message $m \in \mathcal{M}$, a signature $\sigma \in \mathcal{S}$, the ID-tuple $\overline{ID}_{t_A}$ of a sender, and the scheme's public parameters, this algorithm returns either "Invalid" or "Valid", to indicate whether σ is a valid signature of m.

Signing and *Verification* must satisfy the following consistency constraint:

$$\forall\ m \in \mathcal{M}\ \text{Verification}(\sigma, \overline{ID}_{t_A}, \textit{params}) = \text{“valid”},$$
$$\text{if } \sigma = \text{Signing}(m, \overline{ID}_{t_A}, d_{\overline{ID}_{t_A}}, \textit{params}).$$

4.3 Proposed HIBE Scheme

We describe our scheme in two steps (NewHIBE-v1 is first presented, followed by NewHIBE-v2). This is done because of the following two reasons: first, via the argument presented in §3.5.2 (concerning the chosen ciphertext security of New-thABE),

NewHIBE-v2 can be shown to be IND-HIB-CCA secure if NewHIBE-v1 is IND-HIB-CPA secure; second, IND-HIB-CPA security is easier to prove than IND-HIB-CCA security. (See §4.6 for definitions of IND-HIB-CPA and IND-HIB-CCA, and see §3.5.2 for the use of a similar argument.)

NewHIBE-v1

- `Instance Generator` (k). This procedure, denoted by IG, is a randomized algorithm which takes a security parameter $k > 0$, runs in $O(k)$, and outputs $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are two Abelian groups of prime order $q \geq 2^k$, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is an admissible pairing with respect to which $\mathbb{G}_1$ and $\mathbb{G}_2$ are Gap-Diffie-Hellman groups.

- `Root Setup` (k). Given a security parameter $k > 0$, the root *PKG*:

  1. runs IG with input k and obtains $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$.
  2. picks, randomly and uniformly[1], $P_0^{(1)}, P_0^{(2)}, P_0^{(3)}, P_0^{(4)}, P_0^{(5)}, P_0^{(6)} \in \mathbb{G}_1$;
  3. picks $s_0, I_0 \in_R \mathbb{Z}_q^*$;
  4. picks $\ell = \log(k)$ and computes $n = poly_1(k)$, where $poly_1$ is any polynomial over the positive integers (n is the message length and ℓ is the maximal depth of the user hierarchy);
  5. chooses cryptographic hash functions:
     $\mathcal{H}_1 : \{0,1\}^* \rightarrow (\mathbb{Z}_q^*)^{3\ell}$, $\mathcal{H}_2 : \mathbb{G}_2 \rightarrow \{0,1\}^n$, $\mathcal{H}_5 : \mathbb{Z}_q^* \rightarrow \mathbb{Z}_q^*$.
     where the image through $\mathcal{H}_1$ of a (t-long) ID-tuple $\overline{ID}_t = (ID_1, \cdots, ID_t)$ is $\mathcal{H}_1(\overline{ID}_t) = (J_1, \cdots, J_t, J_0, \cdots, J_0) \in (\mathbb{Z}_q^*)^{3\ell}$, where: $J_0 = (I_0, I_0, I_0)$; $J_i = (I_i, p_{(i,1)}, p_{(i,3)})$ for $1 \leq i \leq t$; $I_i = \mathcal{H}_6(ID_i), p_{(i,1)} = \mathcal{H}_7(I_i), p_{(i,3)} = \mathcal{H}_7(p_{(i,1)})$ for $1 \leq i \leq t$; $\mathcal{H}_6$ is any cryptographic hash function from $\{0,1\}^*$ to $\mathbb{Z}_q^*$, and $\mathcal{H}_7$ is any cryptographic hash function from $\mathbb{Z}_q^*$ to $\mathbb{Z}_q^*$.
  6. computes, $(s_i = \mathcal{H}_5(s_{i-1}))_{i=1}^{\ell-1}$, $L_1^{(j)} = s_0 P_0^{(j)}$ for $j = 3, 4$, and $\left(L_i^{(j)} = s_{i-1} L_{i-1}^{(j)}\right)_{i=2}^{\ell}$ for $j = 3, 4$.

  The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = \mathbb{G}_1^2 \times \{0,1\}^n$. The system's public parameters (which must be certified) are $pubParams = (q, n, \hat{e}, I_0, P_0^{(1)}, P_0^{(2)}, P_0^{(3)}, P_0^{(4)}, P_0^{(5)}, P_0^{(6)}, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_5, ((L_i^{(j)})_{i=1}^{\ell})_{j=3}^{4})$, and the *root PKG* keeps s_0 secret, so that $params = (pubParams, s_0)$.

[1] In the sequel, we shall use the notation $x \in_R X$ to indicate that the element x is chosen uniformly at random from the set X.

- `Extract`:
  - *Root-level Extract*: For each first-level user $\overline{ID}_1 = (ID_1)$, the root PKG picks $\alpha_{\overline{ID}_1}, \widetilde{\alpha}_{\overline{ID}_1} \in_R \mathbb{Z}_q^*$, and computes $J_1 = (I_1, p_{(1,1)}, p_{(1,3)})$ (with ID_1 and $\mathcal{H}_1$), $p_{(1,2)} = \mathcal{H}_7(p_{(1,3)})$, $p_{(1,4)} = \frac{p_{(1,2)}p_{(1,3)}}{p_{(1,1)}}$, $I_1' = -p_{(1,2)}$, $I_1'' = -\frac{p_{(1,1)}}{p_{(1,2)}} I_1'$, $\widetilde{I}_1 = I_1 + p_{(1,1)}$, $\widetilde{I}_1' = I_1' + p_{(1,2)}$, $\widetilde{I}_1'' = I_1'' + p_{(1,4)}$, $R_{\overline{ID}_1} = p_{(1,1)}\alpha_{\overline{ID}_1} + p_{(1,2)}\widetilde{\alpha}_{\overline{ID}_1}$, $\widetilde{R}_{\overline{ID}_1} = p_{(1,3)}\alpha_{\overline{ID}_1} + p_{(1,4)}\widetilde{\alpha}_{\overline{ID}_1}$, $s_1 = \mathcal{H}_5(s_0)$, and the following:
    $S_{\overline{ID}_1} = s_0(I_1 P_0^{(1)} + I_1' P_0^{(2)} + \alpha_{\overline{ID}_1} P_0^{(3)})$, $E_{\overline{ID}_1} = s_0(\frac{I_1 I_1''}{I_1'} P_0^{(1)} + I_1'' P_0^{(2)} + \widetilde{\alpha}_{\overline{ID}_1} P_0^{(3)})$,
    $\widetilde{S}_{\overline{ID}_1} = s_0(\alpha_{\overline{ID}_1} P_0^{(4)} + \widetilde{I}_1 P_0^{(5)} + \widetilde{I}_1' P_0^{(6)})$, $\widetilde{E}_{\overline{ID}_1} = s_0(\widetilde{\alpha}_{\overline{ID}_1} P_0^{(4)} + \frac{\widetilde{I}_1 \widetilde{I}_1''}{\widetilde{I}_1'} P_0^{(5)} + \widetilde{I}_1'' P_0^{(6)})$;

    Then, the root PKG gives $d_{\overline{ID}_1} = (S_{\overline{ID}_1}, E_{\overline{ID}_1}, \widetilde{S}_{\overline{ID}_1}, \widetilde{E}_{\overline{ID}_1}, R_{\overline{ID}_1}, \widetilde{R}_{\overline{ID}_1}, p_{(1,2)}, s_1)$ to $\overline{ID}_1$.
  - *Lower-level Extract*: For each child $\overline{ID}_{t+1} = (ID_1, \cdots, ID_{t+1})$ of a user $\overline{ID}_t$ $(t \geq 1)$, $\overline{ID}_t$ computes: $J_t = (I_t, p_{(t,1)}, p_{(t,3)})$ and $J_{t+1} = (I_{t+1}, p_{(t+1,1)}, p_{(t+1,3)})$ (with ID_t, ID_{t+1} and $\mathcal{H}_1$); $I_{t+1}' = -p_{(t,2)}$, $I_{t+1}'' = -\frac{p_{(t,2)}p_{(t,3)}}{p_{(t,1)}}$, $\widetilde{I}_{t+1} = I_{t+1} + p_{(t,1)}$, $\widetilde{I}_{t+1}' = I_{t+1}' + p_{(t,2)}$, $\widetilde{I}_{t+1}'' = I_{t+1}'' + p_{(t,4)}$; $p_{(t+1,2)} = -\frac{p_{(t,1)}}{p_{(t,3)}} p_{(t+1,1)}$, $p_{(t+1,4)} = \frac{p_{(t+1,2)}p_{(t+1,3)}}{p_{(t+1,1)}}$, $\alpha_{\overline{ID}_{t+1}}, \widetilde{\alpha}_{\overline{ID}_{t+1}} \in_R \mathbb{Z}_q^*$, $R_{\overline{ID}_{t+1}} = p_{(t+1,1)}\alpha_{\overline{ID}_{t+1}} + p_{(t+1,2)}\widetilde{\alpha}_{\overline{ID}_{t+1}}$, $\widetilde{R}_{\overline{ID}_{t+1}} = p_{(t+1,3)}\alpha_{\overline{ID}_{t+1}} + p_{(t+1,4)}\widetilde{\alpha}_{\overline{ID}_{t+1}}$; $s_{t+1} = \mathcal{H}_5(s_t)$, and the following:

    $$
    \begin{aligned}
    S_{\overline{ID}_{t+1}} &= s_t(I_{t+1} S_{\overline{ID}_t} + I_{t+1}' E_{\overline{ID}_t}) + \alpha_{\overline{ID}_{t+1}} L_{t+1}^{(3)}, \\
    E_{\overline{ID}_{t+1}} &= s_t(\tfrac{I_{t+1} I_{t+1}''}{I_{t+1}'} S_{\overline{ID}_t} + I_{t+1}'' E_{\overline{ID}_t} + \widetilde{\alpha}_{t+1} L_{t+1}^{(3)}, \\
    \widetilde{S}_{\overline{ID}_{t+1}} &= s_t(\widetilde{I}_{t+1} \widetilde{S}_{\overline{ID}_t} + \widetilde{I}_{t+1}' \widetilde{E}_{\overline{ID}_t}) + (\alpha_{\overline{ID}_{t+1}} - R_{\overline{ID}_t}) L_{t+1}^{(4)}, \text{ and} \\
    \widetilde{E}_{\overline{ID}_{t+1}} &= s_t(\tfrac{\widetilde{I}_{t+1} \widetilde{I}_{t+1}''}{\widetilde{I}_{t+1}'} \widetilde{S}_{\overline{ID}_t} + \widetilde{I}_{t+1}'' \widetilde{E}_{\overline{ID}_t} + (\widetilde{\alpha}_{t+1} - \widetilde{R}_{\overline{ID}_t}) L_{t+1}^{(4)};
    \end{aligned}
    $$

    Then, $\overline{ID}_t$ gives $d_{\overline{ID}_{t+1}} = (S_{\overline{ID}_{t+1}}, E_{\overline{ID}_{t+1}}, \widetilde{S}_{\overline{ID}_{t+1}}, \widetilde{E}_{\overline{ID}_{t+1}}, R_{\overline{ID}_{t+1}}, \widetilde{R}_{\overline{ID}_{t+1}}, p_{(t+1,2)}, s_{t+1})$ to $\overline{ID}_{t+1}$.

- `Encrypt`: Given a message $m \in \mathcal{M}$, a recipient identifier $\overline{ID}_t$, and the system's public parameters, this algorithm:
  1. picks $a \in_R \mathbb{Z}_q^*$, and computes $r = \mathcal{H}_5(a)$, $U_1 = rP_0^{(4)}$ and $U_2 = rP_0^{(3)}$;
  2. computes $(J_1, \cdots, J_t, J_0, \cdots, J_0) = \mathcal{H}_1(\overline{ID}_t) \in (\mathbb{Z}_q^*)^{3\ell}$, $\rho_t^{(1)} = A_t P_0^{(1)} + B_t P_0^{(2)}$, and $\rho_t^{(2)} = C_t P_0^{(5)} + D_t P_0^{(6)}$, where

    $$
    \begin{aligned}
    &p_{(1,2)} = \mathcal{H}_7(p_{(1,3)}),\ p_{(1,4)} = \tfrac{p_{(1,2)}p_{(1,3)}}{p_{(1,1)}},\ I_1' = -p_{(1,2)}, \\
    &I_1'' = -\tfrac{p_{(1,1)}}{p_{(1,2)}} I_1', \widetilde{I}_1 = I_1 + p_{(1,1)}, \widetilde{I}_1' = I_1' + p_{(1,2)}, \widetilde{I}_1'' = I_1'' + p_{(1,4)} \\
    &A_1 = I_1, \quad B_1 = I_1', \quad C_1 = \widetilde{I}_1, \quad D_1 = \widetilde{I}_1', \\
    &\widetilde{A}_1 = \tfrac{I_1 I_1''}{I_1'}, \quad \widetilde{B}_1 = I_1'', \quad \widetilde{C}_1 = \tfrac{\widetilde{I}_1 \widetilde{I}_1''}{\widetilde{I}_1'}, \quad \widetilde{D}_1 = \widetilde{I}_1'',
    \end{aligned}
    $$

and, for $1 \leq i < t$,
$I'_{i+1} = -p_{(i,2)}$, $I''_{i+1} = -\frac{p_{(i,2)}p_{(i,3)}}{p_{(i,1)}}$, $\widetilde{I}_{i+1} = I_{i+1} + p_{(i,1)}$,
$\widetilde{I}'_{i+1} = I'_{i+1} + p_{(i,2)}$, $\widetilde{I}''_{i+1} = I''_{i+1} + p_{(i,4)}$;
$p_{(i+1,2)} = -\frac{p_{(i,1)}}{p_{(i,3)}}p_{(i+1,1)}$, $p_{(i+1,4)} = \frac{p_{(i+1,2)}p_{(i+1,3)}}{p_{(i+1,1)}}$,

$$A_{i+1} = I_{i+1}A_i + I'_{i+1}\widetilde{A}_i, \text{ and } \quad \widetilde{A}_{i+1} = \frac{I_{i+1}I''_{i+1}}{I'_{i+1}}A_i + I''_{i+1}\widetilde{A}_i,$$
$$B_{i+1} = I_{i+1}B_i + I'_{i+1}\widetilde{B}_i, \text{ and } \quad \widetilde{B}_{i+1} = \frac{I_{i+1}I''_{i+1}}{I'_{i+1}}B_i + I''_{i+1}\widetilde{B}_i,$$
$$C_{i+1} = \widetilde{I}_{i+1}C_i + \widetilde{I}'_{i+1}\widetilde{C}_i, \text{ and } \quad \widetilde{C}_{i+1} = \frac{\widetilde{I}_{i+1}\widetilde{I}''_{i+1}}{\widetilde{I}'_{i+1}}C_i + \widetilde{I}''_{i+1}\widetilde{C}_i,$$
$$D_{i+1} = \widetilde{I}_{i+1}D_i + \widetilde{I}'_{i+1}\widetilde{D}_i, \text{ and } \quad \widetilde{D}_{i+1} = \frac{\widetilde{I}_{i+1}\widetilde{I}''_{i+1}}{\widetilde{I}'_{i+1}}D_i + \widetilde{I}''_{i+1}\widetilde{D}_i,$$

3. computes $V = m \oplus \mathcal{H}_2\left(\left(\hat{e}(L_t^{(4)}, \rho_t^{(1)}) \cdot \hat{e}(L_t^{(3)}, \rho_t^{(2)})^{-1}\right)^r\right)$;
4. outputs the ciphertext $c = (U_1, U_2, V)$.

- `Decrypt`: Given a ciphertext $c = (U_1, U_2, V) \in \mathcal{C}$, the private key $d_{\overline{ID}_t}$ of a recipient $\overline{ID}_t$, and the system's public parameters, this algorithm computes and outputs the message:

$$m = V \oplus \mathcal{H}_2(\hat{e}(U_1, S_{\overline{ID}_t})\hat{e}(U_2, \widetilde{S}_{\overline{ID}_t})^{-1}).$$

Correctness. For correctness[2], remark that $S_{\overline{ID}_t} = (\prod_{i=1}^{t} s_{i-1})(\rho_t^{(1)} + F_t P_0^{(3)})$ and $\widetilde{S}_{\overline{ID}_t} = (\prod_{i=1}^{t} s_{i-1})(F_t P_0^{(4)} + \rho_r^{(2)})$, for some $F_t \in \mathbb{Z}_q^*$. Hence, for $t \geq 2$:

$$\begin{aligned}
\hat{e}(U_1, S_{\overline{ID}_t}) &= \hat{e}(rP_0^{(4)}, (\prod_{i=1}^{t} s_{i-1})\rho_t^{(1)}) \cdot \hat{e}(rP_0^{(4)}, (\prod_{i=1}^{t} s_{i-1})F_t P_0^{(3)}) \\
&= \hat{e}((\prod_{i=1}^{t} s_{i-1})P_0^{(4)}, \rho_t^{(1)})^r \cdot \hat{e}((\prod_{i=1}^{t} s_{i-1})F_t P_0^{(4)}, rP_0^{(3)}) \\
&= \hat{e}(L_t^{(4)}, \rho_t)^r \cdot \hat{e}(\widetilde{S}_{\overline{ID}_t} - (\prod_{i=1}^{t} s_{i-1})\rho_t^{(2)}, rP_0^{(3)}) \\
&= \hat{e}(L_t^{(4)}, \rho_t)^r \cdot \hat{e}(\widetilde{S}_{\overline{ID}_t}, U_2) \cdot \hat{e}(L_t^{(3)}, \rho_t^{(2)})^{-r}
\end{aligned}$$

Design Justification. The key part of our HIBE scheme is its key generation procedure (*Extract*). Each user (or PKG) $\overline{ID}_t$ is given a private key $d_{\overline{ID}_t}$ of the form $(S_{\overline{ID}_t}, E_{\overline{ID}_t}, \widetilde{S}_{\overline{ID}_t}, \widetilde{E}_{\overline{ID}_t}, s_t)$. The first four components of this key are unique to the user (i.e. to her hierarchical level and to the path from $\overline{ID}_t$ to the root PKG), and the last component of this key is associated with $\overline{ID}_t$'s hierarchical level. $S_{\overline{ID}_t}$ and $E_{\overline{ID}_t}$

[2]The case $t \geq 1$ is straightforward.

are used for decryption, while $\widetilde{S}_{\overline{ID}_t}$ and $\widetilde{E}_{\overline{ID}_t}$ are used for lower-key generation. The coefficients of $P_0^{(1)}$, $P_0^{(2)}$, $P_0^{(5)}$, and $P_0^{(6)}$ in the first four components of the key are chosen so that it is not possible to write $(\prod_{i=1}^{t} s_{i-1})P_0^{(1)}$, $(\prod_{i=1}^{t} s_{i-1})P_0^{(2)}$, $(\prod_{i=1}^{t} s_{i-1})P_0^{(5)}$, or $(\prod_{i=1}^{t} s_{i-1})P_0^{(6)}$ as a linear combination of $S_{\overline{ID}_t}$, $E_{\overline{ID}_t}$, $\widetilde{S}_{\overline{ID}_t}$ and $\widetilde{E}_{\overline{ID}_t}$. Moreover, the use of different coefficients for both $P_0^{(1)}$ and $P_0^{(5)}$, and $P_0^{(2)}$ and $P_0^{(6)}$ prevents to write $(\prod_{i=1}^{t} s_{i-1})(P_0^{(1)} + FP_0^{(3)})$, $(\prod_{i=1}^{t} s_{i-1})(P_0^{(2)} + F'P_0^{(3)})$, $(\prod_{i=1}^{t} s_{i-1})(FP_0^{(4)} + P_0^{(5)})$, or $(\prod_{i=1}^{t} s_{i-1})(F'P_0^{(4)} + P_0^{(6)})$, as a linear combination of $S_{\overline{ID}_t}$, $E_{\overline{ID}_t}$, $\widetilde{S}_{\overline{ID}_t}$, $\widetilde{E}_{\overline{ID}_t}$ and the private key components of any colluding set of $t - level$ users. In either one of the aforementioned cases, $\overline{ID}_t$ would be able to compute any user's private key. Furthermore, $\overline{ID}_t$ should be able to compute $\widetilde{E}_{\overline{ID}_{t+1}}$ and $\widetilde{S}_{\overline{ID}_{t+1}}$ as a linear combination of $\widetilde{E}_{\overline{ID}_t}$, $\widetilde{S}_{\overline{ID}_{t+1}}$ and L_{t+1}^4, using $\alpha_{\overline{ID}_{t+1}}$, $\widetilde{\alpha}_{\overline{ID}_{t+1}}$ and publicly available values. Our key generation procedure meets these design requirements.

NewHIBE-v2

- `Instance Generator`. Same as in NewHIBE-v1.
- `Root Setup`. Same as in NewHIBE-v1, except that the ciphertext space is now defined as $\mathcal{C} = \mathbb{G}_1^2 \times \{0,1\}^n \times \{0,1\}^n$, and that the system parameters now include two public additional hash functions $\mathcal{H}_3 : \{0,1\}^n \times \{0,1\}^n \rightarrow \mathbb{Z}_q^*$ and $\mathcal{H}_4 : \{0,1\}^n \rightarrow \{0,1\}^n$.
- `Extract`: Same as in NewHIBE-v1.
- `Encrypt`: Given a message $m \in \mathcal{M}$, a recipient identifier $\overline{ID}_t$, and the system's public parameters, this algorithm:
  1. picks $\gamma \in_R \{0,1\}^n$, and computes $r = \mathcal{H}_3(\gamma, m)$, $U_1 = rP_0^{(4)}$ and $U_2 = rP_0^{(3)}$;
  2. computes $(J_1, \cdots, J_t, J_0, \cdots, J_0) = \mathcal{H}_1(\overline{ID}_t) \in (\mathbb{Z}_q^*)^{3\ell}$, $\rho_t^{(1)}$, and $\rho_t^{(2)}$ (as in NewHIBE-v1);
  3. computes $V = \gamma \oplus \mathcal{H}_2\left(\left(\hat{e}(L_t^{(4)}, \rho_t^{(1)}) \cdot \hat{e}(L_t^{(3)}, \rho_t^{(2)})^{-1}\right)^r\right)$ and $W = m \oplus \mathcal{H}_4(\gamma)$;
  4. outputs the ciphertext $c = (U_1, U_2, V, W)$.
- `Decrypt`: Given a ciphertext $c = (U_1, U_2, V, W) \in \mathcal{C}$, the private key $d_{\overline{ID}_t}$ of a recipient $\overline{ID}_t$, and the system's public parameters, this algorithm:
  1. computes $\gamma = V \oplus \mathcal{H}_2(\hat{e}(U_1, S_{\overline{ID}_t})\hat{e}(U_2, \widetilde{S}_{\overline{ID}_t})^{-1})$;

2. computes $W \oplus \mathcal{H}_4(\gamma) = m$;
3. sets $r = \mathcal{H}_3(\gamma, m)$, and verify whether (U_1, U_2, V) is a NewHIBE-v1 encryption of m with r. If this is not the case, c is rejected; otherwise, the algorithm outputs m as the decryption of c.

4.4 Proposed Signature Scheme

Our proposed hierarchical ID-based signature scheme is named NewHIBS.

- `Instance Generator` (k). Same as in NewHIBE-v1.
- `Root Setup` (k). Same as in NewHIBE-v1, except that a signature space $\mathcal{S} = \mathbb{G}_1^3$ is defined instead of the ciphertext space $\mathcal{C}$.
- `Extract`. Same as in NewHIBE-v1.
- `Signature`. Given a message $m \in \mathcal{M}$, the private key $d_{\overline{ID}_t} = (S_{\overline{ID}_t}, E_{\overline{ID}_t}, \widetilde{S}_{\overline{ID}_t}, s_t)$ of a signer $\overline{ID}_t$, and the system's public parameters, this algorithm:
  1. picks $y \in_R \mathbb{Z}_q^*$ and computes $V = y(P_0^{(4)} - P_0^{(3)})$;
  2. computes $\underline{S}_{\overline{ID}_t} = y\mathcal{H}_1(m) + S_{\overline{ID}_t}$ and $\underline{\widetilde{S}}_{\overline{ID}_t} = y\mathcal{H}_1(m) + \widetilde{S}_{\overline{ID}_t}$;
  3. outputs $\sigma = (\underline{S}_{\overline{ID}_t}, \underline{\widetilde{S}}_{\overline{ID}_t}, V)$.
- `Verification`. Given a message $m \in \mathcal{M}$, a signature $\sigma = (\underline{S}_{\overline{ID}_t}, \underline{\widetilde{S}}_{\overline{ID}_t})$, the ID-tuple $\overline{ID}_t$ of a claimed signer, and the system's public parameters, this algorithm computes $\rho_t^{(1)}$ and $\rho_t^{(2)}$ as in NewHIBE-v1's *Encryption* procedure, and returns (m, "Valid") if $\hat{e}(L_t^{(4)}, \rho_t^{(1)}) \cdot \hat{e}(V, \mathcal{H}_1(m)) \cdot \hat{e}(P_0^{(3)}, \underline{\widetilde{S}}_{\overline{ID}_t}) = \hat{e}(P_0^{(4)}, \underline{S}_{\overline{ID}_t}) \cdot \hat{e}(L_t^{(3)}, \rho_t^{(2)})\cdot$, or "Invalid" otherwise.

4.5 Efficiency

Table 4.1 compares the computational and storage requirements of BF-IBE, GS-HIBE, BBG-HIBE, and our HIBE scheme. t denotes both the hierarchical level of a ciphertext's intended recipient and the hierarchical level of a user whose key is to be extracted. Moreover, it is assumed that a binary exclusive OR (i.e. XOR) operation always takes the same time, regardless of its arguments' length. It is also assumed that all hash functions of the compared schemes have the same computational cost, denoted by H. Xor denotes the computational cost of XOR. M_X and A_X respectively denote computational costs of scalar multiplication and addition in the Abelian group

Schemes		Storage Requirements	Computational Requirements
BF-IBE	Public Parameters	11	$M_{G_1} + R_{Z_q^*}$
	Extract	1	M_{G_1}
	Encryption	4	$Ex_{G_2} + 4H + M_{G_1} + \mathbf{P} + R_{\{0,1\}^n} + 2Xor$
	Decryption	3	$3H + M_{G_1} + \mathbf{P} + 2Xor$
GS-HIBE	Public Parameters	11	$M_{G_1} + R_{Z_q^*}$
	Extract	$t+1$	$A_{G_1} + H + 2M_{G_1} + R_{Z_q^*}$
	Encryption	$t+3$	$Ex_{G_2} + (t+1)H + tM_{G_1} + \mathbf{P} + R_{\{0,1\}^n} + 2Xor$
	Decryption	$t+8$	$Ex_{G_2} + (t+3)H + Inv_{G_2} + tM_{G_1} + (t-1)M_{G_2} + (t+1)\mathbf{P} + R_{Z_q^*} + 3Xor$
BBG-HIBE	Public Parameters	$5+\ell$	$(\ell+2)H + 2M_{G_1} + R_{Z_q^*} + R_{G_1}$
	Extract	$3+\ell-t$	$(\ell+2)A_{G_1} + H + (\ell+2)M_{G_1} + R_{Z_q^*}$
	Encryption	$t+5$	$tA_{G_1} + Ex_{G_2} + tH + (t+2)M_{G_1} + M_{G_2}$
	Decryption	3	$Inv_{G_2} + 2M_{G_2} + 2\mathbf{P}$
NewHIBE-v2	Public Parameters	$15+2\ell$	$(\ell-1)H + 2\ell M_{G_1} + 6R_{G_1} + 2R_{Z_q^*}$
	Extract	24	$8A_{G_1} + 7A_{Z_q^*} + 6H + 3Inv_{Z_q^*} + 12M_{G_1} + 22M_{Z_q^*} + 2R_{Z_q^*}$
	Encryption	$3t+24$	$2A_{G_1} + (11t-8)A_{Z_q^*} + Ex_{G_2} + (3t+4)H + Inv_{G_2} + (5t-1)Inv_{Z_q^*} + 6M_{G_1} + M_{G_2} + (24t-16)M_{Z_q^*} + 2\mathbf{P} + R_{\{0,1\}^n} + 2Xor$
	Decryption	10	$Ex_{G_2} + 4H + Inv_{G_2} + 2M_{G_1} + M_{G_2} + 2\mathbf{P} + 3Xor$

Table 4.1: Efficiency Comparison of BF-IBE, GS-HIBE, BBG-HIBE, and NewHIBE

		Computational Requirements					
		$t = 1$ $\ell = 1$	$t = 3$ $\ell = 4$	$t = 4$ $\ell = 5$	$t = 10$ $\ell = 11$	$t = 12$ $\ell = 13$	$t = 20$ $\ell = 21$
BF-IBE	Public Parameters	21×10^3	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
	Extract	21×10^3	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
	Encryption	105×10^3	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
	Decryption	105×10^3	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
GS-HIBE	Public Parameters	21×10^3	21×10^3	21×10^3	21×10^3	21×10^3	21×10^3
	Extract	42×10^3	42×10^3	42×10^3	42×10^3	42×10^3	42×10^3
	Encryption	105×10^3	147×10^3	168×10^3	294×10^3	336×10^3	504×10^3
	Decryption	189×10^3	399×10^3	504×10^3	1.13×10^6	1.34×10^6	2.18×10^6
BBG-HIBE	Public Parameters	42×10^3	42×10^3	42×10^3	42×10^3	42×10^3	42×10^3
	Extract	64×10^3	127×10^3	148×10^3	276×10^3	318×10^3	488×10^3
	Encryption	63×10^3	106×10^3	127×10^3	254×10^3	297×10^3	466×10^3
	Decryption	168×10^3	168×10^3	168×10^3	168×10^3	168×10^3	168×10^3
NewHIBE-v2	Public Parameters	84×10^3	168×10^3	210×10^3	462×10^3	546×10^3	882×10^3
	Extract	255×10^3	255×10^3	255×10^3	255×10^3	255×10^3	255×10^3
	Encryption	296×10^3	299×10^3	300×10^3	310×10^3	313×10^3	325×10^3
	Decryption	210×10^3	210×10^3	210×10^3	210×10^3	210×10^3	210×10^3

Table 4.2: Timing Estimates of BF-IBE, GS-HIBE, BBG-HIBE, and NewHIBE in Microseconds.

Schemes		Storage Requirements	Computational Requirements
CC-IBS	Public Parameters	4	$M_{\mathbb{G}_1} + R_{\mathbb{Z}_q^*}$
	Extract	1	$H + M_{\mathbb{G}_1}$
	Signature	3	$A_{\mathbb{Z}_q^*} + H + 2M_{\mathbb{G}_1} + R_{\mathbb{Z}_q^*}$
	Verification	3	$A_{\mathbb{G}_1} + 2H + M_{\mathbb{G}_1} + 2\mathbf{P}$
GS-HIBS	Public Parameters	11	$M_{\mathbb{G}_1} + R_{\mathbb{Z}_q^*}$
	Extract	$t+1$	$A_{\mathbb{G}_1} + H + 2M_{\mathbb{G}_1} + R_{\mathbb{Z}_q^*}$
	Signature	3	$A_{\mathbb{G}_1} + H + 2M_{\mathbb{G}_1}$
	Verification	$t+2$	$H + (t-1)M_{\mathbb{G}_2} + (t+1)\mathbf{P}$
NewHIBS	Public Parameters	$14 + 2\ell$	$(\ell-1)H + 2\ell M_{\mathbb{G}_1} + 6R_{\mathbb{G}_1} + R_{\mathbb{Z}_q^*}$
	Extract	24	$8A_{\mathbb{G}_1} + 7A_{\mathbb{Z}_q^*} + 6H + 3Inv_{\mathbb{Z}_q^*} + 12M_{\mathbb{G}_1} + 22M_{\mathbb{Z}_q^*} + 2R_{\mathbb{Z}_q^*}$
	Signature	5	$3A_{\mathbb{G}_1} + H + 2M_{\mathbb{G}_1} + R_{\mathbb{Z}_q^*}$
	Verification	$3t + 17$	$2A_{\mathbb{G}_1} + (11t-8)A_{\mathbb{Z}_q^*} + (3t+1)H + (5t-1)Inv_{\mathbb{Z}_q^*} + 4M_{\mathbb{G}_1} + 3M_{\mathbb{G}_2} + (24t-16)M_{\mathbb{Z}_q^*} + 5\mathbf{P}$

Table 4.3: Efficiency Comparison of CC-IBS, GS-HIBS and NewHIBS.

		Computational Requirements					
		$t = 1$ $\ell = 1$	$t = 3$ $\ell = 4$	$t = 4$ $\ell = 5$	$t = 10$ $\ell = 11$	$t = 12$ $\ell = 13$	$t = 20$ $\ell = 21$
CC-IBS	Public Parameters	21×10^3	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
	Extract	21×10^3	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
	Signature	42×10^3	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
	Verification	189×10^3	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
GS-HIBS	Public Parameters	21×10^3	21×10^3	21×10^3	21×10^3	21×10^3	21×10^3
	Extract	42×10^3	42×10^3	42×10^3	42×10^3	42×10^3	42×10^3
	Signature	42×10^3	42×10^3	42×10^3	42×10^3	42×10^3	42×10^3
	Verification	168×10^3	336×10^3	420×10^3	924×10^3	1.1×10^6	1.8×10^6
NewHIBS	Public Parameters	42×10^3	168×10^3	210×10^3	462×10^3	546×10^3	882×10^3
	Extract	255×10^3	255×10^3	255×10^3	255×10^3	255×10^3	255×10^3
	Signature	43×10^3	43×10^3	43×10^3	43×10^3	43×10^3	43×10^3
	Verification	506×10^3	509×10^3	510×10^3	520×10^3	523×10^3	535×10^3

Table 4.4: Timing Estimates of CC-IBS, GS-HIBS and NewHIBS in microseconds.

X. Inv_X denotes the computational cost of inversion in the field X. R_X denotes the computational cost of uniformly selecting a random element in the set X. The computational cost of exponentiation in the group X is denoted by Ex_X, and $\mathbf{P}$ denotes the computational cost of a bilinear pairing operation. For the storage costs, it is assumed that the result of each computation can be stored in a storage unit. Moreover, the storage requirements column includes all temporary variables required to perform the operations of each algorithm. Table 3.2 presents timing estimates of BF-IBE, GS-HIBE, BBG-HIBE, and our HIBE scheme, based information obtained from [HPS02].

Before all, let us recall that NewHIBE issues ciphertexts with constant length. BBG-HIBE features ciphertexts consisting of 3 group elements, compared with 4 group elements in the case of NewHIBE-v2 (NewHIBE). NewHIBE's setup algorithm also involves $O(\ell)$ more scalar multiplications in $\mathbb{G}_1$ than BBG-HIBE's. (This is a one-time cost.)

Note also that NewHIBE's encryption algorithm requires two more pairing than its BBG-HIBE analogue. On the other hand, NewHIBE encryption's algorithm replaces many scalar multiplication and additions in $\mathbb{G}_1$ by multiplications and inversions in $\mathbb{Z}_q^*$. This results in a gain when $\mathbb{G}_1$ is instantiated by a group of elliptic curve points and operations in $\mathbb{G}_1$ are not replaced by operations in $\mathbb{G}_2$ (e.g. via the applications of two inverse maps). For instance [HPS02], if $P \approx 4M_{\mathbb{G}_1}$, $M_{\mathbb{G}_1} \approx 1400M_{\mathbb{Z}_q^*}$, $A_{\mathbb{G}_1} \approx 14M_{\mathbb{Z}_q^*}$, and $Inv_{\mathbb{Z}_q^*} \approx 16M_{\mathbb{Z}_q^*}$, then NewHIBE's encryption algorithm is advantageous when $t \geq 12$. Hence, NewHIBE's encryption algorithm is faster than its BBG-HIBE analogue for relatively large values of t.

However, BBG-HIDE and NewHIBE have almost identical computational cost for decryption (i.e. $2\mathbf{P}$). Furthermore, NewHIBE's key extraction procedure takes a constant time to execute, and involves $O(\ell)$ fewer additions and scalar multiplications in $\mathbb{G}_1$ than BBG-HIBE's. The key extraction storage cost of NewHIBE is also constant, compared with $O(\ell - t)$ in the case of BBG-HIBE.

Compared with GS-HIBE, both BBG-HIBE and NewHIBE achieve significant ($O(1)$ versus $O(t)$) improvements with respect to decryption time.

Hence: (1) BBG-HIBE is faster than NewHIBE with respect to encryption (for $t \leq 12$); (2) BBG-HIBE and NewHIBE have similar performance with respect to decryption; (3) NewHIBE is faster than BBG-HIBE with respect to key generation.

Table 4.3 compares the computational and storage requirements of Cha and Cheon's ID-based signature scheme[3] (CC-IBS), Gentry and Silverberg's HIBS scheme (GS-HIBS), and our proposed HIBS scheme (NewHIBS). Table 4.4 presents corresponding

[3]This scheme is the fastest known ID-based signature signature scheme.

timing estimates based information obtained in [HPS02]. While the *Signing* procedures of GS-HIBS and NewHIBS have essentially the same space and computational requirements, the verification procedure of NewHIBS is significantly less computationally expensive than GS-HIBS (each pairing is essentially replaced by 24 multiplications in $\mathbb{Z}_q^*$). Note, furthermore, that no hierarchical ID-based signature scheme with constant-size signatures had been described before NewHIBS.

4.6 Security

This section presents the security guarantees of our proposed HIBE scheme. The notions of chosen ciphertext security and chosen plaintext security for hierarchical ID-based systems (cf. [GS02]) are defined in §4.6.1 and §4.6.2. Then, §4.6.3 shows that our proposed HIBE scheme is semantically secure with respect to adaptive chosen plaintext attacks, in the random oracle model, if the BDH problem is hard.

4.6.1 Chosen Ciphertext Security

Let Ψ be a *HIBE* scheme. Then, the following game, initiated by a challenger $\mathcal{C}h$ against an attacker $\mathcal{A}$, may then be considered:

Setup: From a security parameter k, $\mathcal{C}h$ uses Ψ's *Root Setup* algorithm to generate the cryptosystem's public and private parameters - keeping the private parameters secret while giving the public ones to $\mathcal{A}$.

Phase 1: $\mathcal{A}$ issues to $\mathcal{C}h$ a polynomially bounded number of queries of any of the following types:

- *Public-Key query*: Given an ID-tuple $\overline{ID}_{(i,t_i)} = (ID_{(i,1)}, \cdots, ID_{(i,t_i)})$, $\mathcal{C}h$ must return $H_1(\overline{ID}_{(i,t_i)})$.
- *Key Extraction query*: Given an ID-tuple $\overline{ID}_{(i,t_i)}$ (as above), $\mathcal{C}h$ must return the private key $d_{\overline{ID}_{(i,t_i)}}$ of $\overline{ID}_{(i,t_i)}$.
- *Decryption query*: Given an ID-tuple $\overline{ID}_{(i,t_i)}$ and a ciphertext c_i, $\mathcal{C}h$ must return a message $m_i \in \mathcal{M}$ whose encryption under $\overline{ID}_{(i,t_i)}$ is c_i.

The above queries may be issued adaptively: each request may depend on its predecessors.

Challenge: Once *Phase 1* is over, $\mathcal{A}$ issues an ID-tuple $\overline{ID}_t^* = (ID_1, \cdots, ID_t)$ of its choice, and a pair (m_0, m_1) of equal-length plaintexts, such that, in *Phase 1*, no *Key Extraction* queries were issued on $\overline{ID}_t^*$ or any of its ancestors. $\mathcal{C}h$

then picks a random bit $b \in \{0,1\}$, computes the encryption $c^* = \text{Encrypt}(m_b, \overline{ID}_t^*, params)$ of m_b with $\overline{ID}_t^*$, and sends c^* to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ issues a polynomially bounded number of *Phase 1* types of queries, under the following restrictions:

- No *Key Extraction* queries are issued on $\overline{ID}_t^*$ or any of its ancestors.
- If a *Decryption* query is issued with c^* as an argument, then the corresponding ID-tuple can be neither $\overline{ID}_t^*$ nor any of its ancestors.

The queries may be issued adaptively: each request may depend on its predecessors.

Guess: Once *Phase 2* is over, $\mathcal{A}$ submits a guess bit b' and wins the game if $b' = b$.

Definition 4.6.1.1. *The above game is called the* IND-HIB-CCA *attack game. The quantity* $|Pr[b' = b] - \frac{1}{2}|$ *- representing the advantage of* $\mathcal{A}$ *over any challenger* $\mathcal{C}h$ *in the game - is denoted by* $Adv_{\mathcal{A},\mathcal{C}h}^{HIB-CCA}$. *A* HIBE *scheme is said to be secure against adaptive chosen ciphertext attacks (*IND-HIB-CCA *secure, in short) if no polynomially bounded attacker can be found that has non-negligible advantage in the above* IND-HIB-CCA *game.*

4.6.2 Chosen Plaintext Security

The notion of chosen plaintext security is similar to (and weaker than) the notion of *chosen ciphertext security.* To define semantic security for HIBE schemes, one may consider a game (called the *IND-HIB-CPA* game), which is identical to the *IND-HIB-CCA* game, except that the attacker $\mathcal{A}$ is unable to issue *Decryption* queries.

Definition 4.6.2.1. *The value* $|Pr[b' = b] - \frac{1}{2}|$ *- representing the advantage of* $\mathcal{A}$ *over any challenger* $\mathcal{C}h$ *in the* IND-HIB-CPA *game - is denoted by* $Adv_{\mathcal{A},\mathcal{C}h}^{HIB-CPA}$. *A* HIBE *scheme is said to be secure against adaptive chosen plaintext attacks (*IND-HIB-CPA *secure, in short) if no polynomially bounded attacker can be found that has non-negligible advantage in a* IND-HIB-CPA *game.*

4.6.3 Security Theorem

Theorem 4.6.1. *Let* T *be a tree-shaped hierarchy whose depth is logarithmic*[4] *in a security parameter k. Assume that, in the NewHIBE-v1 scheme, all hash functions are random oracles. Suppose also that there exists an attacker $\mathcal{A}$ which has non-negligible advantage $\varepsilon(k)$, in time τ, against any challenger Ch, in the* IND-HIB-CPA *game. Then, there exists an algorithm $\mathcal{B}$ which solves the BDH problem, in time $O(\tau)$, with non-negligible advantage at least*

$$\varepsilon(k)\frac{1}{pol(k)}(1-\delta)^{n_{ext}}, \text{where}$$

δ is the largest proportion of ID-tuples having a common root branch, pol is a polynomial such that $\frac{1}{p(k)}$ is a lower bound of all proportions of ID-tuples having a common root branch, n_{ext} is the number of Extraction *queries issued by $\mathcal{A}$ in any* IND-HIB-CPA *game against $\mathcal{B}$ (as challenger).*

Practical Meaning of Theorem 4.6.1: In practice, this theorem indicates that, when a message is encrypted for a user, only this user and her hierarchical ancestors can decrypt the corresponding ciphertext.

Proof. In this proof, we show how to construct $\mathcal{B}$ using $\mathcal{A}$. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two Abelian groups of prime order q, where $\mathbb{G}_1$ is additive and $\mathbb{G}_2$ is multiplicative. Let $P_0^{(1)} \in \mathbb{G}_1^*$ be a generator of $\mathbb{G}_1$, and ν, ω, π, ς, and ϑ be positive integers such that $P_0^{(2)} = \nu P_0^{(1)}$, $P_0^{(3)} = \omega P_0^{(1)}$, $P_0^{(4)} = \pi P_0^{(1)}$, $P_0^{(5)} = \varsigma P_0^{(1)}$, and $P_0^{(6)} = \vartheta P_0^{(1)}$ are other generators of $\mathbb{G}_1$ (i.e. distinct from one another and from $P_0^{(1)}$). Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be a bilinear pairing on $\mathbb{G}_1$, such that the BDH is assumed to be hard with respect to $\hat{e}$, and let $(\mathbb{G}_1, q, P_0^{(1)}, aP_0^{(1)}, bP_0^{(1)}, cP_0^{(1)})$ be a tuple for which a, b, c are unknown to $\mathcal{B}$. $\mathcal{B}$'s goal is to solve the BDH Problem by computing $\hat{e}(P_0^{(1)}, P_0^{(1)})^{abc}$ in polynomial time. To achieve this goal, $\mathcal{B}$ initiates an *IND-HIB-CPA* game with $\mathcal{A}$, using an arbitrary security parameter k.

$\mathcal{B}$ defines three hash functions $\mathcal{H}_1^{sim} : \{0,1\}^* \rightarrow (\mathbb{Z}_q^*)^{3\ell}$, $\mathcal{H}_2^{sim} : \mathbb{G}_2 \rightarrow \{0,1\}^n$, and $\mathcal{H}_5^{sim} : \mathbb{Z}_q^* \rightarrow \mathbb{Z}_q^*$, over which it has full control. $\mathcal{B}$ sets $n = poly_1(k)$, $\ell = \log(k)$, $I_0 \in_R \mathbb{Z}_q^*$, $L_1^{(3)} = \omega(aP_0^{(1)})$, $L_1^{(4)} = \pi(aP_0^{(1)})$, $s_1 \in_R \mathbb{Z}_q^*$, $s_i = \mathcal{H}_5^{sim}(s_{i-1})$ for $2 \leq i \leq \ell - 1$, and $L_i^{(j)} = s_{i-1}L_{i-1}^{(j)}$ for $2 \leq i \leq \ell$ and $j = 3, 4$. $\mathcal{B}$ defines $\mathcal{M} = \{0,1\}^n$, $\mathcal{C} = \mathbb{G}_1^2 \times \{0,1\}^n$, $pubParams = (q, n, \hat{e}, I_0, P_0^{(1)}, P_0^{(2)}, P_0^{(3)}, P_0^{(4)}, P_0^{(5)}, P_0^{(6)}, \mathcal{H}_1^{sim}, \mathcal{H}_2^{sim}, \mathcal{H}_5^{sim}, ((L_i^{(j)})_{i=1}^{\ell})_{j=3}^{4})$,

[4]The proof of this theorem and those which derive from it are built in such a way that, if the hierarchy's depth is not logarithmic in k, then the number of steps required by the simulator $\mathcal{B}$ is exponential. When this is the case, the BDH problem cannot be solved in polynomial time. Hence, hierarchies must have depth logarithmic in k. However, in practice, k can be chosen in such a way that the probability of a successful attack is very low.

and $params = (pubParams, s_0)$, where s_0 is unknown to $\mathcal{B}$. $\mathcal{B}$ sends $pubParams$ to $\mathcal{A}$, and keeps $(s_i)_{i=1}^{\ell}$ secret.

The rest of the proof consists of three sections. The *Hash Simulation* section shows how $\mathcal{B}$ simulates $\mathcal{H}_1^{sim}$, $\mathcal{H}_2^{sim}$ and $\mathcal{H}_5^{sim}$. The *Attack Games* section explains how $\mathcal{B}$ handles the queries of the attack game. Finally, the *Complexity and Probability* section derives the complexity and probability results of the theorem.

Hash Simulation:

In order to store information about the queries of the *IND-HIB-CPA* game, $\mathcal{B}$ maintains a list $\Lambda_{\mathcal{H}_1^{sim}}$ consisting of entries of the form: $(\overline{ID}_{(i,t_i)}, (p_{(i,d,1)})_{d=1}^{t_i}, (p_{(i,d,3)})_{d=1}^{t_i}$, $(I_{(i,d)})_{d=1}^{t_i}, (I'_{(i,d)})_{d=1}^{t_i}, (I''_{(i,d)})_{d=1}^{t_i}, (\widetilde{I}_{(i,d)})_{d=1}^{t_i}, (\widetilde{I}'_{(i,d)})_{d=1}^{t_i}, (\widetilde{I}''_{(i,d)})_{d=1}^{t_i}, (\alpha_{(i,d)})_{d=1}^{t_i}, (\widetilde{\alpha}_{(i,d)})_{d=1}^{t_i}$, $(S_{\overline{ID}_{(i,d)}})_{d=1}^{t_i}, (E_{\overline{ID}_{(i,d)}})_{d=1}^{t_i}, (\widetilde{S}_{\overline{ID}_{(i,d)}})_{d=1}^{t_i}, (\widetilde{E}_{\overline{ID}_{(i,d)}})_{d=1}^{t_i})$, where $\overline{ID}_{(i,t_i)} = (ID_{(i,1)}, \cdots, ID_{(i,t_i)})$ is an ID-tuple of the user hierarchy, and, for $1 \leq d \leq t_i$, $(p_{(i,d,1)})_{d=1}^{t_i}, (p_{(i,d,3)})_{d=1}^{t_i}, I_{(i,d)}, I'_{(i,d)}, I''_{(i,d)}, \widetilde{I}_{(i,d)}, \widetilde{I}'_{(i,d)}, \widetilde{I}''_{(i,d)}, \alpha_{(i,d)}, \widetilde{\alpha}_{(i,d)} \in \mathbb{Z}_q^*$, $S_{\overline{ID}_{(i,d)}}$, $E_{\overline{ID}_{(i,d)}}, \widetilde{S}_{\overline{ID}_{(i,d)}} \in \mathbb{G}_1$, and $\widetilde{E}_{\overline{ID}_{(i,d)}} \in \mathbb{G}_1$ are the simulated values of $\overline{ID}_{(i,d)}$'s key components.

Let $n_{\mathcal{H}_1}$ denote the number of $\mathcal{H}_1$ queries. Every $\mathcal{H}_1$-query is issued on an ID-tuple whose first component we call the *root branch*. Let then n_{rb} denote the number of distinct root branches appearing in $\mathcal{H}_1$-query ID-tuples. Every distinct root branch is indexed using an integer between 1 and n_{rb}. Let μ be a uniformly random element of $\{1, \cdots, n_{rb}\}$. For $\mathcal{B}$ to win the *IND-HIB-CPA* attack game, the index of the ID-tuple associated with the challenge plaintext pair will need to be μ.

To answer $\mathcal{H}_1$ and *Extraction* queries, $\mathcal{B}$ builds $\Lambda_{\mathcal{H}_1^{sim}}$ as follows. Let $\overline{ID}_{(i,t_i)} = (ID_{(i,1)}, \cdots, ID_{(i,t_i)})$ be the ID-tuple associated with the i^{th} $\mathcal{H}_1$-query and let y be maximal such that $(ID_{(i,1)}, \cdots, ID_{(i,y)}) = (ID_{(j,1)}, \cdots, ID_{(j,y)})$, where $(ID_{(j,1)}, \cdots, ID_{(j,t_j)})$ is an ID-tuple which is already in $\Lambda_{\mathcal{H}_1^{sim}}$. Then:

- For $1 \leq d \leq y$: $p_{(i,d,1)} = p_{(j,d,1)}$, $p_{(i,d,3)} = p_{(j,d,3)}$, $I_{(i,d)} = I_{(j,d)}$, $I'_{(i,d)} = I'_{(j,d)}$, $I''_{(i,d)} = I''_{(j,d)}$, $\widetilde{I}_{(i,d)} = \widetilde{I}_{(j,d)}$, $\widetilde{I}'_{(i,d)} = \widetilde{I}'_{(j,d)}$, $\widetilde{I}''_{(i,d)} = \widetilde{I}''_{(j,d)}$, $\alpha_{(i,d)} = \alpha_{(j,d)}$, $\widetilde{\alpha}_{(i,d)} = \widetilde{\alpha}_{(j,d)}$, $S_{(i,d)} = S_{(j,d)}$, $E_{(i,d)} = E_{(j,d)}$, $\widetilde{S}_{(i,d)} = \widetilde{S}_{(j,d)}$, and $\widetilde{E}_{(i,d)} = \widetilde{E}_{(j,d)}$;
- For $y < d \leq t_i$:
  1. If $d = 1$:
     - If $\overline{ID}_{(i,t_i)}$ is the first ID-tuple whose root branch index is μ, then $I_{(i,1)}, p_{(i,1,1)}, p_{(i,1,3)} \in_R \mathbb{Z}_q^*$, $\alpha_{(i,1)} = \widetilde{\alpha}_{(i,1)} = 1$, and $S_{(i,1)} = E_{(i,1)} = \widetilde{S}_{(i,1)} = \widetilde{E}_{(i,1)} = Id_{\mathbb{G}_1}$ (where 1 and the neutral element $Id_{\mathbb{G}_1}$ of $\mathbb{G}_1$ are default values). The other components (i.e. $I'_{(i,1)}$, etc) are constructed according to the method described in the *Key Extraction* algorithm.

- Otherwise, $I_{(i,1)}, p_{(i,1,1)}, p_{(i,1,3)}\alpha_{(i,1)}, \widetilde{\alpha}_{(i,1)} \in_R \mathbb{Z}_q^*$, and the other components (except $S_{(i,1)}$, $E_{(i,1)}$, $\widetilde{S}_{(i,1)}$, and $\widetilde{E}_{(i,1)}$) are computed according to the method described in the *Key Extraction* algorithm. Moreover, $S_{\overline{ID}_1} = I_{(i,1)}(aP_0^{(1)}) + I'_{(i,1)}\nu(aP_0^{(1)}) + \alpha_{\overline{ID}_1}L_1^{(3)}$, $E_{\overline{ID}_1} = \frac{I_{(i,1)}I''_{(i,1)}}{I'_{(i,1)}}(aP_0^{(1)}) + I''_{(i,1)}\nu(aP_0^{(1)}) + \widetilde{\alpha}_{\overline{ID}_1}L_1^{(3)}$, and $\widetilde{S}_{\overline{ID}_1} = \alpha_{\overline{ID}_1}\pi(aP_0^{(1)}) + \widetilde{I}_{(i,1)}\varsigma(aP_0^{(1)}) + \widetilde{I}'_{(i,1)}\vartheta(aP_0^{(1)})$, $\widetilde{E}_{\overline{ID}_1} = \widetilde{\alpha}_{\overline{ID}_1}\pi(aP_0^{(1)}) + \frac{\widetilde{I}_{(i,1)}\widetilde{I}''_{(i,1)}}{\widetilde{I}'_{(i,1)}}\varsigma(aP_0^{(1)}) + \widetilde{I}''_{(i,1)}\vartheta(aP_0^{(1)})$.

2. Otherwise (i.e. if $2 \leq d \leq t_i$), $I_{(i,d)}, p_{(i,d,1)}, p_{(i,d,3)}\alpha_{(i,d)}, \widetilde{\alpha}_{(i,d)} \in_R \mathbb{Z}_q^*$, and the other components (including the I_i's, $S_{(i,d)}$, $E_{(i,d)}$, $\widetilde{S}_{(i,d)}$, and $\widetilde{E}_{(i,d)}$) are computed according to the method described in the *Key Extraction* algorithm.

- $\mathcal{B}$ adds the entry $(\overline{ID}_{(i,t_i)}, (p_{(i,d,1)})_{d=1}^{t_i}, (p_{(i,d,3)})_{d=1}^{t_i}, (I_{(i,d)})_{d=1}^{t_i}, (I'_{(i,d)})_{d=1}^{t_i}, (I''_{(i,d)})_{d=1}^{t_i}, (\widetilde{I}_{(i,d)})_{d=1}^{t_i}, (\widetilde{I}'_{(i,d)})_{d=1}^{t_i}, (\widetilde{I}''_{(i,d)})_{d=1}^{t_i}, (\alpha_{(i,d)})_{d=1}^{t_i}, (\widetilde{\alpha}_{(i,d)})_{d=1}^{t_i}, (S_{\overline{ID}_{(i,d)}})_{d=1}^{t_i}, (E_{\overline{ID}_{(i,d)}})_{d=1}^{t_i}, (\widetilde{S}_{\overline{ID}_{(i,d)}})_{d=1}^{t_i}, (\widetilde{E}_{\overline{ID}_{(i,d)}})_{d=1}^{t_i})$ to the list $\Lambda_{\mathcal{H}_1^{sim}}$.

For $\mathcal{H}_5^{sim}$-queries, $\mathcal{B}$ maintains a list $\Lambda_{\mathcal{H}_5^{sim}}$ whose entries have the form $(\alpha, h_{5,\alpha})$, where $\alpha \in \mathbb{Z}_q^*$ and $h_{2,\alpha} \in \mathbb{Z}_q^*$ is the simulated value of $\mathcal{H}_5(\alpha)$. Thus, on input $\alpha \in \mathbb{Z}_q^*$, $\mathcal{B}$ proceeds as follows:

- If $\Lambda_{\mathcal{H}_5^{sim}}$ already has an entry $(\alpha, h_{5,\alpha})$, then $\mathcal{B}$ returns $h_{5,\alpha}$ as an answer to the $\mathcal{H}_2^{sim}$-query;
- Otherwise, $\mathcal{B}$ computes picks $h_{5,\alpha} \in_R \mathbb{Z}_q^*$, adds $(\alpha, h_{5,\alpha})$ to $\Lambda_{\mathcal{H}_5^{sim}}$, and returns $h_{5,\alpha}$ as an answer to the $\mathcal{H}_5^{sim}$ query.

For $\mathcal{H}_2^{sim}$-queries, $\mathcal{B}$ maintains a list $\Lambda_{\mathcal{H}_2^{sim}}$ whose entries have the form $(\alpha, h_{2,\alpha})$, where $\alpha \in \mathbb{G}_2$ and $h_{2,\alpha} \in \{0,1\}^n$ is the simulated value of $\mathcal{H}_2(\alpha)$. Thus, on input $\alpha \in \mathbb{G}_2$, $\mathcal{B}$ proceeds as follows:

- If $\Lambda_{\mathcal{H}_2^{sim}}$ already has an entry $(\alpha, h_{2,\alpha})$, then $\mathcal{B}$ returns $h_{2,\alpha}$ as an answer to the $\mathcal{H}_2^{sim}$-query;
- If $\alpha = \left(\hat{e}(L_t^{(4)}, \rho_t^{(1)}) \cdot \hat{e}(L_t^{(3)}, \rho_t^{(2)})^{-1}\right)^r$, where r is an output of $\mathcal{H}_5^{sim}$ and $\rho_t^{(1)}, \rho_t^{(2)}$ are associated with a user $\overline{ID}_t$ whose root branch index is μ then, $\mathcal{B}$ proceeds as follows:
  - $\mathcal{B}$ computes $\left(\hat{e}(L_t^{(4)}, \widetilde{\rho}_t^{(1)}) \cdot \hat{e}(L_t^{(3)}, \rho_t^{(2)})^{-1}\right)^r$ where $\widetilde{\rho}_t^{(1)}$ is obtained, for the same user $\overline{ID}_t$, using the following variant of the recursive algorithm meant

to compute A_t: $A_1 = bP_0^{(1)} \in \mathbb{G}_1$; $\widetilde{A}_1 = \frac{I''_{(i,1)}}{I'_{(i,1)}} A_1 \in \mathbb{G}_1$; for $1 \leq i < t$, $A_{i+1} = I_{i+1}A_i + I'_{t+1}\widetilde{A}_i$ and $\widetilde{A}_{i+1} = \frac{I_{i+1}I''_{i+1}}{I'_{i+1}} A_i + I''_{i+1}\widetilde{A}_i$.

- $\mathcal{B}$ computes $h_{2,\alpha} = \hbar(\alpha)$ (where $\hbar$ is a random oracle), adds $(\alpha, h_{2,\alpha})$ to $\Lambda_{\mathcal{H}_2^{sim}}$, and returns $h_{2,\alpha}$ as an answer to the $\mathcal{H}_2^{sim}$ query.

Note that $\mathcal{B}$ must compute α for at most all the users whose root branch index is μ. Since the depth of the hierarchy is logarithmic in k, this search is of polynomial complexity.

- Otherwise, $\mathcal{B}$ computes $h_{2,\alpha} = \hbar(\alpha)$, adds $(\alpha, h_{2,\alpha})$ to $\Lambda_{\mathcal{H}_2^{sim}}$, and returns $h_{2,\alpha}$ as an answer to the $\mathcal{H}_2^{sim}$ query.

Attack Game:

- By generating or reusing the appropriate entries and values of both $\Lambda_{\mathcal{H}_1^{sim}}$ and $\Lambda_{\mathcal{H}_2^{sim}}$, $\mathcal{B}$ answers the *Public Key* and the *Extraction* queries of *Phase 1*, under the following constraint: if the root branch index of the i^{th} ID-tuple of an *Extraction* query is μ, then $\mathcal{B}$ terminates the *IND-HIB-CPA* game and fails to solve the BDH problem.

- Once *Phase 1* is over, the *Challenge* phase starts. $\mathcal{A}$ issues a target ID-tuple $\overline{ID}_t^* = (ID_1, \cdots, ID_t)$ and a pair (m_0, m_1) of two equal-length plaintexts, such that, in *Phase 1*, no *Key Extraction* queries were issued on $\overline{ID}_t^*$ or any of its ancestors.

  If $\overline{ID}_t^*$'s root branch index is not μ, then $\mathcal{B}$ terminates the *IND-HIB-CPA* game and fails to solve the BDH problem. Otherwise, $\mathcal{B}$ picks a random bit $b \in_R \{0,1\}$ and sends to $\mathcal{A}$ the ciphertext $c^* = (U_1, U_2, V)$, where $U_1 = \pi(cP_0^{(1)}) = cP_0^{(4)}$, $U_2 = \omega(cP_0^{(1)}) = cP_0^{(3)}$, and $V \in_R \{0,1\}^n$.

  Note that $S_{\overline{ID}_t^*} = s_0 I_1 (\frac{A_t}{I_1} \prod_{i=2}^t s_{i-1}) P_0^{(1)} + (\prod_{i=2}^t s_{i-1}) B_t(s_0 P_0^{(2)}) +$ $(\prod_{i=2}^t s_{i-1}) F_t(s_0 P_0^{(3)})$, where $I_1 = b$, and both $\frac{A_t}{I_1}$ and B_t can be computed by $\mathcal{B}$ (since they are multivariate polynomials in I_i, I'_i, I''_i for $i = 1, \cdots, t$ – i.e. not in I_1 nor in I_1^{-1}).

  Note also that $\hat{e}(\pi cP_0^{(1)}, (\prod_{i=2}^t s_{i-1}) F_t(s_0 P_0^{(3)})) \cdot \hat{e}(\varsigma cP_0^{(1)}, (\prod_{i=2}^t s_{i-1}) C_t(s_0 P_0^{(3)})) \cdot$ $\hat{e}(\vartheta cP_0^{(1)}, (\prod_{i=2}^t s_{i-1}) D_t(s_0 P_0^{(3)})) = \hat{e}(U_2, \widetilde{S}_{\overline{ID}_t^*})$.

  Let $\Delta_1 = (\frac{A_t}{I_1} \prod_{i=2}^t s_{i-1})$, $\Delta_2 = (\prod_{i=2}^t s_{i-1}) B_t$, $\Delta_3 = (\prod_{i=2}^t s_{i-1}) F_t$, $\Delta_4 = (\prod_{i=2}^t s_{i-1}) C_t$, and $\Delta_5 = (\prod_{i=2}^t s_{i-1}) D_t$. Then, $\hat{e}(U_1, S_{\overline{ID}_t^*}) =$ $\hat{e}(\pi cP_0^{(1)}, s_0 I_1 P_0^{(1)})^{\Delta_1} \cdot \hat{e}(\pi cP_0^{(1)}, s_0 P_0^{(2)})^{\Delta_2} \cdot \hat{e}(\pi cP_0^{(1)}, s_0 P_0^{(3)})^{\Delta_3}$.

Hence, $\hat{e}(P_0^{(1)}, P_0^{(1)})^{abc} = \left((\hat{e}(U_1, S_{\overline{ID}_t^*}) \cdot \hat{e}(U_2, \widetilde{S}_{\overline{ID}_t^*})^{-1}) \cdot \Gamma^{-1}\right)^{(\pi\Delta_1)^{-1}}$, where $\Gamma = \hat{e}(\pi c P_0^{(1)}, s_0 P_0^{(2)})^{\Delta_2} \cdot \hat{e}(\varsigma c P_0^{(1)}, s_0 P_0^{(3)})^{-\Delta_4} \cdot \hat{e}(\vartheta c P_0^{(1)}, s_0 P_0^{(3)})^{-\Delta_5}$.

- According to the rules of the *IND-HIB-CPA* game, $\mathcal{A}$ then makes a second series of queries, which $\mathcal{B}$ answers as in *Phase 1*, under the following constraints: (1) the restrictions of the second phase of the game must be followed; (2) if $\mathcal{A}$ issues an *Extraction* query on an ID-tuple whose root branch index is μ, then $\mathcal{B}$ terminates the *IND-HIB-CPA* game and fails to solve the BDH problem.

- Once the second phase of the *IND-HIB-CPA* game is over, $\mathcal{A}$ makes a guess $\beta' \in \{0, 1\}$ and sends it to $\mathcal{B}$. $\mathcal{B}$ ignores β', uniformly picks a random entry $(\alpha, h_{2,\alpha})$ of $\Lambda_{\mathcal{H}_2^{sim}}$, and submits

$$\left(\alpha \cdot \Gamma^{-1}\right)^{(\pi\Delta_1)^{-1}}$$

as the solution of the BDH problem.

Note that $\mathcal{A}$ must compute $\hat{e}(U_1, S_{\overline{ID}_t^*})\hat{e}(U_2, \widetilde{S}_{\overline{ID}_t^*})^{-1}$ in order to conclude that c^* is an invalid ciphertext. When this happens, then it suffices that $\mathcal{B}$ picks the right entry of $\Lambda_{\mathcal{H}_2^{sim}}$, in order to solve the BDH. Moreover, if $\mathcal{A}$ does not find out that c^* is invalid, then $\mathcal{B}$ does not solve the BDH. Consequently, $\mathcal{B}$ solves the BDH if and only if the following conditions are satisfied: (1) In both *Phase 1* and *Phase 2*, no *Extraction* is issued on an ID-tuple whose root branch index is μ; (2) $\mathcal{A}$ asks to be challenged on an ID-tuple whose root branch index is μ; (3) $\mathcal{A}$ discovers that c^* is an invalid ciphertext; (4) $\mathcal{B}$ picks the right element of $\Lambda_{\mathcal{H}_2^{sim}}$.

Complexity and Probability:

In order to compute the probability that $\mathcal{B}$ solves the BDH problem, let δ_μ be the proportion of ID-tuples whose root branch index is μ (among all possible *ID*-tuples). Assume that, in the *IND-HIB-CPA* game, $\mathcal{A}$ issues a total of n_{ext} *Extraction* queries. If $\varepsilon(k)$ is the advantage of $\mathcal{A}$ in the *IND-HIB-CPA* game, then the probability that $\mathcal{B}$ solves the BDH problem is $\varepsilon(k)(1-\delta_\mu)^{n_{ext}}\delta_\mu$. Now, $\delta_\mu \leq \delta$, since δ is the largest proportion of ID-tuples having a common root branch. Moreover, since all proportions of ID-tuples having a common root branch are assumed to be non-negligible quantities, there exists a polynomial *pol* over $\mathbb{N}$ such that $\delta_\mu \geq \frac{1}{pol(k)}$. Hence, $\varepsilon(k)\delta_\mu(1-\delta_\mu)^{n_{ext}} \geq \varepsilon(k)\frac{1}{pol(k)}(1-\delta)^{n_{ext}}$. Now, since $\varepsilon(k)$ is non-negligible, so is $\varepsilon(k)\frac{1}{pol(k)}(1-\delta)^{n_{ext}}$. Consequently, $\mathcal{B}$ solves the BDH problem with non-negligible probability at least $\varepsilon(k)\frac{1}{pol(k)}(1-\delta)^{n_{ext}}$. Moreover, since each query of the *IND-HIB-CPA* game requires $\mathcal{B}$ to make a polynomial number of operations and trials, it follows

that $\mathcal{B}$ solves the BDH problem in time $O(\tau)$ where τ is the running time of $\mathcal{A}$ in the *IND-HIB-CPA* attack game. **QED**.

4.7 Conclusion

The aim of this chapter was threefold: first, we sought to describe a HIBKG algorithm featuring constant-time key generation and constant-size keys; second, we wanted to design a HIBE scheme with constant ciphertext length and fast decryption time; third, we aimed at devising a HIBS scheme with constant-size signatures and fast signature verification.

This was achieved by a proposed set of schemes which improve very recent independent work in hierarchical ID-based cryptography. The computational and storage requirements of the suggested schemes were examined and compared with existing schemes. The cost of our HIBS scheme's signing algorithm, and the cost of our HIBE scheme's decryption procedure are comparable with the costs of the best known analogue schemes. Moreover, our HIBS and HIBE schemes respectively achieve fast signature verification and fast encryption for deep hierarchies ($t \leq 12$). Furthermore, our HIBKG algorithm is the most efficient one (among known HIBKG algorithms). The security of our HIBE scheme was also studied, showing that it achieves semantic security with respect to chosen ciphertext attacks, in the random oracle model, under a standard number theoretic assumption.

In the next chapter, we investigate selected applications of our HIBE and HIBS schemes, while *the second part* of this dissertation's second main contribution (i.e. the design of hierarchical ID-based signcryption scheme with PCA and FS) is described in Chapter 6.

Chapter 5

Selected HIBE and HIBS Applications

The aim of this chapter is to describe selected cryptographic schemes which can be derived from the HIBE and HIBS schemes proposed in Chapter 4.

Our selection criteria were: (1) practical relevance of the applications supported by the selected schemes; and (2) computational or space efficiency of the selected schemes (in comparison with similar schemes derived from other known hierarchical ID-based algorithms). We selected one *forward-secure hierarchical ID-based encryption* (fsHIBE) scheme, and one *append-only signature* (AOS) scheme.

§5.1 presents such an fsHIBE scheme, and §5.2 describes an AOS scheme.

5.1 Forward-Secure Encryption

The HIBE scheme proposed in Chapter 4 (i.e. NewHIBE) yields various efficient forward-secure public-key encryption (fsPKE) schemes. This section presents an overview of these fsPKE schemes, and then describes our proposed fsHIBE scheme.

5.1.1 Overview of fsPKE schemes

Recall that fsPKE assumes that time is divided into time periods, and that users are given time-bound keys which must be evolved. The motivation for designing fsPKE schemes is to limit the impact of key compromise, by describing a method whereby the compromise of the decryption key associated with a user $\mathcal{U}$ and a time period t_p only affects the confidentiality of ciphertexts encrypted for $\mathcal{U}$ at a time period $\tau \geq t_p$.

fsPKE

Canetti et al. [CHK03] proposed a method to construct a fs-PKE scheme from any HIBE scheme. When Canetti et al.'s scheme [CHK03] is instantiated with BBG-HIBE, the length of time-based keys is $O(\log^2(T))$-long where T is the number of time periods whose information is accessible. However, when NewHIBE is used instead of BBG-HIBE, the length of keys is reduced to $O(\log(T))$.

fsHIBE

NewHIBE can also be used to instantiate Yao et al.'s fsHIBE scheme [YFDL04], and the same is true for BBG-HIBE (i.e. Boneh et al.'s HIBE scheme with constant-size ciphertexts). Let us call NAM-fsHIBE our instantiation of Yao et al.'s fsHIBE scheme, and BBG-fsHIBE a similar instantiation based on BBG-HIBE.

Note that Boneh and al. [BBG05] did not describe BBG-fsHIBE (i.e. an fsHIBE scheme with constant-size ciphertexts). (Instead, these authors devised an fsHIBE scheme with growing ciphertext length, but shorter keys than those which BBG-fsHIBE would have.) NAM-fsHIBE will be described in §5.1.2. However, we already underscore two benefits which our fsHIBE has over BBG-fsHIBE.

Suppose that users of a system are modelled by nodes of a tree-shaped hierarchy whose maximal depth is denoted by ℓ. Let t be the hierarchical depth of an arbitrary user, and T be the number of time periods of a fsHIBE system. Then, compared with BBG-fsHIBE, the NAM-fsHIBE saves $O(\ell \log(T))$ multiplications in $\mathbb{G}_1$ for the generation of each child-key, and $O(\ell \cdot t)$ multiplications in $\mathbb{G}_1$ for each key evolution.

Consider, for instance, a community whose users are modelled as leaves of a tree-shaped hierarchy of depth 4. Suppose that time periods of one hour are considered, over a period of 2 years. Then, compared with a BBG-HIBE instantiation of Yao et al.'s fsHIBE scheme, the proposed fsHIBE scheme saves about 50 multiplications in $\mathbb{G}_1$ for each child-key generation, and about 15 multiplications in $\mathbb{G}_1$ for each key evolution executed by a level-4 user.

Reverse fsHIBE

It is interesting to note that fsHIBE schemes can be used in reverse order, in order to support time-based release of (confidential) information (TIR).

Mont et al. [MHS03] described a mechanism which exploits IBE to encrypt information using date strings, in such a way that the information can be decrypted only when the corresponding date has arrived. To achieve this, a cryptographic server (i.e. a PKG) is trusted to release keys of a given date only when the time has arrived. Using, in reverse order, any fsPKE scheme, it is possible to provide users

with one key which allows them to decrypt all documents released before a given date. Thus, using only one key per time period, one can decrypt all documents released before this given period. Schemes which support time-based information release are called TIR schemes. Moreover, TIR schemes which release keys associated with hierarchically-structured classes of users are called HTIR schemes. For instance, these hierarchically-structured classes can model roles in a RBAC system. Hence, both BBG-fsHIBE and NAM-fsHIBE can be used to construct HTIR schemes (henceforth called BBG-HTIR and NAM-HTIR respectively).

	NAM-HTIR	**BBG-HTIR** with O(1)-long ciphertexts [BBG05]
Root Setup	$O(\ell \cdot \log(\mathcal{T}))(M_{\mathbb{G}_1})$	$O(1)M_{\mathbb{G}_1}$
Lower-Level Setup	$O(\log^2(\mathcal{T}))(A_{\mathbb{G}_1} + M_{\mathbb{G}_1} + M_{\mathbb{Z}_q^*})$	$O(\ell \cdot \log^2(\mathcal{T}))(A_{\mathbb{G}_1} + M_{\mathbb{G}_1})$
Encryption	$O(t \cdot \log(\mathcal{T}))M_{\mathbb{Z}_q^*}$ $+ O(1)(M_{\mathbb{G}_1} + Ex_{\mathbb{G}_2} + \mathbf{P})$	$O(t \cdot \log(\mathcal{T}))(A_{\mathbb{G}_1} + M_{\mathbb{G}_1})$ $+ O(1)\mathbf{P}$
Evolution	$O(t)(A_{\mathbb{G}_1} + M_{\mathbb{G}_1} + M_{\mathbb{Z}_q^*})$	$O(\ell \cdot t)(A_{\mathbb{G}_1} + M_{\mathbb{G}_1})$
M-Evolution	$O(\log(\mathcal{T}) \cdot t)(A_{\mathbb{G}_1} + M_{\mathbb{G}_1} + M_{\mathbb{Z}_q^*})$	$O(\log(\mathcal{T}) \cdot \ell \cdot t)(A_{\mathbb{G}_1} + M_{\mathbb{G}_1})$
Decryption	$O(1)(Ex_{\mathbb{G}_2} + Inv_{\mathbb{G}_2} + M_{\mathbb{G}_1}$ $+ M_{\mathbb{G}_2} + M_{\mathbb{Z}_q^*} + \mathbf{P})$	$O(1)(Inv_{\mathbb{G}_2} + M_{\mathbb{G}_2} + \mathbf{P})$
Key Stack Length	$O(\log(\mathcal{T}))$	$O((\ell - t) \cdot \log(\mathcal{T}))$

Table 5.1: Comparison of NAM-HTIR with BBG-HTIR.

Table 5.1 compares the computational requirements of NAM-HTIR with BBG-HTIR. t denotes both the hierarchical level of a PKG (user) and the hierarchical level of a decryptor (or ciphertext's intended recipient.) $\mathcal{T}$ denotes the number of time periods handled by the HTIR schemes. ℓ denotes the depth of the user hierarchy. M_X and A_X respectively denote the computational costs of scalar multiplication and addition in the Abelian group X. The computational cost of exponentiation in the group X is denoted by Ex_X, $\mathbf{P}$ denotes the computational cost of a bilinear pairing operation, and Inv_X denotes the computational cost of inversion in X. The computational cost of hashing is not considered, due to its insignificance in comparison with the computational cost of pairing and that of operations in $\mathbb{G}_1$ and $\mathbb{G}_2$. M-Evolution refers the process whereby, at time period i, a user runs the evolution algorithm multiple (say j) times, in order to generate the key required to access a documents associated with time period $i - j$. It should be noted that the storage requirements of each algorithm essentially follows the order of its computational complexity (with the exception of BBG-HTIR's *Root Setup* algorithm, which requires $O(\ell \cdot \log(\mathcal{T}))$ storage

units[1].)

The following methodology was used to compute the requirements of BBG-HTIR. Since BBG-HIBE requires 2 scalar multiplications in $\mathbb{G}_1$ and $O(\ell)$ hash function evaluations, it is estimated that BBG-HTIR would require $O(1)$ scalar multiplications in $\mathbb{G}_1$. (Recall that the cost of hash function evaluation is not taken into account in Table 5.1.) For the *Lower-Level Setup* algorithm, we evaluate the cost of applying $O(\mathcal{T})$ times the *Key Extraction* algorithm of BBG-HIBE (i.e. one key-extraction per time identifier component), for each element of a key stack. Since the cost of each key extraction of BBG-HIBE is $O(\ell)(M_{\mathbb{G}_1} + A_{\mathbb{G}_1})$, and since there are, on average, $O(\mathcal{T})$ keys in each key stack, the computational cost of the *Lower-Level Setup* procedure is estimated to be $O(\ell \cdot \log^2(\mathcal{T}))(A_{\mathbb{G}_1} + M_{\mathbb{G}_1})$. For the cost of BBG-HTIR's *Encryption*, it is assumed (according to Yao et al.'s scheme) that encryption requires $O(\ell) \cdot t$ steps (i.e. one step for each time identifier component and, for each time identifier component, one step for each user identifier component). Since the cost of each step of BBG-HIBE's encryption routine is $O(1)(M_{\mathbb{G}_1} + A_{\mathbb{G}_1})$, we obtain the result shown in Table 5.1 (recall that BBG-HIBE's encryption algorithm requires one pairing only). *Key Evolution* requires at most two key extractions of BBG-HIBE, for each component of the associated user identifier; hence, the cost *Key Extraction* is $O(\ell \cdot t)(A_{\mathbb{G}_1} + M_{\mathbb{G}_1})$. Thus, by applying *Evolution* multiple times (i.e. $O(\mathcal{T})$ times since the length of time identifiers is $O(\mathcal{T})$), we obtain the cost of *M-Evolution* displayed in Table 5.1. Finally, the cost of BBG-HTIR's *Decryption* algorithm follows directly from that of BBG-HIBE decryption routine (i.e. $O(1)(Inv_{\mathbb{G}_2} + M_{\mathbb{G}_2} + P)$.)

Table 5.1 shows that the computational cost of our proposed HTIR scheme's *Setup* algorithm is greater than BBG-HTIR's. This is a one-time cost, which must be paid at the beginning of the system. Table 5.1 also shows that BBG-HTIR's *Lower-Level Setup* and *Evolution* algorithm take $O(\ell)$ more steps than their analogues in NAM-HTIR. This factor is increased by a factor of $O(\log(\mathcal{T}))$ in the case of M-Evolution. Thus, if users of a HTIR system are modelled as leaves of a 4-level hierarchy, and if time periods of one hour are considered for a total time of 2 years, then M-Evolution (i.e. one of the most frequent operations of the HTIR schemes) is, on average, about 30 times slower with BBG-TIR than with our scheme (for users lying at the bottom of the hierarchy.)

Thus, we have seen in this section that NewHIBE can be used to instantiate various fsPKE schemes, including a fsHIBE and a reverse fsHIBE scheme. We have also established the fact that these derived schemes feature several computational benefits over analogous schemes obtained from BBG-HIBE. This being done, we now proceed to the formal description of NAM-fsHIBE.

[1]The term *storage unit* refers to the amount of space required to store an element of $\mathbb{Z}_q^*$. The corresponding numeric value (e.g. 512 bits [BF01]) may vary from one platform to the other.

5.1.2 Proposed fsHIBE Scheme

Fundamental notational definitions concerning fsHIBE schemes are first presented, followed by the actual fsHIBE scheme.

Fundamental Notational Definitions

ID-tuples. For the description of fsHIBE schemes, PKGs are assumed to be organized in a tree-shaped hierarchy whose root is called the root PKG and is denoted by both $rPKG$ and $\overline{ID}_0$. Apart from $rPKG$, every PKG is identified with a tuple $\overline{ID}_t = (ID_1, ID_2, \cdots, ID_t)$ (called *ID-tuple*). Under such a notation, $\overline{ID}_t$'s ancestors consists of $rMan$ and the $\overline{ID}_i$'s such that $1 \leq i < t$.

Time Identifiers. For the definition of fsHIBE schemes, time is assumed to be sliced in a sequence of time periods, each of which is labelled with a non-negative integer. Each time period $i \in \mathbb{N}$ is associated with a unique node $\underline{i}$, in a binary-tree shaped hierarchy denoted by T. T's root is also denoted by ϵ and corresponds to time 0 – the beginning of time for the associated forward secure system. Since each node $\underline{i}$ of the binary tree T corresponds to a time period i, each time period of the system has a binary representation which corresponds to the path (in T) starting at ϵ and ending at $\underline{i}$. We denote by $[i] = i_1 \cdots i_\theta$ this binary representation of i, and by $i|_j$ the integer whose binary representation is $i_1 \cdots i_j$. (By definition, $i|_0$ denotes 0.) The binary representation of a time period is computed as follows: if $\underline{i}$ is an internal node of T, then $\underline{i+1}$ is the left child of $\underline{i}$; otherwise (i.e. if $\underline{i}$ is a leaf node of T), $\underline{i+1}$ is the right child of the node $\underline{j}$, where $[j]$ is the longest string such that $[j]0$ is a substring of $[i]$. Moreover, the size θ of $[i] = i_1 \cdots i_\theta$ is denoted by $|[i]|$.

Private Key Stacks. The definition of fsHIBE schemes is predicated on the assumption that each PKG (user) holds a *private key stack*. Such a stack is associated with both a PKG and a time period i. $\Sigma_{(\overline{ID}_t,i)}$ denotes the private key stack associated with a PKG $\overline{ID}_t$ and time period i. $\Sigma_{(\overline{ID}_t,i)}$ only contains all the information required to perform the following tasks: (1) to decrypt ciphertexts associated with $\overline{ID}_t$ and i; (2) to construct $\Sigma_{(\overline{ID}_t,j)}$, where $j > i$. To meet these requirements, $\Sigma_{(\overline{ID}_t,i)}$ is structured as follows: $\Sigma_{(\overline{ID}_t,i)} = ((d_{(\overline{ID}_t,j)})_{j\in\Psi}, d_{(\overline{ID}_t,i)})$, where $d_{(\overline{ID}_t,i)}$ is the decryption key associated with $\overline{ID}_t$ and i, and where Ψ is the identifier (i.e. label) sequence of all the right siblings of the nodes lying on the path from ϵ to $\underline{i}$ (in T, if these nodes exist). In other words, $\Sigma_{(\overline{ID}_t,i)}$ contains $d_{(\overline{ID}_t,i)}$ and the private keys associated with $\overline{ID}_t$ and all the right siblings of the nodes lying on the path going from ϵ to $\underline{i}$. $d_{(\overline{ID}_t,i)}$ is the top element of the stack.

Forward-Secure HIBE Scheme. Each forward-secure hierarchical ID-based encryption (fsHIBE) scheme is composed of five randomized algorithms, whose functions are described below:

1. `Root Setup` (k): Given a security parameter k, this algorithm is used by the root PKG to return a tuple *params* of system parameters. *params* includes a user hierarchy depth $\ell \in O(k)$, a number $\mathcal{T} \in O(k)$ of time periods, and a description of both the message space $\mathcal{M}$ and the ciphertext space $\mathcal{C}$, along with a secret piece of data $d_{\overline{ID}_0}$ called the root PKG's private key. Other parameters are allowed, *provided they are not unique to any user.* Some parameters may be public (including those describing $\mathcal{M}$ and $\mathcal{C}$), while others remain secret (including the root secret key).

2. `Lower-Level Setup` $(ID_{t+1}, \Sigma_{(\overline{ID}_t,i)})$: Given the scheme's public parameters, an arbitrary identifier ID_{t+1}, and the private key stack $\Sigma_{(\overline{ID}_t,i)}$ associated with both a PKG $\overline{ID}_t$ and a time period i, this algorithm returns the private key stack $\Sigma_{(\overline{ID}_{t+1},i)}$ associated with $\overline{ID}_{t+1}$ and i.

3. `Key Update` $(\Sigma_{(\overline{ID}_t,i)})$: Given the scheme's public parameters and the private key stack $\Sigma_{(\overline{ID}_t,i)}$ associated with a PKG $\overline{ID}_t$ and a time period i, this algorithm returns $\Sigma_{(\overline{ID}_t,i+1)}$, i.e. $\overline{ID}_t$'s private key stack for time $i+1$.

4. `Encrypt` $(m, \overline{ID}_t, i)$: Given the scheme's public parameters, a message $m \in \mathcal{M}$, the ID-tuple $\overline{ID}_t$ of an intended recipient, and a time period i, this algorithm returns a ciphertext $c \in \mathcal{C}$ associated with m and i.

5. `Decrypt` $(c, i, \Sigma_{(\overline{ID}_t,i)})$: Given the scheme's public parameters, a ciphertext $c \in \mathcal{C}$ (issued for time i), and the private key stack $\Sigma_{(\overline{ID}_t,i)}$ associated with $\overline{ID}_t$ and i, this algorithm returns the message $m \in \mathcal{M}$ associated with c and i.

Proposed Scheme

- `Instance Generator` (k). This procedure, denoted by IG, is a randomized algorithm which takes a security parameter $k > 0$, runs in $O(k)$, and outputs $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are two Abelian Gap-Diffie-Hellman groups of prime order $q \geq 2^k$, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is an admissible pairing with respect to which the BDH problem is intractable.

- `Root Setup` (k). Given a security parameter $k > 0$, the root *PKG*:

  1. runs IG with input k and obtains $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$;

2. picks, randomly and uniformly[2], $P_0^{(1)}, P_0^{(2)}, P_0^{(3)}, P_0^{(4)}, P_0^{(5)}, P_0^{(6)} \in \mathbb{G}_1$;
3. picks $s_0 \in_R \mathbb{Z}_q^*$, and sets $d_{\overline{ID}_0} = (s_0)$;
4. computes $n = poly_1(k)$, $n_\tau = poly_2(k)$, and $\ell = poly_3(k)$, $n_\Omega = poly_4(k)$, where $poly_i$ is a polynomial over the positive integers, for $i = 1, 2, 3$;
5. chooses cryptographic hash functions:
   $\mathcal{H}_1 : \{0,1\}^* \to (\mathbb{Z}_q^*)^{3\ell}$, $\mathcal{H}_2 : \{0,1\}^* \to (\mathbb{Z}_q^*)^{3n_\tau}$, $\mathcal{H}_5 : \mathbb{Z}_q^* \to \mathbb{Z}_q^*$,
   where $\mathcal{H}_1$ and $\mathcal{H}_2$ are defined using the same methodology, and $\mathcal{H}_1$ is defined as in NewHIBE-v1.
6. computes, $(s_i = \mathcal{H}_5(s_{i-1}))_{i=1}^{\ell \cdot n_\Omega - 1}$, $L_1^{(j)} = s_0 P_0{}^{(j)}$ for $j = 3, 4$, and $(L_i^{(j)} = s_{i-1} L_{i-1}^{(j)})_{i=2}^{\ell \cdot n_\Omega}$ for $j = 3, 4$.

The message space is $\mathcal{M} = \{0,1\}^n$ and the signature space is $\mathcal{C} = \mathbb{G}_1^2 \times \{0,1\}^n$. The system's public parameters (which must be certified) are $pubParams = (q, n, \hat{e}, I_0, P_0^{(1)}, P_0^{(2)}, P_0^{(3)}, P_0^{(4)}, P_0^{(5)}, P_0^{(6)}, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_5, ((L_i^{(j)})_{i=1}^{\ell \cdot n_\Omega})_{j=3}^{4})$, and the *root PKG* keeps s_0 secret, so that $params = (pubParams, s_0)$. Note that $params$ implicitly include ℓ and $\mathcal{T} = 2^{n_\tau}$ via the definition of $\mathcal{H}_1$ and $\mathcal{H}_2$.

- `Lower-Level Setup` $(ID_{t+1}, \Sigma_{(\overline{ID}_t, i)})$: For each PKG $\overline{ID}_{t+1}$ which becomes child of a PKG $\overline{ID}_t$, at time i, the following takes place:
  1. Let $[i] = i_1 \cdots i_\theta$. $\overline{ID}_t$ picks $\alpha_{(\overline{ID}_{t+1}, i|_j)}, \widetilde{\alpha}_{(\overline{ID}_{t+1}, i|_j)} \in_R \mathbb{Z}_q^*$, for $j = 0, \cdots, \theta$.
  2. $\overline{ID}_t$ computes $(\hat{\mathrm{J}}_1, \cdots, \hat{\mathrm{J}}_\theta, \hat{\mathrm{J}}_0, \cdots, \hat{\mathrm{J}}_0) = \mathcal{H}_2([i])$.
  3. – If $t = 0$, then $rPKG$ computes $p_{(1,2)} = \mathcal{H}_7(p_{(1,3)})$, $p_{(1,4)} = \frac{p_{(1,2)} p_{(1,3)}}{p_{(1,1)}}$, $I_1' = -p_{(1,2)}$, $I_1'' = -\frac{p_{(1,1)}}{p_{(1,2)}} I_1'$, $\widetilde{I}_1 = I_1 + p_{(1,1)}$, $\widetilde{I}_1' = I_1' + p_{(1,2)}$, $\widetilde{I}_1'' = I_1'' + p_{(1,4)}$, $R_{(\overline{ID}_1, 0)} = p_{(1,1)} \alpha_{(\overline{ID}_1, 0)} + p_{(1,2)} \widetilde{\alpha}_{(\overline{ID}_1, 0)}$, $\widetilde{R}_{(\overline{ID}_1, 0)} = p_{(1,3)} \alpha_{(\overline{ID}_1, 0)} + p_{(1,4)} \widetilde{\alpha}_{(\overline{ID}_1, 0)}$, $s_1 = \mathcal{H}_5(s_0)$, and the following:

$$
\begin{aligned}
S_{(\overline{ID}_1, 0)} &= s_0 (I_1 P_0^{(1)} + I_1' P_0^{(2)} + \alpha_{(\overline{ID}_1, 0)} P_0^{(3)}), \\
E_{(\overline{ID}_1, 0)} &= s_0 (\tfrac{I_1 I_1''}{I_1'} P_0^{(1)} + I_1'' P_0^{(2)} + \widetilde{\alpha}_{(\overline{ID}_1, 0)} P_0^{(3)}), \\
\widetilde{S}_{(\overline{ID}_1, 0)} &= s_0 (\alpha_{(\overline{ID}_1, 0)} P_0^{(4)} + \widetilde{I}_1 P_0^{(5)} + \widetilde{I}_1' P_0^{(6)}), \\
\widetilde{E}_{(\overline{ID}_1, 0)} &= s_0 (\widetilde{\alpha}_{(\overline{ID}_1, 0)} P_0^{(4)} + \tfrac{\widetilde{I}_1 \widetilde{I}_1'}{\widetilde{I}_1'} P_0^{(5)} + \widetilde{I}_1'' P_0^{(6)}), \\
d_{(\overline{ID}_1, 0)} &= (S_{(\overline{ID}_1, 0)}, E_{(\overline{ID}_1, 0)}, \widetilde{S}_{(\overline{ID}_1, 0)}, \widetilde{E}_{(\overline{ID}_1, 0)}, R_{(\overline{ID}_1, 0)}, \widetilde{R}_{(\overline{ID}_1, 0)}, p_{(1,2)}, s_1), \Sigma_{(\overline{ID}_1, 0)}).
\end{aligned}
$$

  – If $t = 0$ and $i \geq 1$, then $rPKG$ recursively applies the *Key Update* algorithm i times, starting with $\Sigma_{(\overline{ID}_1, 0)}$.

[2] In the sequel, we shall use the notation $x \in_R X$ to indicate that the element x is chosen uniformly at random from the set X.

- If $t \geq 1$, $\overline{ID}_t$ computes: $I'_{t+1} = -p_{(t,2)}$, $I''_{t+1} = -\frac{p_{(t,2)}p_{(t,3)}}{p_{(t,1)}}$, $\widetilde{I}_{t+1} = I_{t+1} + p_{(t,1)}$, $\widetilde{I}'_{t+1} = I'_{t+1} + p_{(t,2)}$, $\widetilde{I}''_{t+1} = I''_{t+1} + p_{(t,4)}$; $p_{(t+1,2)} = -\frac{p_{(t,1)}}{p_{(t,3)}}p_{(t+1,1)}$, $p_{(t+1,4)} = \frac{p_{(t+1,2)}p_{(t+1,3)}}{p_{(t+1,1)}}$; $R_{(\overline{ID}_{t+1},i|_0)} = p_{(t+1,1)}\alpha_{(\overline{ID}_{t+1},i|_0)} + p_{(t+1,2)}\widetilde{\alpha}_{(\overline{ID}_{t+1},i|_0)}$, $\widetilde{R}_{(\overline{ID}_{t+1},i|_0)} = p_{(t+1,3)}\alpha_{(\overline{ID}_{t+1},i|_0)} + p_{(t+1,4)}\widetilde{\alpha}_{(\overline{ID}_{t+1},i|_0)}$; and $s_{t\cdot(\theta+1)+1} = \mathcal{H}_5(s_{t\cdot(\theta+1)})$. Then, for each private key $d_{(\overline{ID}_t,j)}$ of $\Sigma_{(\overline{ID}_t,i)}$ ($j \in \Psi \bigcup \{i\}$), $\overline{ID}_t$ computes the following:

(a)
$$
\begin{aligned}
S_{(\overline{ID}_{t+1},0,j)} &= s_{t\cdot(\theta+1)}(I_{t+1}S_{(\overline{ID}_t,j)} + I'_{t+1}E_{(\overline{ID}_t,i)}) + \alpha_{(\overline{ID}_{t+1},i|_0)}L^{(3)}_{t\cdot(\theta+1)+1},\\
E_{(\overline{ID}_{t+1},0,j)} &= s_{t\cdot(\theta+1)}(\tfrac{I_{t+1}I''_{t+1}}{I'_{t+1}}S_{(\overline{ID}_t,j)} + I''_{t+1}E_{(\overline{ID}_t,j)}) + \widetilde{\alpha}_{(\overline{ID}_{t+1},i|_0)}L^{(3)}_{t\cdot(\theta+1)+1},\\
\widetilde{S}_{(\overline{ID}_{t+1},0,j)} &= s_{t\cdot(\theta+1)}(\widetilde{I}_{t+1}\widetilde{S}_{(\overline{ID}_t,j)} + \widetilde{I}'_{t+1}\widetilde{E}_{(\overline{ID}_t,j)})\\
&\quad + (\alpha_{(\overline{ID}_{t+1},i|_0)} - R_{(\overline{ID}_t,i|_0)})L^{(4)}_{t\cdot(\theta+1)+1},\\
\widetilde{E}_{(\overline{ID}_{t+1},0,j)} &= s_{t\cdot(\theta+1)}(\tfrac{\widetilde{I}_{t+1}\widetilde{I}''_{t+1}}{\widetilde{I}'_{t+1}}\widetilde{S}_{(\overline{ID}_t,j)} + \widetilde{I}''_{t+1}\widetilde{E}_{(\overline{ID}_t,j)})\\
&\quad + (\widetilde{\alpha}_{(\overline{ID}_{t+1},i|_0)} - \widetilde{R}_{(\overline{ID}_t,i|_0)})L^{(3)}_{t\cdot(\theta+1)+1},
\end{aligned}
$$

(b) Then, for $0 \leq a < \theta$, $\overline{ID}_t$ computes: $\hat{I}'_{a+1} = -p_{(a,2)}$, $\hat{I}''_{a+1} = -\frac{p_{(a,2)}p_{(a,3)}}{p_{(a,1)}}$, $\widetilde{\hat{I}}_{a+1} = \hat{I}_{a+1} + p_{(a,1)}$, $\widetilde{\hat{I}}'_{a+1} = \hat{I}'_{a+1} + p_{(a,2)}$, $\widetilde{\hat{I}}''_{a+1} = \hat{I}''_{a+1} + p_{(a,4)}$; $p_{(a+1,2)} = -\frac{p_{(a,1)}}{p_{(a,3)}}p_{(a+1,1)}$, $p_{(a+1,4)} = \frac{p_{(a+1,2)}p_{(a+1,3)}}{p_{(a+1,1)}}$;
$R_{(\overline{ID}_{t+1},i|_{a+1})} = p_{(a+1,1)}\alpha_{(\overline{ID}_{t+1},i|_{a+1})} + p_{(a+1,2)}\widetilde{\alpha}_{(\overline{ID}_{t+1},i|_{a+1})}$,
$\widetilde{R}_{(\overline{ID}_{t+1},i|_{a+1})} = p_{(a+1,3)}\alpha_{(\overline{ID}_{t+1},i|_{a+1})} + p_{(a+1,4)}\widetilde{\alpha}_{(\overline{ID}_{t+1},i|_{a+1})}$;
$s_{t\cdot(\theta+1)+2+a} = \mathcal{H}_5(s_{t\cdot(\theta+1)+1+a})$, and the following:

$$
\begin{aligned}
S_{(\overline{ID}_{t+1},i|_{a+1},j)} &= s_{t\cdot(\theta+1)+1+a}(\hat{I}_{a+1}S_{(\overline{ID}_{t+1},i|_a,j)} + \hat{I}'_{a+1}E_{(\overline{ID}_{t+1},i|_a,j)})\\
&\quad + \alpha_{(\overline{ID}_{t+1},i|_{a+1})}L^{(3)}_{t\cdot(\theta+1)+2+a},\\
E_{(\overline{ID}_{t+1},i|_{a+1},j)} &= s_{t\cdot(\theta+1)+1+a}(\tfrac{\hat{I}_{a+1}\hat{I}''_{a+1}}{\hat{I}'_{a+1}}S_{(\overline{ID}_{t+1},i|_a,j)} + \hat{I}''_{a+1}E_{(\overline{ID}_{t+1},i|_a,j)})\\
&\quad + \widetilde{\alpha}_{(\overline{ID}_{t+1},i|_{a+1})}L^{(3)}_{t\cdot(\theta+1)+2+a},\\
\widetilde{S}_{(\overline{ID}_{t+1},i|_{a+1},j)} &= s_{t\cdot(\theta+1)+1+a}(\widetilde{\hat{I}}_{a+1}\widetilde{S}_{(\overline{ID}_{t+1},i|_a,j)} + \widetilde{\hat{I}}'_{a+1}\widetilde{E}_{(\overline{ID}_{t+1},i|_a,j)})\\
&\quad + (\alpha_{(\overline{ID}_{t+1},i|_{a+1})} - R_{(\overline{ID}_{t+1},i|_a)})L^{(3)}_{t\cdot(\theta+1)+2+a},\\
\widetilde{E}_{(\overline{ID}_{t+1},i|_{a+1},j)} &= s_{t\cdot(\theta+1)+1+a}(\tfrac{\widetilde{\hat{I}}_{a+1}\widetilde{\hat{I}}''_{a+1}}{\widetilde{\hat{I}}'_{a+1}}\widetilde{S}_{(\overline{ID}_{t+1},i|_a,j)} + \widetilde{\hat{I}}''_{a+1}\widetilde{E}_{(\overline{ID}_{t+1},i|_a,j)})\\
&\quad + (\widetilde{\alpha}_{(\overline{ID}_{t+1},i|_{a+1})} - \widetilde{R}_{(\overline{ID}_{t+1},i|_a)})L^{(3)}_{t\cdot(\theta+1)+2+a},
\end{aligned}
$$

(c) $d_{(\overline{ID}_{t+1},i)} = (S_{(\overline{ID}_{t+1},i,j)}, E_{(\overline{ID}_{t+1},i|_{a+1},j)}, \widetilde{S}_{(\overline{ID}_{t+1},i|_{a+1},j)}, \widetilde{E}_{(\overline{ID}_{t+1},i|_{a+1},j)}, R_{(\overline{ID}_{t+1},i|_{a+1})}, \widetilde{R}_{(\overline{ID}_{t+1},i|_{a+1})}, p_{(t+1,2)}, s_{(t+1)\cdot(\theta+1)})$.
$\Sigma_{(\overline{ID}_{t+1},i)} = ((d_{(\overline{ID}_{t+1},j)})_{j\in\Psi}, d_{(\overline{ID}_{t+1},i)})$, where
$\Sigma_{(\overline{ID}_t,i)} = ((d_{(\overline{ID}_t,j)})_{j\in\Psi}, d_{(\overline{ID}_t,i)})$.

4. Finally, $\overline{ID}_t$ secretly gives $\Sigma_{(\overline{ID}_{t+1},i)}$ to $\overline{ID}_{t+1}$.

- `Key Update` $(\Sigma_{(\overline{ID}_t,i)})$: Given a private key stack $\Sigma_{(\overline{ID}_t,i)}$ associated with a PKG $\overline{ID}_t$ and a time period i, the following takes place:

  - Recall that $d_{(\overline{ID}_t,i)}$ is the top element of $\Sigma_{(\overline{ID}_t,i)}$. Let $\Sigma_{(\overline{ID}_t,i)} = ((d_{(\overline{ID}_t,j)})_{j\in\Psi}, d_{(\overline{ID}_t,i)})$.
    If $\underline{i}$ is a leaf node, then: (1) $d_{(\overline{ID}_t,i)}$ is popped off the stack; (2) $d_{(\overline{ID}_t,a)}$ (where a is the last identifier of Ψ) is moved to first element of $\Sigma_{(\overline{ID}_t,i)}$ in order to replace $d_{(\overline{ID}_t,i)}$ (thereby making Ψ one element shorter); (3) $\Sigma_{(\overline{ID}_t,i+1)}$ is the resulting stack.
    Otherwise (i.e. if $\underline{i}$ is an internal node), the following takes place: (1) $d_{(\overline{ID}_t,i)}$ is popped off the stack; (2) $\overline{ID}_t$ computes $d_{(\overline{ID}_t,i0)}$ and $d_{(\overline{ID}_t,i1)}$ (using the *ComputeNext* procedure presented below); (3) $\overline{ID}_t$ pushes first $d_{(\overline{ID}_t,i0)}$ and then $d_{(\overline{ID}_t,i1)}$ onto the stack (thereby making $i1$ the last identifier of Ψ and replacing[3] $d_{(\overline{ID}_t,i)}$ with $d_{(\overline{ID}_t,i0)}$); (4) $\Sigma_{(\overline{ID}_t,i+1)}$ is the resulting stack.
  - *ComputeNext* $(d_{(\overline{ID}_t,i)})$:
    1. Let $[i] = i_1 \cdots i_\theta$. Pick $\alpha_{(\overline{ID}_t,i0)}, \widetilde{\alpha}_{(\overline{ID}_t,i0)} \in_R \mathbb{Z}_q^*$.
    2. Compute $(J_1, \cdots, J_t, J_0, \cdots, J_0) = \mathcal{H}_1(\overline{ID}_t)$ and $(\hat{J}_1, \cdots, \hat{J}_\theta, \hat{J}_{(\theta+1,0)}, \hat{J}_0, \cdots, \hat{J}_0) = \mathcal{H}_2([i0])$, $(\hat{J}_1, \cdots, \hat{J}_\theta, \hat{J}_{(\theta+1,1)}, \hat{J}_0, \cdots, \hat{J}_0) = \mathcal{H}_2([i1])$, $\hat{I}'_{(\theta+1,i0)} = -p_{((\theta,i0),2)}$, $\hat{I}''_{(\theta+1,i0)} = -\frac{p_{((\theta,i0),2)}p_{((\theta,i0),3)}}{p_{((\theta,i0),1)}}$, $\widetilde{\hat{I}}_{(\theta+1,i0)} = \hat{I}_{(\theta+1,i0)} + p_{((\theta,i0),1)}$, $\widetilde{\hat{I}}'_{(\theta+1,i0)} = \hat{I}'_{(\theta+1,i0)} + p_{((\theta,i0),2)}$, $\widetilde{\hat{I}}''_{(\theta+1,i0)} = \hat{I}''_{(\theta+1,i0)} + p_{((\theta,i0),4)}$; $p_{((\theta+1,i0),2)} = -\frac{p_{((\theta,i0),1)}}{p_{((\theta,i0),3)}} p_{((\theta+1,i0),1)}$, $p_{((\theta+1,i0),4)} = \frac{p_{((\theta+1,i0),2)}p_{((\theta+1,i0),3)}}{p_{((\theta+1,i0),1)}}$;
    $R_{(\overline{ID}_{t+1},i|_{(\theta+1,i0)})} = p_{((\theta+1,i0),1)}\alpha_{(\overline{ID}_{t+1},i|_{(\theta+1,i0)})} + p_{((\theta+1,i0),2)}\widetilde{\alpha}_{(\overline{ID}_{t+1},i|_{(\theta+1,i0)})}$,
    $\widetilde{R}_{(\overline{ID}_{t+1},i|_{(\theta+1,i0)})} = p_{((\theta+1,i0),3)}\alpha_{(\overline{ID}_{t+1},i|_{(\theta+1,i0)})} + p_{((\theta+1,i0),4)}\widetilde{\alpha}_{(\overline{ID}_{t+1},i|_{(\theta+1,i0)})}$;
    3. Compute $s_{t\cdot(\theta+1)+1} = \mathcal{H}_5(s_{t\cdot(\theta+1)})$, and the following:

    (a)
    $$\begin{aligned}
    S_{(\overline{ID}_1,i0,t)} &= s_{t\cdot(\theta+1)}(\hat{I}_{(\theta+1,i0)} S_{(\overline{ID}_t,i)} + \hat{I}'_{(\theta+1,i0)} E_{(\overline{ID}_t,i)}) \\
    &\quad + \alpha_{(\overline{ID}_1,i0)} L^{(3)}_{t\cdot(\theta+1)+1}, \\
    E_{(\overline{ID}_1,i0,t)} &= s_{t\cdot(\theta+1)}(\tfrac{\hat{I}_{(\theta+1,i0)}\hat{I}''_{(\theta+1,i0)}}{\hat{I}'_{(\theta+1,i0)}} S_{(\overline{ID}_t,i)} + \hat{I}''_{(\theta+1,i0)} E_{(\overline{ID}_t,i)}) \\
    &\quad + \widetilde{\alpha}_{(\overline{ID}_1,i0)} L^{(3)}_{t\cdot(\theta+1)+1}, \\
    \widetilde{S}_{(\overline{ID}_1,i0,t)} &= s_{t\cdot(\theta+1)}(\widetilde{\hat{I}}_{(\theta+1,i0)} \widetilde{S}_{(\overline{ID}_t,i)} + \widetilde{\hat{I}}'_{(\theta+1,i0)} \widetilde{E}_{(\overline{ID}_t,i)}) \\
    &\quad + (\alpha_{(\overline{ID}_t,i0)} - R_{(\overline{ID}_t,i0)}) L^{(4)}_{t\cdot(\theta+1)+1}, \\
    \widetilde{E}_{(\overline{ID}_1,i0,t)} &= s_{t\cdot(\theta+1)}(\tfrac{\widetilde{\hat{I}}_{(\theta+1,i0)}\widetilde{\hat{I}}''_{(\theta+1,i0)}}{\widetilde{\hat{I}}'_{(\theta+1,i0)}} \widetilde{S}_{(\overline{ID}_t,i)} + \widetilde{\hat{I}}''_{(\theta+1,i0)} \widetilde{E}_{(\overline{ID}_t,i)}) \\
    &\quad + (\widetilde{\alpha}_{(\overline{ID}_t,i0)} - \widetilde{R}_{(\overline{ID}_t,i0)}) L^{(3)}_{t\cdot(\theta+1)+1}.
    \end{aligned}$$

    (b) Then, for $1 \leq j < t$, compute $s_{t\cdot(\theta+1)+2+j} = \mathcal{H}_5(s_{t\cdot(\theta+1)+1+j})$, and the following:

[3] $d_{(\overline{ID}_t,i0)}$ becomes $d_{(\overline{ID}_t,i+1)}$.

$$S_{(\overline{ID}_{j+1},i0,t)} = s_{t\cdot(\theta+1)+1+j}(\hat{\mathrm{I}}_{(\theta+1,i0)}S_{(\overline{ID}_j,i0,t)} + \hat{\mathrm{I}}'_{(\theta+1,i0)}E_{(\overline{ID}_j,i0)}) + \alpha_{(\overline{ID}_{j+1},i0)}L^{(3)}_{t\cdot(\theta+1)+2+j},$$

$$E_{(\overline{ID}_{j+1},i0,t)} = s_{t\cdot(\theta+1)+1+j}(\frac{\hat{\mathrm{I}}_{(\theta+1,i0)}\hat{\mathrm{I}}''_{(\theta+1,i0)}}{\hat{\mathrm{I}}'_{(\theta+1,i0)}}S_{(\overline{ID}_j,i0,t)} + \hat{\mathrm{I}}''_{(\theta+1,i0)}E_{(\overline{ID}_j,i0,t)}) + \widetilde{\alpha}_{(\overline{ID}_{j+1},i0)}L^{(3)}_{t\cdot(\theta+1)+2+j},$$

$$\widetilde{S}_{(\overline{ID}_{j+1},i0,t)} = s_{t\cdot(\theta+1)+1+j}(\widetilde{\hat{\mathrm{I}}}_{(\theta+1,i0)}\widetilde{S}_{(\overline{ID}_j,i0,t)} + \widetilde{\hat{\mathrm{I}}}'_{(\theta+1,i0)}\widetilde{E}_{(\overline{ID}_j,i0,t)}) + (\alpha_{(\overline{ID}_{j+1},i0)} - R_{(\overline{ID}_{j+1},i0)})L^{(4)}_{t\cdot(\theta+1)+2+j},$$

$$\widetilde{E}_{(\overline{ID}_{j+1},i0,t)} = s_{t\cdot(\theta+1)+1+j}(\frac{\widetilde{\hat{\mathrm{I}}}_{(\theta+1,i0)}\widetilde{\hat{\mathrm{I}}}''_{(\theta+1,i0)}}{\widetilde{\hat{\mathrm{I}}}'_{(\theta+1,i0)}}\widetilde{S}_{(\overline{ID}_j,i0,t)} + \widetilde{\hat{\mathrm{I}}}''_{(\theta+1,i0)}\widetilde{E}_{(\overline{ID}_j,i0,t)}) + (\widetilde{\alpha}_{(\overline{ID}_{j+1},i0)} - \widetilde{R}_{(\overline{ID}_{j+1},i0)})L^{(3)}_{t\cdot(\theta+1)+2+j}.$$

4. Thus, $d_{(\overline{ID}_t,i0)} = (S_{(\overline{ID}_t,i0)}, E_{(\overline{ID}_t,i0)}, \widetilde{S}_{(\overline{ID}_t,i0)}, \widetilde{E}_{(\overline{ID}_t,i0)}, R_{(\overline{ID}_{t+1},i0)}, \widetilde{R}_{(\overline{ID}_{t+1},i0)}, s_{t\cdot(\theta+2)})$ is obtained.
5. Likewise, $d_{(\overline{ID}_t,i1)} = (S_{(\overline{ID}_t,i1)}, E_{(\overline{ID}_t,i1)}, \widetilde{S}_{(\overline{ID}_t,i1)}, \widetilde{E}_{(\overline{ID}_t,i1)}, R_{(\overline{ID}_{t+1},i1)}, \widetilde{R}_{(\overline{ID}_{t+1},i1)}, s_{t\cdot(\theta+2)})$ is obtained.

- `Encryption` $(m, \overline{ID}_t, i)$: Given a message $m \in \mathcal{M}$, the ID-tuple $\overline{ID}_t$ of an intended recipient, given a time period i, and given *params*, this algorithm:
  1. Picks $r \in_R \mathbb{Z}_q^*$ and computes both $U_1 = rP_0^{(4)}$ and $U_2 = rP_0^{(3)}$.
  2. Computes $(J_1, \cdots, J_t, J_0, \cdots, J_0) = \mathcal{H}_1(\overline{ID}_t)$ and $(\hat{\mathrm{J}}_1, \cdots, \hat{\mathrm{J}}_\theta, \hat{\mathrm{J}}_0, \cdots, \hat{\mathrm{J}}_0) = \mathcal{H}_2([i])$, where $[i] = i_1 i_2 \cdots i_\theta$.
  3. Computes $\rho_t^{(1)} = A_{(t,\theta)}P_0^{(1)} + B_{(t,\theta)}P_0^{(2)}$ and $\rho_t^{(2)} = C_{(t,\theta)}P_0^{(5)} + D_{(t,\theta)}P_0^{(6)}$, where:

$$p_{(1,2)} = \mathcal{H}_7(p_{(1,3)}),\ p_{(1,4)} = \frac{p_{(1,2)}p_{(1,3)}}{p_{(1,1)}},\ I'_1 = -p_{(1,2)},$$
$$I''_1 = -\frac{p_{(1,1)}}{p_{(1,2)}}I'_1, \widetilde{I}_1 = I_1 + p_{(1,1)}, \widetilde{I}'_1 = I'_1 + p_{(1,2)}, \widetilde{I}''_1 = I''_1 + p_{(1,4)}$$
$$A_{(1,0)} = I_1, \quad B_{(1,0)} = I'_1, \quad C_{(1,0)} = \widetilde{I}_1, \quad D_{(1,0)} = \widetilde{I}'_1,$$
$$\widetilde{A}_{(1,0)} = \frac{I_1 I''_1}{I'_1}, \quad \widetilde{B}_{(1,0)} = I''_1, \quad \widetilde{C}_{(1,0)} = \frac{\widetilde{I}_1\widetilde{I}''_1}{\widetilde{I}'_1}, \quad \widetilde{D}_{(1,0)} = \widetilde{I}''_1,$$

  and, for $1 \le i < t$ and $1 \le a < \theta$:

$$\hat{\mathrm{I}}'_{a+1} = -p_{(a,2)},\ \hat{\mathrm{I}}''_{a+1} = -\frac{p_{(a,2)}p_{(a,3)}}{p_{(a,1)}},\ \widetilde{\hat{\mathrm{I}}}_{a+1} = \hat{\mathrm{I}}_{a+1} + p_{(a,1)},$$
$$\widetilde{\hat{\mathrm{I}}}'_{a+1} = \hat{\mathrm{I}}'_{a+1} + p_{(a,2)},\ \widetilde{\hat{\mathrm{I}}}''_{a+1} = \hat{\mathrm{I}}''_{a+1} + p_{(a,4)};$$
$$p_{(a+1,2)} = -\frac{p_{(a,1)}}{p_{(a,3)}}p_{(a+1,1)},\ p_{(a+1,4)} = \frac{p_{(a+1,2)}p_{(a+1,3)}}{p_{(a+1,1)}},$$
$$I'_{i+1} = -p_{(i,2)},\ I''_{i+1} = -\frac{p_{(i,2)}p_{(i,3)}}{p_{(i,1)}},\ \widetilde{I}_{i+1} = I_{i+1} + p_{(i,1)},$$
$$\widetilde{I}'_{i+1} = I'_{i+1} + p_{(i,2)},\ \widetilde{I}''_{i+1} = I''_{i+1} + p_{(i,4)};$$
$$p_{(i+1,2)} = -\frac{p_{(i,1)}}{p_{(i,3)}}p_{(i+1,1)},\ p_{(i+1,4)} = \frac{p_{(i+1,2)}p_{(i+1,3)}}{p_{(i+1,1)}},$$

$$A_{(i,a+1)} = \hat{I}_{a+1} A_{(i,a)} + \hat{I}'_{a+1} \widetilde{A}_{(i,a)} \quad \widetilde{A}_{(i,a+1)} = \frac{\hat{I}_{a+1}\hat{I}''_{a+1}}{\hat{I}'_{a+1}} A_{(i,a)} + \hat{I}''_{a+1} \widetilde{A}_{(i,a)},$$
$$A_{(i+1,0)} = I_{i+1} \widetilde{A}_{(i,\theta)} + I'_{i+1} \widetilde{A}_{(i,\theta)}, \quad \widetilde{A}_{(i+1,0)} = \frac{I_{i+1} I''_{i+1}}{I'_{i+1}} \widetilde{A}_{(i,\theta)} + I''_{i+1} \widetilde{A}_{(i,\theta)},$$
$$B_{(i,a+1)} = \hat{I}_{a+1} B_{(i,a)} + \hat{I}'_{a+1} \widetilde{B}_{(i,a)} \quad \widetilde{B}_{(i,a+1)} = \frac{\hat{I}_{a+1}\hat{I}''_{a+1}}{\hat{I}'_{a+1}} B_{(i,a)} + \hat{I}''_{a+1} \widetilde{B}_{(i,a)},$$
$$B_{(i+1,0)} = I_{i+1} \widetilde{B}_{(i,\theta)} + I'_{i+1} \widetilde{B}_{(i,\theta)}, \quad \widetilde{B}_{(i+1,0)} = \frac{I_{i+1} I''_{i+1}}{I'_{i+1}} \widetilde{B}_{(i,\theta)} + I''_{i+1} \widetilde{B}_{(i,\theta)},$$
$$C_{(i,a+1)} = \widetilde{\hat{I}}_{a+1} C_{(i,a)} + \widetilde{\hat{I}}'_{a+1} \widetilde{C}_{(i,a)} \quad \widetilde{C}_{(i,a+1)} = \frac{\widetilde{\hat{I}}_{a+1}\widetilde{\hat{I}}''_{a+1}}{\widetilde{\hat{I}}'_{a+1}} C_{(i,a)} + \widetilde{\hat{I}}''_{a+1} \widetilde{C}_{(i,a)},$$
$$C_{(i+1,0)} = \widetilde{I}_{i+1} \widetilde{C}_{(i,\theta)} + \widetilde{I}'_{i+1} \widetilde{C}_{(i,\theta)}, \quad \widetilde{C}_{(i+1,0)} = \frac{\widetilde{I}_{i+1} \widetilde{I}''_{i+1}}{\widetilde{I}'_{i+1}} \widetilde{C}_{(i,\theta)} + \widetilde{I}''_{i+1} \widetilde{C}_{(i,\theta)},$$
$$D_{(i,a+1)} = \widetilde{\hat{I}}_{a+1} D_{(i,a)} + \widetilde{\hat{I}}'_{a+1} \widetilde{D}_{(i,a)} \quad \widetilde{D}_{(i,a+1)} = \frac{\widetilde{\hat{I}}_{a+1}\widetilde{\hat{I}}''_{a+1}}{\widetilde{\hat{I}}'_{a+1}} D_{(i,a)} + \widetilde{\hat{I}}''_{a+1} \widetilde{D}_{(i,a)},$$
$$D_{(i+1,0)} = \widetilde{I}_{i+1} \widetilde{D}_{(i,\theta)} + \widetilde{I}'_{i+1} \widetilde{D}_{(i,\theta)}, \quad \widetilde{D}_{(i+1,0)} = \frac{\widetilde{I}_{i+1} \widetilde{I}''_{i+1}}{\widetilde{I}'_{i+1}} \widetilde{D}_{(i,\theta)} + \widetilde{I}''_{i+1} \widetilde{D}_{(i,\theta)},$$

4. computes $V = m \oplus \mathcal{H}_2\left(\left(\hat{e}(L^{(4)}_{t\cdot(\theta+1)}, \rho^{(1)}_{(t,\theta)}) \cdot \hat{e}(L^{(3)}_{t\cdot(\theta+1)}, \rho^{(2)}_{(t,\theta)})^{-1}\right)^r\right)$;
5. outputs the ciphertext $c = (U_1, U_2, V)$.

- `Decryption` $(c,\ i,\ d_{(\overline{ID}_t,i)})$: Given a ciphertext $c = (U_1, U_2, V) \in \mathcal{C}$ which was issued for time period i, given the private key $d_{(\overline{ID}_t,i)}$ of a recipient $\overline{ID}_t$, and given *params*, this algorithm computes and outputs the message:

$$m = V \oplus \mathcal{H}_2(\hat{e}(U_1, S_{(\overline{ID}_t,i)})\hat{e}(U_2, \widetilde{S}_{(\overline{ID}_t,i)})^{-1}).$$

Note that the provable security of the above scheme could be enhanced by the application of Fujisaki-Okamoto padding [FO99]. Since this enhancement mechanism is generic (cf. [BF01, GS02, NAM05b]), we simply refer to §4.3, for a concrete example of how this could be done. Moreover, we note that Yao et al. showed that their fsHIBE scheme is secure, in the random oracle model, assuming the intractability of the bilinear Diffie-Hellman (BDH) problem, when the underlying HIBE scheme is IND-HIB-CCA secure under the same number theoretic assumption [YFDL04]. Since NewHIBE can be proved to be IND-HIB-CCA secure if the BDH problem is intractable (cf. the introduction of §4.3), our proposed fsHIBE scheme is hence secure. This implies that the HTIR scheme derived the proposed fsHIBE scheme is also secure, in the random oracle model, if the BDH problem is intractable.

In this section, we have shown that NewHIBE yields an efficient and secure fsHIBE scheme. In the next section, we explain that NewHIBS yields an efficient append-only signature scheme.

5.2 Append-Only Signature

This section explains how to derive an efficient append-only signature (AOS) scheme from the hierarchical ID-based signature scheme presented in Chapter 4 (i.e. from

NewHIBS). §5.2.1 provides motivations for the design of AOS schemes, and §5.2.2 identifies the computational benefits of a NewHIBS-derived AOS scheme. Then, §5.2.3 presents the methodology to construct our proposed AOS scheme.

5.2.1 Motivation for AOS scheme

Recall that AOS signatures have the following property: given an AOS signature on a message m_1, any user is able to issue an AOS signature on the concatenation of m_1 with any message m_2 only. Moreover, the procedure whereby an AOS signature is issued on an appropriate concatenated message is called the *Append* algorithm of this AOS scheme. AOS schemes also include a *Setup* algorithm (for the definition of system-level secret and public parameters), a *Key Generation* algorithm, and a *Verification* algorithm (to determine the validity of any given AOS signature).

Privilege Delegation Systems

The "append-only" property of AOS schemes is needed in many applications which involve privilege delegation. Suppose for instance that a system manages privileges, and that users are given privileges which they can delegate to others. Then users should only be able to delegate privileges which they have received from authorized parties (whether this authorized party be a user or a trusted privilege server (TPS)). Thus, AOS signatures can be issued on messages which orderly identify all delegated privileges, all delegates'public keys, and, potentially, the identities of all delegates. Any third party could then determine whether a given chain of delegated privileges includes privileges which a delegator did not have.

Miltchev et al. [MPI+03] designed such a system, in order to support global file sharing, *without requiring users to register in a local TPS.* Note that this feature is suitable for large scale and distributed sharing of restricted resources. Imagine, for instance, that each user of this system gives, to an unpredictable group of friends, the ability to access restricted multimedia information (e.g. photos, music, or video files). Then each friend could be able to delegate (in a recursive fashion) this access right to other friends, and any authorized friend would be able to access the restricted multimedia information (without having to create an account at the original delegator's TPS).

More precisely, Miltchev et al.'s system implements privilege delegation as follows. Suppose that a user $\mathcal{A}$ has access to a file ϕ, and wants to give a user $\mathcal{B}$ access to ϕ. Then, $\mathcal{A}$ creates a credential that contains $\mathcal{B}$'s public key and ϕ's file handle. Next, $\mathcal{A}$ signs the credential, and gives $\mathcal{B}$ both the credential and its associated signature. $\mathcal{B}$ is then able to access ϕ by showing a cryptographic signature issued on a random message with the private key which matches the public key specified

in the aforementioned credential[4]. Hence, when an AOS scheme is used to support this system, $\mathcal{A}$ simply gives $\mathcal{B}$ an AOS signature on a message whose suffix includes ϕ's file handle and $\mathcal{B}$'s public key. $\mathcal{B}$ can then append this signature with a dummy character, and give the appended signature to the party who controls ϕ's access.

One feature of this file sharing system is that delegation chains are likely to be long, because Web-based sharing of multimedia information is popular. (In fact, the ability to handle long delegation was a design requirement of Miltchev et al.'s system.) As we will see in §5.2.2, this is precisely the class of scenarios for which our proposed AOS scheme is suitable.

It should be noted that Miltchev et al.'s system uses a cryptographic construct which is similar to an AOS scheme, but which does not allow the verification of a chain of delegations to be done using a TPS's public key only. Since HIBS signature verification only requires the obtainment of public parameters associated with a root PKG, we note that AOS signatures would be particularly suitable to support the aforementioned file sharing system.

As a summary: the design of an AOS scheme is motivated by the possibility to support convenient file sharing in large and distributed user communities.

Internet Routing Protocols

AOS schemes can also be used to support path authenticity in Internet routing protocols.

The Internet consists of many routing domains, where each domain (also called an *autonomous system*) is essentially a set of routers under a single technical administration. Routing inside a domain is determined by an intra-domain routing protocol (e.g. OSPF [Moy98]), and routing between domains is governed by an inter-domain routing protocol (e.g. BGP [LR89]).

The primary function of the border gateway protocol (BGP) is to exchange network reachability information between autonomous systems (ASes). Network paths between ASes are called AS paths, and BGP broadcasts AS paths so that each AS can route network messages using the shortest path between itself and the destination ASes associated these network messages. Hence, BGP requires trust in the accuracy of announced AS paths. This can be ensured by using AOS signatures in the following way. Let $\mathcal{A}_0$ be an AS. For each AS neighbor $\mathcal{A}_1$ of $\mathcal{A}_0$, $\mathcal{A}_0$ issues an AOS signature σ_1 on the message $ID_{\mathcal{A}_0}$, where ID_X is the identifier of any AS X. Then $\mathcal{A}_1$ uses σ_1 to produce an AOS signature σ_2 on the message $ID_{\mathcal{A}_0}||ID_{\mathcal{A}_1}$ for each AS neighbor

[4]Note that the random message should be a challenge sent by the entity enforcing access.

Schemes		Computational Requirements
BBG_2-AOS	Public Parameters	$M_{\mathbb{G}_1} + \mathbf{P} + (2\sqrt{t}+1)R_{\mathbb{Z}_q^*}$
	Append	$(3+\sqrt{t})A_{\mathbb{G}_1} + (3+\sqrt{t})M_{\mathbb{G}_1} + R_{\mathbb{Z}_q^*}$
	Verification	$tA_{\mathbb{G}_1} + (1+\sqrt{t}+t)M_{\mathbb{G}_1} + (1+\sqrt{t})\mathbf{P}$
	Signature Length	$(2\sqrt{t}+1)\lvert\mathbb{G}_1\rvert$
NewAOS	Public Parameters	$(\ell-1)H + 2\ell M_{\mathbb{G}_1} + 4R_{\mathbb{G}_1} + 2R_{\mathbb{Z}_q^*}$
	Append	$8A_{\mathbb{G}_1} + 7A_{\mathbb{Z}_q^*} + 6H + 3Inv_{\mathbb{Z}_q^*} + 12M_{\mathbb{G}_1} + 22M_{\mathbb{Z}_q^*} + 2R_{\mathbb{Z}_q^*}$
	Verification	$5A_{\mathbb{G}_1} + (11t-8)A_{\mathbb{Z}_q^*} + (3t+2)H + (5t-1)Inv_{\mathbb{Z}_q^*}$ $+ 6M_{\mathbb{G}_1} + 3M_{\mathbb{G}_2} + (24t-16)M_{\mathbb{Z}_q^*} + 5\mathbf{P} + R_{\mathbb{Z}_q^*}$
	Signature Length	$6\lvert\mathbb{G}_1\rvert + \lvert\mathbb{Z}_q^*\rvert$

Table 5.2: Efficiency Comparison of BBG-AOS and NewAOS.

$\mathcal{A}_2$ of $\mathcal{A}_1$. This is performed recursively so that each AS for which there is a k-long path from $\mathcal{A}_0$ to $\mathcal{A}_k$ obtains an AOS signature σ_k on the message $ID_{\mathcal{A}_0}||\cdots||ID_{\mathcal{A}_{k-1}}$.

Kiltz et al. [KMPR05] described a mechanism to derive AOS schemes from HIBS schemes. When this procedure is used, then only the public parameters of an originating AS are needed to verify the validity of AOS signatures that are issued on messages which append the identifier of this originating AS. Hence, AOS signature can support path authentication in both inter-domain and intra-domain Internet routing, and HIBS-based AOS signatures are particularly convenient for the verification of paths'validity.

5.2.2 Benefits of Proposed AOS Scheme

Table 5.2 compares the computational and space requirements of the AOS scheme described in §5.2.3 (i.e. NewAOS) and the HIBS-based AOS scheme described by Kiltz et al. [KMPR05] (BBG_2-AOS[5]). ℓ denotes the maximal length of an HIBS-bases AOS signature (i.e. the maximal number of messages (e.g. file handle and public key) which can be concatenated), and t denotes the length of an arbitrary HIBS-based AOS signature. $|X|$ denotes the bit size of any element of X, and all other symbols are defined as in the previous chapters. (Note also that $\sqrt{t}$ should always be replaced by $\lceil\sqrt{t}\rceil$ whenever t is not the square of an integer.)

[5]Since Kiltz et al.'s HIBS-based AOS scheme is derived from a variant of BBG-HIBE's key generation scheme, we call BBG_2-AOS this AOS scheme.

Schemes		Computational Requirements					
		$t = 1$, $\ell = 2$	$t = 3$, $\ell = 4$	$t = 4$, $\ell = 5$	$t = 10$, $\ell = 11$	$t = 12$, $\ell = 13$	$t = 20$, $\ell = 21$
BBG_2-AOS	Public Parameters	105×10^3	105×10^3	105×10^3	105×10^3	105×10^3	105×10^3
	Append	85×10^3	100×10^3	106×10^3	131×10^3	137×10^3	158×10^3
	Verification	231×10^3	379×10^3	400×10^3	737×10^3	780×10^3	1.1×10^6
	Signature Length	$3\lvert\mathbb{G}_1\rvert$	$5\lvert\mathbb{G}_1\rvert$	$5\lvert\mathbb{G}_1\rvert$	$8\lvert\mathbb{G}_1\rvert$	$8\lvert\mathbb{G}_1\rvert$	$10\lvert\mathbb{G}_1\rvert$
NewAOS	Public Parameters	42×10^3	168×10^3	210×10^3	462×10^3	546×10^3	882×10^3
	Append	255×10^3	255×10^3	255×10^3	255×10^3	255×10^3	255×10^3
	Verification	548×10^3	551×10^3	553×10^3	562×10^3	565×10^3	578×10^3
	Signature Length	$6\lvert\mathbb{G}_1\rvert + \lvert\mathbb{Z}_q^*\rvert$	$6\lvert\mathbb{G}_1\rvert + \lvert\mathbb{Z}_q^*\rvert$	$6\lvert\mathbb{G}_1\rvert + \lvert\mathbb{Z}_q^*\rvert$	$6\lvert\mathbb{G}_1\rvert + \lvert\mathbb{Z}_q^*\rvert$	$6\lvert\mathbb{G}_1\rvert + \lvert\mathbb{Z}_q^*\rvert$	$6\lvert\mathbb{G}_1\rvert + \lvert\mathbb{Z}_q^*\rvert$

Table 5.3: Efficiency Comparison of BBG-AOS and NewAOS.

Table 5.2 shows that NewAOS has a *Setup* algorithm which is computationally more expensive than the BBG_2-AOS's *Setup* procedure ($2\ell M_{\mathbb{G}_1}$ instead of $1\dot{M}_{\mathbb{G}_1}$). Note, however, that this is a one-time cost which is incurred at startup time.

A closer look at Table 5.2 shows that, NewAOS's Verification algorithm is faster than its BBG_2-AOS analogue, when $t \geq 8$, that NewOAS signatures are shorter than their BBG_2-AOS counterparts when $t \geq 10$, and that BBG_2-AOS's Append algorithm is faster than NewAOS's for practical values of t (i.e. $t \leq 30$). (See also Table 5.3, which presents timing estimates of BBG_2-AOS and NewAOS, based on information presented in [HPS02].)

Hence, the AOS scheme described in §5.2.3 is advantageous when a system performs a large number of AOS signature verification on medium and long messages. This is certainly the case when AOS signatures are used to support privilege delegation *à* la Miltchev et al. [MPI+03], and this also the case in the context of intra-domain routing (for various network topologies). However, we are not certain whether empirical distributions of AS path lengths in BGP favor BBG_2-AOS over NewAOS.

5.2.3 HIBS-based AOS Scheme

Kiltz et al. [KMPR05] recently described a method to transform any HIBS scheme into an AOS scheme. Since this procedure is straightforward (unlike the methodology used by Yao et al. [YFDL04] to derive fsHIBE schemes from any HIBE scheme), we simply present an overview of the method, instead of formally describing NewAOS.

Kiltz et al.'s method works as follows. Let SIG be a HIBS scheme and $\overline{ID}_t =$

$(ID_1, \cdots, ID_t)$ be an ID-tuple of SIG's hierarchy. The AOS scheme derived from SIG has the same setup algorithm as SIG. Moreover, an AOS signature on $\overline{ID}_t$ consists of the HIBS signing key of $\overline{ID}_t$. Indeed, note that the only HIBS secret keys which $\overline{ID}_t$ can generate are those of its hierarchical descendants, starting with the key of any of its children (say $\overline{ID}_{t+1} = (ID_1, \cdots, ID_t, ID_{t+1})$). Thus, the *Append* procedure of the AOS signature scheme derived from SIG is SIG's *Key Generation* algorithm[6]. Furthermore, the *Verification* algorithm of the derived AOS scheme consists of the following steps. Let $\overline{ID}_t$ be the ID-tuple associated with a given AOS signer. Then, SIG is first used to issue a signature σ on any message m. Then, SIG is used to verify the validity of σ with respect to m and $\overline{ID}_t$. Consequently, NewAOS (the AOS scheme derived from NewHIBS) is as (computationally and space-) efficient as NewHIBS.

5.3 Conclusion

In this Chapter, we have shown that NewHIBE can be used to construct an efficient fsHIBE scheme, and that NewHIBS yields an AOS scheme which is particularly efficient when relatively long AOS signatures are handled.

fsHIBE limits the security-impact of private key compromise, and can be used to support role-based access control. fsHIBE can also be used to support time-based release of confidential information in large user communities. AOS signatures, on the other side, can be used to support both path authentication in Internet routing, and convenient resource file sharing, in large communities. In the next chapter, we present the *second part* of the second main contribution of this dissertation, namely the design a hierarchical ID-based signcryption scheme which supports encrypted message authentication.

[6]Note that, given an AOS signature, any party can append this signature with an arbitrary message. While this property may seem undesirable at first sight, it is actually useful for the applications discussed in this chapter (i.e. path authenticity and privilege delegation). In the case of path authenticity, for instance, any AS that receives an OAS signature appends this signature and sends the appended signature to all its neighbors (except the sender of the received AOS signature). Thus, any receiver of an OAS signature (e.g. a user or an AS) is able to determine whether an AS path is authentic and of minimal length (compared to other AS paths).

Chapter 6

Hierarchical ID-based Signcryption

Signcryption schemes with public ciphertext authenticity enables third parties (such as firewalls) to stop or route encrypted messages based on the public keys of these messages'senders. Identity-based (ID-based) cryptographic schemes allow public keys to be derived from identifiers (such as email and IP addresses) – thereby exempting users and their collaborators from having to use both public-key certificates and the associated (cumbersome) infrastructures. In large or hierarchically-structured user communities, it has been recommended [HL02, GS02] (for increased efficiency and suitability reasons) to use *hierarchical* ID-based cryptographic schemes. This chapter describes an efficient hierarchial ID-based signcryption (HIBSC) scheme providing both public ciphertext authenticity and forward security. The security of our signcryption scheme is examined in the random oracle model, and its efficiency is compared with that of related schemes.

§6.1 reviews our motivation for the design of a HIBSC scheme, and §6.2 reviews standard terminology concerning HIBSC. §6.3 describes our proposed HIBSC scheme, and §6.4 discusses its efficiency. The security guarantees of the proposed HIBSC scheme are summarized in §6.5, and §6.6 concludes the chapter.

6.1 Introduction

Signcryption [Zhe97] enables users to sign and encrypt messages more efficiently than by sequentially signing a message and encrypting both this message and its signature. Thus, signcryption provides efficiency, and preserves the security benefits of both encryption and cryptographic signature (i.e. data confidentiality, user authentication, and commitment non-repudiation).

Two desirable properties of signcryption schemes are public ciphertext authenticity (PCA) and forward-security (FS).

PCA [CYHC03, GLZ99] enables third parties to determine, without the help of any intended recipient, whether a signcrypted message has been modified (since it was first signcrypted), and whether a claimed user is the issuer of this signcrypted message. PCA can therefore be used by firewalls in order to block signcrypted messages which do not originate from trusted users, and by public file servers in order to identify signcryptexts issued by untrusted users. Indeed, these untrusted users could flood networks and public file servers with signcryptexts of random data, and thereby cause either denial of service attacks or significant delays in the retrieval of trusted data. Hence, the use of signcryption with PCA is more suitable, in practice, than the use of encryption (whether or not the encrypted data includes a cryptographic signature).

FS [LQ03b], on the other hand, ensures that the compromise of user's signing key does not compromise the confidentiality of ciphertexts issued by this user. This is useful because the threats of private key compromised are realistic when cryptographic operations are executed on inter-connected personal computers.

We focus, in this chapter, on *hierarchical ID-based* signcryption because of the convenience of use which ID-based schemes provide, and because of the scalability which hierarchical (ID-based) schemes offer. Moreover, we are not aware of any hierarchical ID-based signcryption scheme offering PCA, and the only known hierarchical ID-based signcryption scheme [CYHY04] is computationally expensive.

6.2 Preliminaries

In this section, we recall fundamental definitions related to HIBSC. Readers familiar with [GS02] and [CYHY04] may go to §6.3.

ID-tuples

For the description of the proposed HIBSC scheme, PKGs are assumed to be hierarchically organized. We only consider tree-shaped hierarchies, and denote by *user* any PKG except the *root PKG*. This root PKG is the root of the tree-shaped hierarchy, and is denoted by $rPKG$. Every user is identified with a tuple $\overline{ID}_t = (ID_1, ID_2, \cdots, ID_t)$, as in the previous chapters.

Hierarchical ID-based Signcryption Scheme

Each HIBSC scheme is composed of four algorithms, whose functions are described below:

1. `Root Setup`: Given a security parameter k, this algorithm is used by the root PKG to return a tuple *params* of system parameters. These parameters include

a description of the message space $\mathcal{M}$ and the ciphertext space $\mathcal{C}$, along with a secret piece of data s_0 called the *root secret key.* Other parameters are allowed, *provided they are not unique to any user.* Ideally, these other parameters should not be unique to a strict subset of users (*including users who all belong the same hierarchical level.*). However, this latter optimization is not mandatory. Some parameters may be public (including those describing $\mathcal{M}$ and $\mathcal{S}$), while others remain secret (including the root secret key).

2. `Key Extraction`: Given the scheme's public parameters, an arbitrary identifier ID_{t+1}, and the private key $d_{\overline{ID}_t}$ of a PKG identified by $\overline{ID}_t$, this algorithm returns the private key $d_{\overline{ID}_{t+1}}$ of the *child* $\overline{ID}_{t+1}$ of $\overline{ID}_t$.

3. `Signcryption`: Given a message $m \in \mathcal{M}$, the ID-tuple $\overline{ID}_{t_A}$ of a sender, the ID-tuple $\overline{ID}_{t_B}$ of an intended recipient, the private key $d_{\overline{ID}_{t_A}}$ of $\overline{ID}_{t_A}$, and the scheme's public parameters, this algorithm returns a signcryptext $\sigma \in \mathcal{C}$ corresponding to m.

4. `Unsigncryption`: Given a signcryptext $\sigma \in \mathcal{C}$, the ID-tuple $\overline{ID}_{t_A}$ of a sender, the ID-tuple $\overline{ID}_{t_B}$ of an intended recipient, the private key $d_{\overline{ID}_{t_B}}$ of $\overline{ID}_{t_B}$, and the scheme's public parameters, this algorithm returns either "Invalid" or a pair $(m,$"Valid"$)$, where $m \in \mathcal{M}$ is the message associated with σ.

Signcryption and *Unsigncryption* must satisfy the following consistency constraint:

$$\forall\, m \in \mathcal{M}\ \text{Unsigncryption}(\sigma, \overline{ID}_{t_A}, \overline{ID}_{t_B}, d_{\overline{ID}_{t_B}}, params) = (m, \text{"valid"}),$$
$$\text{if } \sigma = \text{Signcryption}(m, \overline{ID}_{t_A}, \overline{ID}_{t_B}, d_{\overline{ID}_{t_A}}, params).$$

6.3 Proposed Signcryption Scheme

- `Instance Generator` (k). This procedure, denoted by IG, is a randomized algorithm which takes a security parameter $k > 0$, runs in $O(k)$, and outputs $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are two Abelian groups of prime order $q \geq 2^k$, and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is an admissible pairing with respect to which $\mathbb{G}_1$ and $\mathbb{G}_2$ are Gap-Diffie-Hellman groups.

- `Root Setup` (k). Given a security parameter $k > 0$, the root *PKG*:

  1. runs IG with input k and obtains $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$.
  2. picks, randomly and uniformly[1], $P_0^{(1)}, P_0^{(2)}, P_0^{(3)}, P_0^{(4)}, P_0^{(5)}, P_0^{(6)}, P_0^{(7)} \in \mathbb{G}_1$;

[1] In the sequel, we shall use the notation $x \in_R X$ to indicate that the element x is chosen uniformly at random from the set X.

3. picks $s_0, I_0 \in_R \mathbb{Z}_q^*$;
4. computes $n = poly_1(k)$ and $\ell = \log(k)$, $\xi = poly_2(k)$, where $poly_1$ and $poly_2$ are polynomials over the positive integers (n is the message length, ℓ is the maximal depth of the user hierarchy, and ξ is a bit size of a symmetric cipher's keys);
5. chooses cryptographic hash functions:
   $\mathcal{H}_1 : \{0,1\}^* \to (\mathbb{Z}_q^*)^{3\ell}$, $\mathcal{H}_2 : \mathbb{G}_2 \to \{0,1\}^n$,
   $\mathcal{H}_3 : \{0,1\}^n \times \mathbb{G}_2 \to \{0,1\}^n$, $\mathcal{H}_4 : \mathbb{Z}_q^* \to \mathbb{Z}_q^*$,
   where $\mathcal{H}_1$ is defined as in NewHIBE.
6. computes, $(s_i = \mathcal{H}_4(s_{i-1}))_{i=1}^{\ell-1}$, $L_1^{(j)} = s_0 P_0^{(j)}$ for $j = 3, 4$, and $\left(L_i^{(j)} = s_{i-1} L_{i-1}^{(j)}\right)_{i=2}^{\ell}$ for $j = 3, 4$.
7. chooses a symmetric cipher $E : \{0,1\}^n \to \{0,1\}^n$ whose keys belong the set $\{0,1\}^\xi$ and whose inverse is denoted by E^{-1}. For instance, E can be instantiated with AES [oSN99].

The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = \mathbb{G}_1^5 \times \{0,1\}^n \times \{0,1\}^n$. The system's public parameters (which must be certified) are $pubParams = (q, n, \hat{e}, I_0, P_0^{(1)}, P_0^{(2)}, P_0^{(3)}, P_0^{(4)}, P_0^{(5)}, P_0^{(6)}, P_0^{(7)}, E, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4, ((L_i^{(j)})_{i=1}^{\ell})_{j=3}^{4})$, and the *root PKG* keeps s_0 secret, so that $params = (pubParams, s_0)$.

- `Extract`: Same as NewHIBE.
- `Signcryption`: Given a message $m \in \mathcal{M}$, the ID-tuple $\overline{ID}_{t_a}$ of a sender, the ID-tuple $\overline{ID}_{t_b}$ of an intended recipient, the private key $d_{\overline{ID}_{t_a}}$ of $\overline{ID}_{t_a}$, and $pubParams$, this algorithm:
  1. picks $x_1, y \in_R \mathbb{Z}_q^*$, and computes $x = \mathcal{H}_5(x_1)$, $U_1 = xP_0^{(4)}$, $U_2 = xP_0^{(3)}$ and $V = y(P_0^{(4)} - P_0^{(3)})$;
  2. computes $(J_1, \cdots, J_{t_b}, J_0, \cdots, J_0) = \mathcal{H}_1(\overline{ID}_{t_b}) \in (\mathbb{Z}_q^*)^{3\ell}$, $\rho_{t_b}^{(1)} = A_{t_b} P_0^{(1)} + B_{t_b} P_0^{(2)}$, and $\rho_{t_b}^{(2)} = C_{t_b} P_0^{(5)} + D_{t_b} P_0^{(6)}$, where

$$p_{(1,2)} = \mathcal{H}_7(p_{(1,3)}),\ p_{(1,4)} = \frac{p_{(1,2)} p_{(1,3)}}{p_{(1,1)}},\ I_1' = -p_{(1,2)},$$
$$I_1'' = -\frac{p_{(1,1)}}{p_{(1,2)}} I_1', \widetilde{I}_1 = I_1 + p_{(1,1)}, \widetilde{I}_1' = I_1' + p_{(1,2)}, \widetilde{I}_1'' = I_1'' + p_{(1,4)}$$
$$A_1 = I_1, \quad B_1 = I_1', \quad C_1 = \widetilde{I}_1, \quad D_1 = \widetilde{I}_1',$$
$$\widetilde{A}_1 = \frac{I_1 I_1''}{I_1'}, \quad \widetilde{B}_1 = I_1'', \quad \widetilde{C}_1 = \frac{\widetilde{I}_1 \widetilde{I}_1''}{\widetilde{I}_1'}, \quad \widetilde{D}_1 = \widetilde{I}_1'',$$

and, for $1 \le i < t_b$,

$$I_{i+1}' = -p_{(i,2)},\ I_{i+1}'' = -\frac{p_{(i,2)} p_{(i,3)}}{p_{(i,1)}},\ \widetilde{I}_{i+1} = I_{i+1} + p_{(i,1)},$$
$$\widetilde{I}_{i+1}' = I_{i+1}' + p_{(i,2)},\ \widetilde{I}_{i+1}'' = I_{i+1}'' + p_{(i,4)};$$
$$p_{(i+1,2)} = -\frac{p_{(i,1)}}{p_{(i,3)}} p_{(i+1,1)},\ p_{(i+1,4)} = \frac{p_{(i+1,2)} p_{(i+1,3)}}{p_{(i+1,1)}},$$

$$A_{i+1} = I_{i+1}A_i + I'_{i+1}\widetilde{A}_i, \text{ and} \quad \widetilde{A}_{i+1} = \frac{I_{i+1}I''_{i+1}}{I'_{i+1}}A_i + I''_{i+1}\widetilde{A}_i,$$
$$B_{i+1} = I_{i+1}B_i + I'_{i+1}\widetilde{B}_i, \text{ and} \quad \widetilde{B}_{i+1} = \frac{I_{i+1}I''_{i+1}}{I'_{i+1}}B_i + I''_{i+1}\widetilde{B}_i,$$
$$C_{i+1} = \widetilde{I}_{i+1}C_i + \widetilde{I}'_{i+1}\widetilde{C}_i, \text{ and} \quad \widetilde{C}_{i+1} = \frac{\widetilde{I}_{i+1}\widetilde{I}''_{i+1}}{\widetilde{I}'_{i+1}}C_i + \widetilde{I}''_{i+1}\widetilde{C}_i,$$
$$D_{i+1} = \widetilde{I}_{i+1}D_i + \widetilde{I}'_{i+1}\widetilde{D}_i, \text{ and} \quad \widetilde{D}_{i+1} = \frac{\widetilde{I}_{i+1}\widetilde{I}''_{i+1}}{\widetilde{I}'_{i+1}}D_i + \widetilde{I}''_{i+1}\widetilde{D}_i,$$

3. computes $\kappa_1 = \hat{e}(P_0^{(4)} - P_0^{(3)}, P_0^{(7)})^y$ and $\kappa_2 = \mathcal{H}_2((\hat{e}(L_{t_b}^{(4)}, \rho_{t_b}^{(1)}) \cdot \hat{e}(L_{t_b}^{(3)}, \rho_{t_b}^{(2)})^{-1})^x)$;
4. computes $c = E_{\kappa_2}(m)$, $r = \mathcal{H}_3(c, \kappa_1)$, $\underline{S}_{t_a} = yP_0^{(7)} - rS_{t_a}$, and $\widetilde{\underline{S}}_{t_a} = -yP_0^{(7)} + r\widetilde{S}_{t_a}$;
5. outputs $\sigma = (U_1, U_2, V, \underline{S}_{t_a}, \widetilde{\underline{S}}_{t_a}, c, r)$.

- `Unsigncryption`: Given a ciphertext $\sigma = (U_1, U_2, V, \underline{S}_{t_a}, \widetilde{\underline{S}}_{t_a}, c, r) \in \mathcal{C}$, the ID-tuple $\overline{ID}_{t_a}$ of a sender, the ID-tuple $\overline{ID}_{t_b}$ of an intended recipient, the private key $d_{\overline{ID}_{t_b}}$ of $\overline{ID}_{t_b}$, and $pubParams$, this algorithm:

1. computes $(J_1, \cdots, J_{t_a}, J_0, \cdots, J_0) = \mathcal{H}_1(\overline{ID}_{t_a}) \in (\mathbb{Z}_q^*)^{3\ell}$, $\rho_{t_a}^{(1)} = \underline{A}_{t_a}P_0^{(1)} + \underline{B}_{t_a}P_0^{(2)}$, and $\rho_{t_a}^{(2)} = \underline{C}_{t_a}P_0^{(5)} + \underline{D}_{t_a}P_0^{(6)}$, where
$p_{(1,2)} = \mathcal{H}_7(p_{(1,3)})$, $p_{(1,4)} = \frac{p_{(1,2)}p_{(1,3)}}{p_{(1,1)}}$, $I'_1 = -p_{(1,2)}$,
$I''_1 = -\frac{p_{(1,1)}}{p_{(1,2)}}I'_1, \widetilde{I}_1 = I_1 + p_{(1,1)}, \widetilde{I}'_1 = I'_1 + p_{(1,2)}, \widetilde{I}''_1 = I''_1 + p_{(1,4)}$
$\underline{A}_1 = I_1, \quad \underline{B}_1 = I'_1, \quad \underline{C}_1 = \widetilde{I}_1, \quad \underline{D}_1 = \widetilde{I}'_1,$
$\widetilde{\underline{A}}_1 = \frac{I_1 I''_1}{I'_1}, \quad \widetilde{\underline{B}}_1 = I''_1, \quad \widetilde{\underline{C}}_1 = \frac{\widetilde{I}_1\widetilde{I}''_1}{\widetilde{I}'_1}, \quad \widetilde{\underline{D}}_1 = \widetilde{I}''_1,$
and, for $1 \leq i < t_a$,
$I'_{i+1} = -p_{(i,2)}$, $I''_{i+1} = -\frac{p_{(i,2)}p_{(i,3)}}{p_{(i,1)}}$, $\widetilde{I}_{i+1} = I_{i+1} + p_{(i,1)}$,
$\widetilde{I}'_{i+1} = I'_{i+1} + p_{(i,2)}$, $\widetilde{I}''_{i+1} = I''_{i+1} + p_{(i,4)}$;
$p_{(i+1,2)} = -\frac{p_{(i,1)}}{p_{(i,3)}}p_{(i+1,1)}$, $p_{(i+1,4)} = \frac{p_{(i+1,2)}p_{(i+1,3)}}{p_{(i+1,1)}}$,
$\underline{A}_{i+1} = I_{i+1}\underline{A}_i + I'_{i+1}\widetilde{\underline{A}}_i$, and $\widetilde{\underline{A}}_{i+1} = \frac{I_{i+1}I''_{i+1}}{I'_{i+1}}\underline{A}_i + I''_{i+1}\widetilde{\underline{A}}_i$,
$\underline{B}_{i+1} = I_{i+1}\underline{B}_i + I'_{i+1}\widetilde{\underline{B}}_i$, and $\widetilde{\underline{B}}_{i+1} = \frac{I_{i+1}I''_{i+1}}{I'_{i+1}}\underline{B}_i + I''_{i+1}\widetilde{\underline{B}}_i$,
$\underline{C}_{i+1} = \widetilde{I}_{i+1}\underline{C}_i + \widetilde{I}'_{i+1}\widetilde{\underline{C}}_i$, and $\widetilde{\underline{C}}_{i+1} = \frac{\widetilde{I}_{i+1}\widetilde{I}''_{i+1}}{\widetilde{I}'_{i+1}}\underline{C}_i + \widetilde{I}''_{i+1}\widetilde{\underline{C}}_i$,
$\underline{D}_{i+1} = \widetilde{I}_{i+1}\underline{D}_i + \widetilde{I}'_{i+1}\widetilde{\underline{D}}_i$, and $\widetilde{\underline{D}}_{i+1} = \frac{\widetilde{I}_{i+1}\widetilde{I}''_{i+1}}{\widetilde{I}'_{i+1}}\underline{D}_i + \widetilde{I}''_{i+1}\widetilde{\underline{D}}_i$,
2. computes $\kappa_1 = \hat{e}(P_0^{(4)}, \underline{S}_{t_a}) \cdot \hat{e}(P_0^{(3)}, \widetilde{\underline{S}}_{t_a}) \cdot (\hat{e}(L_{t_a}^{(4)}, \rho_{t_a}^{(1)}) \cdot \hat{e}(L_{t_a}^{(3)}, \rho_{t_a}^{(2)})^{-1})^r$;
3. computes $\tau = \hat{e}(U_1, S_{\overline{ID}_{t_b}})\hat{e}(U_2, \widetilde{S}_{\overline{ID}_{t_b}})^{-1}$, $\kappa_2 = \mathcal{H}_2(\tau)$, and $m = E_{\kappa_2}^{-1}(c)$;
4. outputs $(m,$"Valid"$)$ if $\kappa_1 = \hat{e}(V, P_0^{(7)})$ and $r = \mathcal{H}_3(c, \kappa_1)$, and outputs "Invalid" otherwise.

6.4 Efficiency

Schemes	Un+Sign-cryption Cost	Features	
		Hierarchical	PCA
Boy-IDSC [Boy03]	$R_{\mathbb{Z}_q^*} + 5P + 2M_{\mathbb{G}_1} + 1M_{\mathbb{Z}_q^*} + 13H$	N	N
Chow-IDSC [CYHC03]	$R_{\mathbb{Z}_q^*} + 6P + 3M_{\mathbb{G}_1} + 6H$	N	Y
Chow-HIBSC [CYHY04]	$R_{\mathbb{Z}_q^*} + (t_a + t_b + 1)\mathbf{P} + (t_b + 1)M_{\mathbb{G}_1}$ $(t_a + t_b - 3)M_{\mathbb{G}_2} + 2Inv_{\mathbb{G}_2}$ $+(t_a + t_b + 2)H + Ex_{\mathbb{G}_2} + A_{\mathbb{G}_1}$	Y	N
NewHIBSC	$2R_{\mathbb{Z}_q^*} + 9\mathbf{P} + 14M_{\mathbb{G}_1} + 5M_{\mathbb{G}_2} + (26(t_a + t_b) - 40)M_{\mathbb{Z}_q^*}$ $+ 3Inv_{\mathbb{G}_2} + (-4 + 5(t_a + t_b))Inv_{\mathbb{Z}_q^*} + (5 + 3(t_a + t_b))H +$ $+ 3Ex_{\mathbb{G}_2} + (11(t_a + t_b) - 16)A_{\mathbb{Z}_q^*} + 6A_{\mathbb{G}_1}$	Y	Y

Table 6.1: Comparison of NewHIBSC with related IDSC schemes.

Schemes	Un+Sign-cryption Cost					
	$t_a = t_b = 2$	$t_a = t_b = 4$	$t_a = t_b = 5$	$t_a = t_b = 10$	$t_a = t_b = 12$	$t_a = t_b = 20$
Boy-IDSC [Boy03]	462×10^3 †	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
Chow-IDSC [CYHC03]	567×10^3 †	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
Chow-HIBSC [CYHY04]	484×10^3	862×10^3	1.1×10^6	2×10^6	2.4×10^6	3.9×10^6
NewHIBSC	1.1×10^6	1.1×10^6	1.1×10^6	1.1×10^6	1.1×10^6	1.1×10^6

Table 6.2: Timing Estimates of Boy-IDSC, Chow-IDSC, Chow-HIBSC, and NewHIBSC in Microseconds. †This value corresponds to $t_a = t_b = 1$.

Table 6.1 compares the computational requirements of Boyen's ID-based signcryption (IDSC) scheme (Boyen-IDSC [Boy03]), Chow et al.'s efficient IDSC scheme (Chow-IDSC [CYHC03]), Chow et al.'s hierarchical IDSC scheme (Chow-HIBSC [CYHY04]), and our proposed HIBSC scheme (NewHIBSC). (See also Table 6.2 for corresponding timing estimates, based on information obtained from [HPS02].) t_A and t_B respectively denote the hierarchical depth of a signcryptext's sender and intended recipient. All hash functions of the compared schemes are assumed to have the same computational cost, denoted by H. M_X and A_X respectively denote computational costs of scalar multiplication and addition in the Abelian group X. Inv_X denotes the computational cost of inversion in the field X. R_X denotes the computational cost of uniformly selecting a random element in the set X. The computational cost of exponentiation in the group X is denoted by Ex_X, and $\mathbf{P}$ denotes the computational cost of a bilinear pairing operation.

We recall that all the compared schemes are provably forward secure, semantically secure, and existentially unforgeable. The *Hierarchical* column indicates whether each

given scheme handles hierarchically-structured communities, and the *PCA* column indicates whether the scheme features public ciphertext authenticity.

We emphasize that all schemes (except Chow-HIBSC) have constant signcryptext length, and we note that for some applications (e.g. email or IP routing) $t_A, t_B \geq 4$ can be assumed.

Using the empirical computational timings presented in [HPS02], Table 6.1 indicates that NewHIBSC is faster than Chow-HIBSC when $t_a, t_b \geq 5$. In particular, our proposed scheme requires $O(1)$ pairings. When $t_a = t_b = 4$, however, Chow-HIBSC is faster than NewHIBSC. Nevertheless, the proposed HIBSC scheme is the only one which simultaneously handles hierarchies and features public ciphertext authenticity.

Another analysis of interest consits of comparing signcryption schemes with their underlying encryption and signature schemes. Such an analysis shows that NewHIBSC requires the same number of pairings as a sequential execution of its underlying signature and encryption schemes. This extra cost must be paid in order to obtain PCA.

Noteworthy is the fact that the concept of signcryption comes with the assumption that this procedure should preserve the semantics of the *sign-then-encrypt* methodology [Zhe97]. This is due to the fact that the *encrypt-then-sign* methodology allows an attacker to take a given ciphertext and associate it with a signature issued by the attacker. Hence, signcryption with PCA preserves the semantics of the *sign-then-encrypt* methodology, while a sequential application of encryption and signing does not. While encryption followed by signing allows third parties to trace attackers, problems arise when attackers are (or impersonate) trusted parties (e.g. employees of an organization) which, *ipso facto*, are unnoticed at first. To avoid such a possibility (and to save on the resources required to solve the associated misunderstanding), it is generally accepted that ciphertexts and signatures should be "non-detachable" when they should be issued by a single party. This is why signcryption with PCA is generally preferable to the *encrypt-then-sign* methodology. When, however, a signcryption scheme is not more efficient than a sequential execution of the underlying encryption and signing algorithms, this preference can be debated, depending on the application and the resources required to resolve the aforementioned attack. Since our signcryption scheme is *as efficient as* a sequential execution of its underlying algorithms, this debate would not affect the relevance of our scheme. Nevertheless, we leave, for future work, the possibility of improving the efficiency of our signcryption scheme.

6.5 Security

This section presents the security guarantees of our proposed HIBSC scheme. The notions of *semantic security* and *existential unforgeability* for hierarchical ID-based

signcryption schemes (cf. [CYHY04]) are defined in §6.5.1 and §6.5.2. §6.5.3 presents the security guarantees in the form of theorems.

As a preliminary note, we remark that the threat models presented below rely on the assumption that signers and recipients of signcryptexts are different parties. This can be ensured by assigning a sender and a receiver identity to each user of a given signcryption system.

6.5.1 Semantic Security

Let Ψ be a HIBSC scheme. Then, the following game, initiated by a challenger $\mathcal{C}h$ against an attacker $\mathcal{A}$, may be considered:

Setup: From a security parameter k, $\mathcal{C}h$ uses Ψ's *Root Setup* algorithm to generate the cryptosystem's public and private parameters - keeping the private parameters secret while giving the public ones to $\mathcal{A}$.

Phase 1: $\mathcal{A}$ issues to $\mathcal{C}h$ a polynomially bounded number of queries of any of the following types:

- *Public-Key query*: Given an ID-tuple $\overline{ID}_{(i,t_i)} = (ID_{(i,1)}, \cdots, ID_{(i,t_i)})$, $\mathcal{C}h$ must return $H_1(\overline{ID}_{(i,t_i)})$.
- *Key Extraction query*: Given an ID-tuple $\overline{ID}_{(i,t_i)}$ (as above), $\mathcal{C}h$ must return the private key $d_{\overline{ID}_{(i,t_i)}}$ of $\overline{ID}_{(i,t_i)}$.
- *Signcryption query*: Given a message $m_i \in \mathcal{M}$, the ID-tuple $\overline{ID}_{(i,t_i)}$ of a sender, the ID-tuple $\overline{ID}_{(i,t_j)}$ of an intended recipient, the private key $d_{\overline{ID}_{(i,t_i)}}$ of $\overline{ID}_{(i,t_i)}$, and the scheme's public parameters, this algorithm returns a signcryptext $\sigma_i \in \mathcal{C}$ corresponding to m_i.
- *Unsigncryption query*: Given a signcryptext $\sigma_i \in \mathcal{C}$, the ID-tuple $\overline{ID}_{(i,t_i)}$ of a sender, the ID-tuple $\overline{ID}_{(i,t_j)}$ of an intended recipient, the private key $d_{\overline{ID}_{(i,t_j)}}$ of $\overline{ID}_{(i,t_j)}$, and the scheme's public parameters, this algorithm returns either "Invalid" or a pair (m_i,"Valid"), where $m_i \in \mathcal{M}$ is the message associated with σ_i.

The above queries may be issued adaptively: each request may depend on its predecessors.

Challenge: Once *Phase 1* is over, $\mathcal{A}$ issues two ID-tuples $\overline{ID}^*_{t_A}$ and $\overline{ID}^*_{t_B}$ of its choice, and a pair (m_0, m_1) of equal-length plaintexts, such that, in *Phase 1*: (1) no *Key Extraction* queries were issued on $\overline{ID}^*_{t_B}$ or any of its ancestors; and (2) neither m_0 nor m_1 were part of the output of an *Unsigncryption* query issued

with $\overline{ID}^*_{t_B}$ as the intended recipient. $\mathcal{C}h$ then picks a random bit $b \in \{0,1\}$, computes the signcryption $\sigma^* = \text{Signcryption}(m_b, \overline{ID}^*_{t_A}, \overline{ID}^*_{t_B}, d_{\overline{ID}_{t_A}}, params)$ of m_b with $\overline{ID}^*_{t_A}$ as the signer and $\overline{ID}^*_{t_B}$ as the intended recipient. Then, $\mathcal{C}h$ sends σ^* to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ issues a polynomially bounded number of *Phase 1* types of queries, under the following restrictions:

- No *Key Extraction* queries are issued on $\overline{ID}^*_{t_B}$ or any of its ancestors.
- If an *Unsigncryption* query is issued with σ^* as an argument, then the corresponding recipient ID-tuple can be neither $\overline{ID}^*_{t_B}$ nor any of its ancestors.

The queries may be issued adaptively: each request may depend on its predecessors.

Guess: Once *Phase 2* is over, $\mathcal{A}$ submits a guess bit b' and wins the game if $b' = b$.

Definition 6.5.1.1. *The above game is called the* IND-HIBSC-CCA *attack game. The quantity* $|Pr[b' = b] - \frac{1}{2}|$ *– representing the advantage of* $\mathcal{A}$ *over any challenger* $\mathcal{C}h$ *in the game – is denoted by* $Adv^{HIBSC-CCA}_{\mathcal{A},\mathcal{C}h}$. *A HIBSC scheme is said to be secure against adaptive chosen ciphertext attacks (*IND-HIBSC-CCA *secure, in short) if no polynomially bounded attacker can be found that has non-negligible advantage in the above IND-HIBSC-CCA game.*

6.5.2 Existential Unforgeability

Let Ψ be a HIBSC scheme. Then, the following game, initiated by a challenger $\mathcal{C}h$ against an attacker $\mathcal{A}$, may be considered:

Setup: As in the IND-HIBSC-CCA game.

Phase 1: $\mathcal{A}$ issues to $\mathcal{C}h$ a polynomially bounded number of queries, as in the HIBSC-CCA game. These queries may be issued adaptively.

Forgery: $\mathcal{A}$ issues a signcryptext σ^* and the ID-tuple $\overline{ID}^*_{t_B}$ of a recipient, and wins the game if the unsigncryption of σ^* with some sender $\overline{ID}^*_{t_A}$ and recipient $\overline{ID}^*_{t_B}$ yields (m^*,"Valid"), where: (1) $m^* \in \mathcal{M}$; (2) no *key extraction* query was issued on $\overline{ID}^*_{t_A}$; (3) σ^* is not the output of a *signcryption* query issued with m^*, $\overline{ID}^*_{t_A}$ as sender, and $\overline{ID}^*_{t_B}$ as recipient.

Definition 6.5.2.1. *The above game is called the* EU-HIBSC-CMA *attack game. The probability that $\mathcal{A}$ wins the game (i.e. $\mathcal{A}$'s advantage over Ch) is denoted by $Adv_{\mathcal{A},Ch}^{EU-HIBSC-CMA}$. A HIBSC scheme is said to be existentially unforgeable against adaptive chosen message attacks (*EU-HIBSC-CMA *secure, in short) if no polynomially bounded attacker can be found that has non-negligible advantage in the above EU-HIBSC-CMA game.*

6.5.3 Security Theorems

Theorem 6.5.1. *Let* T *be a tree-shaped hierarchy whose depth is logarithmic in a security parameter k. Assume that, all the hash functions of NewHIBSC are random oracles. Suppose also that there exists an attacker $\mathcal{A}$ which has non-negligible advantage $\varepsilon(k)$, in time τ, against any challenger Ch, in the IND-HIBSC-CCA game. Then, there exists an algorithm $\mathcal{B}$ which solves the BDH problem, in time $O(\tau)$, with non-negligible advantage at least*

$$\varepsilon(k)\frac{1}{pol(k)}(1-\delta)^{n_{ext}+1}, \text{where}$$

δ is the largest proportion of ID-tuples having a common root branch, pol is a polynomial such that $\frac{1}{pol(k)}$ is a lower bound of all proportions of ID-tuples having a common root branch, and n_{ext} is the number of Key Extraction *queries made by $\mathcal{A}$ in any IND-HIBSC-CCA (assuming that each* Signcryption *query is preceded*[2] *by a* Key Extraction *on the signer ID-tuple, and that each* Unsigncryption *query is preceded by a* Key Extraction *query on the recipient ID-tuple.)*

Theorem 6.5.2. *Let* T*, k, δ, pol be defined as in Theorem 6.5.1. Let also n_{ext} be the number of* Key Extraction *queries made by $\mathcal{A}$ in any EU-HIBSC-CMA game against $\mathcal{B}$ (assuming that each* Signcryption *query is preceded by a* Key Extraction *on the signer ID-tuple, and that each* Unsigncryption *query is preceded by a* Key Extraction *query on the recipient ID-tuple.) Assume that all the hash functions of NewHIBSC are random oracles. Suppose also that there exists an attacker $\mathcal{A}$ which has non-negligible advantage $\varepsilon(k)$, in time τ, against any challenger Ch, in the EU-HIBSC-CMA game. Then, there exists an algorithm $\mathcal{B}$ which solves the BDH problem, in time $O(\tau)$, with non-negligible advantage at least*

$$\varepsilon(k)\frac{1}{pol(k)}(1-\delta)^{n_{ext}+1}.$$

[2] In practice, a signcryption requires the possession of a signing key and this key must be generated by trusted authority. Hence, the stated assumption is reasonable in practice.

Practical Meaning of Theorem 6.5.2: In practice, this theorem indicates that, when a message is signcrypted for a user, only this user and her hierarchical ancestors are able to unsigncrypt the corresponding signcrypted.

Theorem 6.5.3. *If NewHIBSC is EU-HIBSC-CMA secure and $\mathcal{H}_3$ is a one-way function, then the probability that NewHIBSC does not feature public ciphertext authenticity is negligible.*

Practical Meaning of Theorem 6.5.3: In practice, this theorem indicates that, only a user and her hierarchical ancestors are able to signcrypt a message on behalf of the user.

Theorem 6.5.4. *If NewHIBSC is IND-HIBSC-CCA secure, then NewHIBSC is forward secure.*

Practical Meaning of Theorem 6.5.4: In practice, this theorem indicates that, when a message has been signcrypted, no one (including the signcryptor) can unsigncrypt the corresponding signcryptext. (This obviously excludes cases in which the intended recipient of the signcryptext is the signcryptor herself.)

Proof of Theorem 6.5.1. In this proof, we show how to construct $\mathcal{B}$ using $\mathcal{A}$. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two Abelian groups of prime order q, where $\mathbb{G}_1$ is additive and $\mathbb{G}_2$ is multiplicative. Let $P_0^{(1)} \in \mathbb{G}_1^*$ be a generator of $\mathbb{G}_1$, and ν_2, ν_3, ν_4, ν_5, ν_6, and ν_7 be three positive integers such that $P_0^{(2)} = \nu_2 P_0^{(1)}$, $P_0^{(3)} = \nu_3 P_0^{(1)}$, $P_0^{(4)} = \nu_4 P_0^{(1)}$, $P_0^{(5)} = \nu_5 P_0^{(1)}$, $P_0^{(6)} = \nu_6 P_0^{(1)}$, and $P_0^{(7)} = \nu_7 P_0^{(1)}$ are six other generators of $\mathbb{G}_1$ (i.e. distinct from one another and from $P_0^{(1)}$). Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be a Bilinear pairing on $\mathbb{G}_1$, such that the BDH is assumed to be hard with respect to $\hat{e}$, and let $(\mathbb{G}_1, q, P_0^{(1)}, aP_0^{(1)}, bP_0^{(1)}, cP_0^{(1)})$ be a tuple for which a, b, c are unknown to $\mathcal{B}$. $\mathcal{B}$'s goal is to solve the BDH Problem by computing $\hat{e}(P_0^{(1)}, P_0^{(1)})^{abc}$ in polynomial time. To achieve this goal, $\mathcal{B}$ initiates an *IND-HIBSC-CCA* game with $\mathcal{A}$, using an arbitrary security parameter k.

$\mathcal{B}$ defines three hash functions $\mathcal{H}_1^{sim} : \{0,1\}^* \rightarrow (\mathbb{Z}_q^*)^\ell$, $\mathcal{H}_2^{sim} : \mathbb{G}_2 \rightarrow \{0,1\}^n$, and $\mathcal{H}_5^{sim} : \mathbb{Z}_q^* \rightarrow \mathbb{Z}_q^*$ over which it has full control. $\mathcal{B}$ sets $n = poly_1(k)$, $\ell = \log(k)$, $\xi = poly_2(k)$, $I_0 \in_R \mathbb{Z}_q^*$, $L_1^{(3)} = \nu_3(aP_0^{(1)})$, $L_1^{(4)} = \nu_4(aP_0^{(1)})$, $s_1 \in_R \mathbb{Z}_q^*$, $s_i = \mathcal{H}_5^{sim}(s_{i-1})$ for $2 \leq i \leq \ell - 1$, and $L_i^{(j)} = s_{i-1} L_{i-1}^{(j)}$ for $2 \leq i \leq \ell$ and $j = 3, 4$. $\mathcal{B}$ defines $\mathcal{M} = \{0,1\}^n$, $\mathcal{C} = \mathbb{G}_1^5 \times \{0,1\}^n \times \{0,1\}^n$, $pubParams = (q, n, \hat{e}, I_0, P_0^{(1)}, P_0^{(2)}, P_0^{(3)}, P_0^{(4)}, P_0^{(5)}, P_0^{(6)}, P_0^{(7)}, \mathcal{H}_1^{sim}, \mathcal{H}_2^{sim}, \mathcal{H}_3, \mathcal{H}_5^{sim}, ((L_i^{(j)})_{i=1}^{\ell})_{j=3}^{4})$, and $params = (pubParams, s_0)$, where $s_0 = a$ is unknown to $\mathcal{B}$, and both $\mathcal{H}_4 : \mathbb{Z}_q^* \rightarrow \mathbb{Z}_q^*$ and $\mathcal{H}_3 : \{0,1\}^n \times \mathbb{G}_2 \rightarrow \{0,1\}^n$ are arbitrary one-way hash functions. $\mathcal{B}$ sends $pubParams$ to $\mathcal{A}$, and keeps $(s_i)_{i=1}^{\ell}$ secret.

The rest of the proof consists of three sections. The *Hash Simulation* section shows how $\mathcal{B}$ simulates $\mathcal{H}_1^{sim}$, $\mathcal{H}_2^{sim}$ and $\mathcal{H}_5^{sim}$. The *Attack Games* section explains how $\mathcal{B}$ handles the queries of the attack game. Finally, the *Complexity and Probability* section derives the complexity and probability results of the theorem.

Hash Simulation:

In order to store information about the queries of the *IND-HIBSC-CCA* game, $\mathcal{B}$ maintains a list $\Lambda_{\mathcal{H}_1^{sim}}$ consisting of entries of the form: $(\overline{ID}_{(i,t_i)}, (p_{(i,d,1)})_{d=1}^{t_i}, (p_{(i,d,3)})_{d=1}^{t_i}$, $(I_{(i,d)})_{d=1}^{t_i}, (I'_{(i,d)})_{d=1}^{t_i}, (I''_{(i,d)})_{d=1}^{t_i}, (\widetilde{I}_{(i,d)})_{d=1}^{t_i}, (\widetilde{I}'_{(i,d)})_{d=1}^{t_i}, (\widetilde{I}''_{(i,d)})_{d=1}^{t_i}, (\alpha_{(i,d)})_{d=1}^{t_i}, (\widetilde{\alpha}_{(i,d)})_{d=1}^{t_i}$, $(S_{\overline{ID}_{(i,d)}})_{d=1}^{t_i}, (E_{\overline{ID}_{(i,d)}})_{d=1}^{t_i}, (\widetilde{S}_{\overline{ID}_{(i,d)}})_{d=1}^{t_i}, (\widetilde{E}_{\overline{ID}_{(i,d)}})_{d=1}^{t_i})$, where $\overline{ID}_{(i,t_i)} = (ID_{(i,1)}, \cdots, ID_{(i,t_i)})$ is an ID-tuple of the user hierarchy, and, for $1 \leq d \leq t_i$, $(p_{(i,d,1)})_{d=1}^{t_i}, (p_{(i,d,3)})_{d=1}^{t_i}, I_{(i,d)}$, $I'_{(i,d)}, I''_{(i,d)}, \widetilde{I}_{(i,d)}, \widetilde{I}'_{(i,d)}, \widetilde{I}''_{(i,d)}, \alpha_{(i,d)}, \widetilde{\alpha}_{(i,d)} \in \mathbb{Z}_q^*$, $S_{\overline{ID}_{(i,d)}}, E_{\overline{ID}_{(i,d)}}, \widetilde{S}_{\overline{ID}_{(i,d)}} \in \mathbb{G}_1$, and $\widetilde{E}_{\overline{ID}_{(i,d)}} \in \mathbb{G}_1$ are the simulated values of $\overline{ID}_{(i,d)}$'s key components.

Let $n_{\mathcal{H}_1}$ denote the number of $\mathcal{H}_1$ queries. Every $\mathcal{H}_1$-query is issued on an ID-tuple whose first component we call the *root branch*. Let then n_{rb} denote the number of distinct root branches appearing in $\mathcal{H}_1$-query ID-tuples. Every distinct root branch is indexed using an integer between 1 and n_{rb}. Let μ be a uniformly random element of $\{1, \cdots, n_{rb}\}$. For $\mathcal{B}$ to win the IND-HIBSC-CCA attack game, the index of the ID-tuple associated with the challenge plaintext pair will need to be μ.

To answer $\mathcal{H}_1$ and *Key Extraction* queries, $\mathcal{B}$ builds $\Lambda_{\mathcal{H}_1^{sim}}$ as follows. Let $\overline{ID}_{(i,t_i)} = (ID_{(i,1)}, \cdots, ID_{(i,t_i)})$ be the ID-tuple associated with the i^{th} $\mathcal{H}_1$-query and let y be maximal such that $(ID_{(i,1)}, \cdots, ID_{(i,y)}) = (ID_{(j,1)}, \cdots, ID_{(j,y)})$, where $(ID_{(j,1)}, \cdots, ID_{(j,t_j)})$ is an ID-tuple which is already in $\Lambda_{\mathcal{H}_1^{sim}}$. Then:

- For $1 \leq d \leq y$: $p_{(i,d,1)} = p_{(j,d,1)}$, $p_{(i,d,3)} = p_{(j,d,3)}$, $I_{(i,d)} = I_{(j,d)}$, $I'_{(i,d)} = I'_{(j,d)}$, $I''_{(i,d)} = I''_{(j,d)}$, $\widetilde{I}_{(i,d)} = \widetilde{I}_{(j,d)}$, $\widetilde{I}'_{(i,d)} = \widetilde{I}'_{(j,d)}$, $\widetilde{I}''_{(i,d)} = \widetilde{I}''_{(j,d)}$, $\alpha_{(i,d)} = \alpha_{(j,d)}$, $\widetilde{\alpha}_{(i,d)} = \widetilde{\alpha}_{(j,d)}$, $S_{(i,d)} = S_{(j,d)}$, $E_{(i,d)} = E_{(j,d)}$, $\widetilde{S}_{(i,d)} = \widetilde{S}_{(j,d)}$, and $\widetilde{E}_{(i,d)} = \widetilde{E}_{(j,d)}$;
- For $y < d \leq t_i$:
  1. If $d = 1$:
     - If $\overline{ID}_{(i,t_i)}$ is the first ID-tuple whose root branch index is μ, then $I_{(i,1)}, p_{(i,1,1)}, p_{(i,1,3)} \in_R \mathbb{Z}_q^*$, $\alpha_{(i,1)} = \widetilde{\alpha}_{(i,1)} = 1$, and $S_{(i,1)} = E_{(i,1)} = \widetilde{S}_{(i,1)} = \widetilde{E}_{(i,1)} = Id_{\mathbb{G}_1}$ (where 1 and the neutral element $Id_{\mathbb{G}_1}$ of $\mathbb{G}_1$ are default values). The other components (i.e. $I'_{(i,1)}$, etc) are constructed according to the method described in the *Key Extraction* algorithm.
     - Otherwise, $I_{(i,1)}, p_{(i,1,1)}, p_{(i,1,3)} \alpha_{(i,1)}, \widetilde{\alpha}_{(i,1)} \in_R \mathbb{Z}_q^*$, and the other components (except $S_{(i,1)}$, $E_{(i,1)}$, $\widetilde{S}_{(i,1)}$, and $\widetilde{E}_{(i,1)}$) are computed according to the method described in the *Key Extraction* algorithm. Moreover,

$S_{\overline{ID}_1} = I_{(i,1)}(aP_0^{(1)}) + I'_{(i,1)}\nu_2(aP_0^{(1)}) + \alpha_{\overline{ID}_1}L_1^{(3)}$, $E_{\overline{ID}_1} = \frac{I_{(i,1)}I''_{(i,1)}}{I'_{(i,1)}}(aP_0^{(1)}) + I''_{(i,1)}\nu_2(aP_0^{(1)}) + \widetilde{\alpha}_{\overline{ID}_1}L_1^{(3)}$, and $\widetilde{S}_{\overline{ID}_1} = \alpha_{\overline{ID}_1}\nu_4(aP_0^{(1)}) + \widetilde{I}_{(i,1)}\nu_5(aP_0^{(1)}) + \widetilde{I}'_{(i,1)}\nu_6(aP_0^{(1)})$, $\widetilde{E}_{\overline{ID}_1} = \widetilde{\alpha}_{\overline{ID}_1}\nu_4(aP_0^{(1)}) + \frac{\widetilde{I}_{(i,1)}\widetilde{I}''_{(i,1)}}{\widetilde{I}'_{(i,1)}}\nu_5(aP_0^{(1)}) + \widetilde{I}''_{(i,1)}\nu_6(aP_0^{(1)})$.

2. Otherwise (i.e. if $2 \leq d \leq t_i$), $I_{(i,d)}, p_{(i,d,1)}, p_{(i,d,3)}\alpha_{(i,d)}, \widetilde{\alpha}_{(i,d)} \in_R \mathbb{Z}_q^*$, and the other components (including the I_i's, $S_{(i,d)}$, $E_{(i,d)}$, $\widetilde{S}_{(i,d)}$, and $\widetilde{E}_{(i,d)}$) are computed according to the method described in the *Key Extraction* algorithm.

- $\mathcal{B}$ adds the entry $(\overline{ID}_{(i,t_i)}, (p_{(i,d,1)})_{d=1}^{t_i}, (p_{(i,d,3)})_{d=1}^{t_i}, (I_{(i,d)})_{d=1}^{t_i}, (I'_{(i,d)})_{d=1}^{t_i}, (I''_{(i,d)})_{d=1}^{t_i}, (\widetilde{I}_{(i,d)})_{d=1}^{t_i}, (\widetilde{I}'_{(i,d)})_{d=1}^{t_i}, (\widetilde{I}''_{(i,d)})_{d=1}^{t_i}, (\alpha_{(i,d)})_{d=1}^{t_i}, (\widetilde{\alpha}_{(i,d)})_{d=1}^{t_i}, (S_{\overline{ID}_{(i,d)}})_{d=1}^{t_i}, (E_{\overline{ID}_{(i,d)}})_{d=1}^{t_i}, (\widetilde{S}_{\overline{ID}_{(i,d)}})_{d=1}^{t_i}, (\widetilde{E}_{\overline{ID}_{(i,d)}})_{d=1}^{t_i})$ to the list $\Lambda_{\mathcal{H}_1^{sim}}$.

For $\mathcal{H}_5^{sim}$-queries, $\mathcal{B}$ maintains a list $\Lambda_{\mathcal{H}_5^{sim}}$ whose entries have the form $(\alpha, h_{5,\alpha})$, where $\alpha \in \mathbb{Z}_q^*$ and $h_{2,\alpha} \in \mathbb{Z}_q^*$ is the simulated value of $\mathcal{H}_5(\alpha)$. Thus, on input $\alpha \in \mathbb{Z}_q^*$, $\mathcal{B}$ proceeds as follows:

- If $\Lambda_{\mathcal{H}_5^{sim}}$ already has an entry $(\alpha, h_{5,\alpha})$, then $\mathcal{B}$ returns $h_{5,\alpha}$ as an answer to the $\mathcal{H}_2^{sim}$-query;
- Otherwise, $\mathcal{B}$ computes picks $h_{5,\alpha} \in_R \mathbb{Z}_q^*$, adds $(\alpha, h_{5,\alpha})$ to $\Lambda_{\mathcal{H}_5^{sim}}$, and returns $h_{5,\alpha}$ as an answer to the $\mathcal{H}_5^{sim}$ query.

For $\mathcal{H}_2^{sim}$-queries, $\mathcal{B}$ maintains a list $\Lambda_{\mathcal{H}_2^{sim}}$ whose entries have the form $(\alpha, h_{2,\alpha})$, where $\alpha \in \mathbb{G}_2$ and $h_{2,\alpha} \in \{0,1\}^n$ is the simulated value of $\mathcal{H}_2(\alpha)$. Thus, on input $\alpha \in \mathbb{G}_2$, $\mathcal{B}$ proceeds as follows:

- If $\Lambda_{\mathcal{H}_2^{sim}}$ already has an entry $(\alpha, h_{2,\alpha})$, then $\mathcal{B}$ returns $h_{2,\alpha}$ as an answer to the $\mathcal{H}_2^{sim}$-query;
- If $\alpha = \left(\hat{e}(L_t^{(4)}, \rho_t^{(1)}) \cdot \hat{e}(L_t^{(3)}, \rho_t^{(2)})^{-1}\right)^x$, where x is an output of $\mathcal{H}_5^{sim}$ and $\rho_t^{(1)}, \rho_t^{(2)}$ are associated with a user $\overline{ID}_t$ whose root branch index is μ then, $\mathcal{B}$ proceeds as follows:
  - $\mathcal{B}$ computes $\left(\hat{e}(L_t^{(4)}, \widetilde{\rho}_t^{(1)}) \cdot \hat{e}(L_t^{(3)}, \rho_t^{(2)})^{-1}\right)^x$ where $\widetilde{\rho}_t^{(1)}$ is obtained, for the same user $\overline{ID}_t$, using the following variant of the recursive algorithm meant to compute A_t: $A_1 = bP_0^{(1)} \in \mathbb{G}_1$; $\widetilde{A}_1 = \frac{I''_{(i,1)}}{I'_{(i,1)}}A_1 \in \mathbb{G}_1$; for $1 \leq i < t$, $A_{i+1} = I_{i+1}A_i + I'_{t+1}\widetilde{A}_i$ and $\widetilde{A}_{i+1} = \frac{I_{i+1}I''_{i+1}}{I'_{i+1}}A_i + I''_{i+1}\widetilde{A}_i$.

- $\mathcal{B}$ computes $h_{2,\alpha} = \hbar(\alpha)$ (where $\hbar$ is a random oracle), adds $(\alpha, h_{2,\alpha})$ to $\Lambda_{\mathcal{H}_2^{sim}}$, and returns $h_{2,\alpha}$ as an answer to the $\mathcal{H}_2^{sim}$ query.

Note that $\mathcal{B}$ must compute α for at most all the users whose root branch index is μ. Since the depth of the hierarchy is logarithmic in k, this search is of polynomial complexity.

- Otherwise, $\mathcal{B}$ computes $h_{2,\alpha} = \hbar(\alpha)$, adds $(\alpha, h_{2,\alpha})$ to $\Lambda_{\mathcal{H}_2^{sim}}$, and returns $h_{2,\alpha}$ as an answer to the $\mathcal{H}_2^{sim}$ query.

Attack Game:

- *Phase 1*:
  - By generating or reusing the appropriate entries and values of both $\Lambda_{\mathcal{H}_1^{sim}}$ and $\Lambda_{\mathcal{H}_2^{sim}}$, $\mathcal{B}$ answers the *Public Key* and the *Key Extraction* queries of *Phase 1*, under the following constraint: if the root branch index of the i^{th} ID-tuple of an *Key Extraction* query is μ, then $\mathcal{B}$ terminates the IND-HIBSC-CCA game and fails to solve the BDH problem.
  - *Signcryption* queries are handled as follows. Given a message $m_i \in \mathcal{M}$, the ID-tuple $\overline{ID}_{(i,t_i)}$ of a sender, the ID-tuple $\overline{ID}_{(i,t_j)}$ of an intended recipient, the private key $d_{\overline{ID}_{(i,t_i)}}$ of $\overline{ID}_{(i,t_i)}$, and $pubParams$:
    - If μ is the root branch index of neither $\overline{ID}_{(i,t_i)}$ nor $\overline{ID}_{(i,t_j)}$, then $\mathcal{B}$ runs the *Signcryption* algorithm with the appropriate values of $\Lambda_{\mathcal{H}_1^{sim}}$ and $\Lambda_{\mathcal{H}_2^{sim}}$.
    - If μ is the root branch of $\overline{ID}_{(i,t_i)}$, then $\mathcal{B}$ terminates the IND-HIBSC-CCA game and fails to solve the BDH problem.
    - If μ is the root branch of $\overline{ID}_{(i,t_j)}$ (but not of of $\overline{ID}_{(i,t_i)}$), then $\mathcal{B}$ runs the *Signcryption* algorithm with the appropriate values of $\Lambda_{\mathcal{H}_1^{sim}}$ and $\Lambda_{\mathcal{H}_2^{sim}}$, and (for the encryption part) with the variant of $\overline{ID}_{(i,t_j)}$'s public-key generating algorithm described above in the $\mathcal{H}_1^{sim}$-simulation section.
  - *Unsigncryption* queries are handled as follows: Given a signcryptext $\sigma_i \in \mathcal{C}$, the ID-tuple $\overline{ID}_{(i,t_i)}$ of a sender, the ID-tuple $\overline{ID}_{(i,t_j)}$ of an intended recipient, the private key $d_{\overline{ID}_{(i,t_j)}}$ of $\overline{ID}_{(i,t_j)}$, and the scheme's public parameters:
    - If μ is the root branch index of neither $\overline{ID}_{(i,t_i)}$ nor $\overline{ID}_{(i,t_j)}$, then $\mathcal{B}$ runs the *Unsigncryption* algorithm with the appropriate values of $\Lambda_{\mathcal{H}_1^{sim}}$ and $\Lambda_{\mathcal{H}_2^{sim}}$.

  - If μ is the root branch of $\overline{ID}_{(i,t_i)}$ but not of $\overline{ID}_{(i,t_j)}$, then $\mathcal{B}$ runs the *Unsigncryption* algorithm with the appropriate values of $\Lambda_{\mathcal{H}_1^{sim}}$ and $\Lambda_{\mathcal{H}_2^{sim}}$, and (for the signature verification part) with the variant of $\overline{ID}_{(i,t_i)}$'s public-key generating algorithm described above in the $\mathcal{H}_1^{sim}$-simulation section.
  - If μ is the root branch of $\overline{ID}_{(i,t_j)}$, then $\mathcal{B}$ terminates the IND-HIBSC-CCA game and fails to solve the BDH problem.

- *Challenge Phase*:

  Once *Phase 1* is over, the *Challenge* phase starts. $\mathcal{A}$ issues two ID-tuples $\overline{ID}^*_{t_a}$ and $\overline{ID}^*_{t_b}$ of its choice, and a pair (m_0, m_1) of equal-length plaintexts, such that, in *Phase 1*: (1) no *Key Extraction* queries were issued on $\overline{ID}^*_{t_b}$ or any of its ancestors; and (2) neither m_0 nor m_1 were part of the output of an *Unsigncryption* query issued with $\overline{ID}^*_{t_b}$ as the intended recipient.

  If $\overline{ID}^*_{t_a}$'s root branch index is μ, then $\mathcal{B}$ terminates the IND-HIBSC-CCA game and fails to solve the BDH problem. $\mathcal{B}$ also terminates the IND-HIBSC-CCA game and fails to solve the BDH problem if $\overline{ID}^*_{t_b}$'s root branch index is not μ. Consequently, it is henceforth assumed that $\overline{ID}^*_{t_a}$'s root branch index is not μ, and that $\overline{ID}^*_{t_b}$'s root branch index is μ.

  $\mathcal{B}$ picks a random bit $b \in_R \{0,1\}$ and sends to $\mathcal{A}$ the signcryptext $\sigma^* = (U_1, U_2, V, \underline{S}_{t_a}, \widetilde{S}_{t_a}, c^*, r^*)$, where $U_1 = \nu_4(cP_0^{(1)})$, $U_2 = \nu_3(cP_0^{(1)})$, $V = y(P_0^{(4)} - P_0^{(3)})$ (with $y \in_R \mathbb{Z}_q^*$), $\kappa_1 = \hat{e}(P_0^{(4)} - P_0^{(3)}, P_0^{(7)})^y$, $\kappa_2 \in_R \{0,1\}^n$, $c^* = E_{\kappa_2}(m_b)$, $r^* = \mathcal{H}_3(c, \kappa_1)$, $\underline{S}_{t_a} = yP_0^{(7)} - r^* S_{t_a}$, and $\widetilde{\underline{S}}_{t_a} = -yP_0^{(7)} + r^* \widetilde{S}_{t_a}$.

  Note that $S_{\overline{ID}^*_{t_b}} = s_0 I_1 (\frac{A_{t_b}}{I_1} \prod_{i=2}^{t_b} s_{i-1}) P_0^{(1)} + (\prod_{i=2}^{t_b} s_{i-1}) B_{t_b}(s_0 P_0^{(2)}) + (\prod_{i=2}^{t_b} s_{i-1}) F_{t_b}(s_0 P_0^{(3)})$. Note also that $I_1 = b$, and that $\frac{A_{t_b}}{I_1}$, B_{t_b}, and F_{t_b} can be computed by $\mathcal{B}$ because they are multivariate polynomials in $I_i, I_i', I_i'', \widetilde{I}_i, \widetilde{I}_i', \widetilde{I}_i''$ for $i = 2, \cdots, t_b$ (i.e. not in I_1 nor in I_1^{-1}). See that $\hat{e}(\nu_4 cP_0^{(1)}, (\prod_{i=2}^{t} s_{i-1}) F_{t_b}(s_0 P_0^{(3)})) \cdot \hat{e}(\nu_5 cP_0^{(1)}, (\prod_{i=2}^{t} s_{i-1}) C_{t_b}(s_0 P_0^{(3)})) \cdot \hat{e}(\nu_6 cP_0^{(1)}, (\prod_{i=2}^{t} s_{i-1}) D_{t_b}(s_0 P_0^{(3)})) = \hat{e}(U_2, \widetilde{S}_{\overline{ID}_t^*})$.

  Let $\Delta_1 = (\frac{A_{t_b}}{I_1} \prod_{i=2}^{t_b} s_{i-1})$, $\Delta_2 = (\prod_{i=2}^{t_b} s_{i-1}) B_{t_b}$, $\Delta_3 = (\prod_{i=2}^{t} s_{i-1}) C_{t_b}$, $\Delta_4 = (\prod_{i=2}^{t_b} s_{i-1}) C_{t_b}$, and $\Delta_5 = (\prod_{i=2}^{t_b} s_{i-1}) D_{t_b}$.

  Then, $\hat{e}(P_0^{(1)}, P_0^{(1)})^{abc} = \left((\hat{e}(U_1, S_{\overline{ID}^*_{t_b}}) \cdot \hat{e}(U_2, \widetilde{S}_{\overline{ID}^*_{t_b}})^{-1}) \cdot \Gamma^{-1} \right)^{(\nu_4 \Delta_1)^{-1}}$, where $\Gamma = \hat{e}(\nu_4 cP_0^{(1)}, s_0 P_0^{(2)})^{\Delta_2} \cdot \hat{e}(\nu_5 cP_0^{(1)}, s_0 P_0^{(3)})^{-\Delta_4} \cdot \hat{e}(\nu_6 cP_0^{(1)}, s_0 P_0^{(3)})^{-\Delta_5}$.

- *Phase 2*:

According to the rules of the IND-HIB-CCA game, $\mathcal{A}$ then makes a second series of queries, which $\mathcal{B}$ answers, *as in Phase 1*, under the restrictions of the second phase of the IND-HIBSC-CCA game.

- *Guess Phase*:

  Once the second phase of the IND-HIBSC-CCA game is over, $\mathcal{A}$ makes a guess $\beta' \in \{0,1\}$ and sends it to $\mathcal{B}$. $\mathcal{B}$ ignores β', uniformly picks a random entry $(\alpha, h_{2,\alpha})$ of $\Lambda_{\mathcal{H}_2^{sim}}$, and submits

  $$\left(\alpha \cdot \Gamma^{-1}\right)^{(\nu_4 \Delta_1)^{-1}}$$

  as the solution of the BDH problem.

  Note that $\mathcal{A}$ must compute $\hat{e}(U, S_{\overline{ID}_t^*})\hat{e}(\widetilde{U}, \widetilde{S}_{\overline{ID}_t^*})^{-1}$ in order to conclude that c^* is an invalid ciphertext. When this happens, then it suffices that $\mathcal{B}$ picks the right entry of $\Lambda_{\mathcal{H}_2^{sim}}$, in order to solve the BDH. Moreover, if $\mathcal{A}$ does not find out that c^* is invalid, then $\mathcal{B}$ does not solve the BDH. Consequently, $\mathcal{B}$ solves the BDH if and only if the following conditions are satisfied: (1) In both *Phase 1* and *Phase 2*, no *Key Extraction* is issued on an ID-tuple whose root branch index is μ (assuming that: (1a) each *Signcryption* query is preceded by a *Key Extraction* on the signer ID-tuple; and (1b) each *Unsigncryption* query is preceded by a *Key Extraction* query on the recipient ID-tuple); (2) $\mathcal{A}$ asks to be challenged on a signer-recipient ID-tuple pair $(\overline{ID}_{t_a}^*, \overline{ID}_{t_b}^*)$ such that the root branch index of $\overline{ID}_{t_a}^*$ is not μ, but the root branch index of $\overline{ID}_{t_b}^*$ is μ; (3) $\mathcal{A}$ discovers that c^* is an invalid ciphertext; (4) $\mathcal{B}$ picks the right element of $\Lambda_{\mathcal{H}_2^{sim}}$.

Complexity and Probability:
In order to compute the probability that $\mathcal{B}$ solves the BDH problem, let δ_μ be the proportion of ID-tuples whose root branch index is μ (among all possible ID-tuples). Assume that, in the IND-HIBSC-CCA game, $\mathcal{A}$ issues a total of n_{ext} *Key Extraction* queries. If $\varepsilon(k)$ is the advantage of $\mathcal{A}$ in the IND-HIBSC-CCA game, then the probability that $\mathcal{B}$ solves the BDH problem is at least $\varepsilon(k)(1-\delta_\mu)\delta_\mu(1-\delta_\mu)^{n_{ext}}$. Now, $\delta_\mu \leq \delta$, since δ is the largest proportion of ID-tuples having a common root branch. Moreover, since all proportions of ID-tuples having a common root branch are assumed to be non-negligible quantities, there exists a polynomial pol over $\mathbb{N}$ such that $\delta_\mu \geq \frac{1}{pol(k)}$. Hence, $\varepsilon(k)\delta_\mu(1-\delta_\mu)^{n_{ext}+1} \geq \varepsilon(k)\frac{1}{pol(k)}(1-\delta)^{n_{ext}+1}$. Now, since $\varepsilon(k)$ is non-negligible, so is $\varepsilon(k)\frac{1}{pol(k)}(1-\delta)^{n_{ext}+1}$. Consequently, $\mathcal{B}$ solves the BDH problem with non-negligible probability at least $\varepsilon(k)\frac{1}{pol(k)}(1-\delta)^{n_{ext}+1}$. Moreover, since each query of the IND-HIBSC-CCA game requires $\mathcal{B}$ to make a polynomial number of

operations, it follows that $\mathcal{B}$ solves the BDH problem in time $O(\tau)$ where τ is the running time of $\mathcal{A}$ in the IND-HIBSC-CCA attack game. **QED**.

Proof of Theorem 6.5.2: The setup (i.e. definition of parameters, of lists, and simulation of hash functions) of this proof is done exactly as in the proof of Theorem 6.5.1, except that $\mathcal{B}$ challenges $\mathcal{A}$ in a EU-HIBSC-CMA game (instead of a IND-HIBSC-CCA game).

Phase 1 of the EU-HIBSC-CMA game is handled by $\mathcal{B}$ as in the proof of Theorem 6.5.1. Let then $\sigma^* = (U_1, U_2, V, \underline{S}_{t_a}, \widetilde{S}_{t_a}, c^*, r^*)$ be the forged signcryptext issued by $\mathcal{A}$ in the *Forgery* phase. If the unsigncryption of σ^* returns (m^*, "Valid") under the signer $\overline{ID}^*_{t_a}$ and the recipient $\overline{ID}^*_{t_b}$, then $\underline{S}_{t_a} = Y - r^* S_{t_a}$ and $\widetilde{S}_{t_a} = -Y + r^* \widetilde{S}_{t_a}$, for some $Y \in \mathbb{G}_1$ such that $\kappa_1 = \hat{e}(V, P_0^{(7)}) = \hat{e}(P_0^{(4)} - P_0^{(3)}, Y)$ (where $\mathcal{H}_3(c^*, \kappa_1) = r^*$.)

If the root branch index of $\overline{ID}^*_{t_a}$ is not μ, then $\mathcal{B}$ terminates the EU-HIBSC-CMA game and fails to solve the BDH problem. It is therefore henceforth assumed that the root branch index of $\overline{ID}^*_{t_a}$ is μ.

Thus, since the remarks made in the proof Theorem 6.5.1 with respect to $S_{\overline{ID}^*_{t_b}}$, $E_{\overline{ID}^*_{t_b}}$, and $\widetilde{S}_{\overline{ID}^*_{t_b}}$ are valid for $S_{\overline{ID}^*_{t_a}}$, $E_{\overline{ID}^*_{t_a}}$, and $\widetilde{S}_{\overline{ID}^*_{t_a}}$, then the following is true when σ^* is valid with $\overline{ID}^*_{t_a}$ as signer:

$$\hat{e}(P_0^{(1)}, P_0^{(1)})^{abc} = \left((\hat{e}(\nu_4(cP_0^{(1)}), S_{\overline{ID}^*_{t_a}}) \cdot \hat{e}(\nu_3(cP_0^{(1)}), \widetilde{S}_{\overline{ID}^*_{t_a}})^{-1}) \cdot \Gamma^{-1} \right)^{(\nu_4 \Delta_1)^{-1}},$$

where $\Gamma = \hat{e}(\nu_4 cP_0^{(1)}, s_0 P_0^{(2)})^{\Delta_2} \cdot \hat{e}(\nu_5 cP_0^{(1)}, s_0 P_0^{(3)})^{-\Delta_4} \cdot \hat{e}(\nu_6 cP_0^{(1)}, s_0 P_0^{(3)})^{-\Delta_5}$.

Consequently, $\mathcal{B}$ computes

$$Sol = \hat{e}(V, \nu_7(cP_0^{(7)}))^{-1} \cdot \hat{e}(\nu_4(cP_0^{(1)}), \underline{S}_{t_a}) \hat{e}(\nu_4(cP_0^{(1)}), \widetilde{S}_{t_a}) \cdot \Gamma^{-1},$$

and submits Sol as the solution of the BDH problem.

$\mathcal{B}$ solves the BDH if $\mathcal{A}$ issues a valid signcryptext. Consequently, $\mathcal{B}$ solves the BDH if the following conditions are satisfied: (1) In *Phase 1*, no *Key Extraction* is issued on an ID-tuple whose root branch index is μ (assuming that: (1a) each *Signcryption* query is preceded by a *Key Extraction* on the signer ID-tuple; and (1b) each *Unsigncryption* query is preceded by a *Key Extraction* query on the recipient ID-tuple); (2) $\mathcal{A}$ submits a signcryptext whose unsigncryption is valid with a signer-recipient ID-tuple pair $(\overline{ID}^*_{t_a}, \overline{ID}^*_{t_b})$ such that the root branch index of $\overline{ID}^*_{t_a}$ is μ, but the root branch index of $\overline{ID}^*_{t_b}$ is not μ.

The *Complexity and Success Probability* analysis of $\mathcal{B}$'s ability to solve the BDH problem is done according to the argument present in the proof of Theorem 6.5.1. The only change in the argument is that the root branch of $\overline{ID}^*_{t_a}$ must be μ while that root branch index of $\overline{ID}^*_{t_b}$ must be different from μ. The resulting success probability is the same, and so is the complexity of $\mathcal{B}$'s work. **QED**.

Proof of Theorem 6.5.3. Assume that NewHIBSC is EU-HIBSC-CMA secure. Then, the *Unsigcryption* algorithm guarantees that the probability of obtaining the same value of κ_1 from two different public keys (and hence signer ID-tuples) is negligible. Moreover, the fact that $\mathcal{H}_3$ is a one-way function guarantees that the probability of obtaining the same value of r with two distinct value of c and the same value of κ_1 is negligible. Furthermore, the fact that $\mathcal{H}_3$ is a one-way function guarantees that the probability of obtaining r from the same value of c and two different public keys (and hence signer ID-tuples) is negligible. Thus, by addition of the two preceding probabilities, it follows that, since r and c are public/fixed (given any signcryptext), the probability that NewHIBSC does not feature public ciphertext authenticity is negligible. **QED**.

Proof of Theorem 6.5.4. Assume that NewHIBSC is IND-HIBSC-CCA secure. Then, the compromise of a signer's private key does not compromise the confidentiality of a signcrypted message (as long as the signer's key is different from the intended recipient of this signcrypted message). Hence, NewHIBSC is forward secure. **QED**.

6.6 Conclusion

The main goal of this chapter was to describe an efficient hierarchical ID-based signcryption scheme providing both public ciphertext authenticity and forward security. PCA enables third parties (such as firewalls) to route or stop signcrypted messages depending on the origin and integrity of the signcryptexts. FS prevents the theft of a user's signing key from compromising the confidentiality of signcryptexts issued by this user.

To achieve the aforementioned goal, NewHIBE and NewHIBS were efficiently combined with mechanisms enabling PCA and FS. Moreover, the proposed HIBSC scheme was proved to be both semantically secure with respect to adaptive chosen ciphertext attacks and existentially unforgeable with respect to adaptive chosen message attacks (in the random oracle model.)

An open question is whether one can design a hierarchical ID-based signcryption scheme featuring ciphertext anonymity (i.e. the property whereby no information concerning both the sender and the recipient of any given signcryptext is revealed to an active third party.)

In the next chapter, an efficient hierarchical ID-based group signature scheme is presented.

Chapter 7

Hierarchical ID-Based Group Signature

The third main contribution of this dissertation deals with the issue of hierarchical anonymous authentication (HAA). HAA can be formulated as follows. Suppose that a large user group $\mathcal{G}$ consists of publicly-identifiable, non-overlapping, and hierarchically-structured subgroups. Then, one seeks to determine whether it is possible to identify the subgroup of any member of $\mathcal{G}$, while keeping the identity of each member secret from all but a designated party of this member's subgroup. (This designated party will be called the *subgroup leader*.)

In order to solve the above problem, in the case of large user groups (e.g. 10,000-user groups), this chapter describes a hierarchical ID-based group signature (HIBGS) scheme featuring a property called PSI (public subgroup identification).

§7.1 reviews the relevance of the aforementioned problem, and provides an overview of this chapter's contributions. §7.2 presents fundamental terminology concerning HIBGS schemes. §7.3 describes our proposed HIBGS scheme, and §7.4 discusses its computational and space requirements in comparison with related schemes. §7.5 summarizes the security guarantees of our scheme, and §7.6 concludes the chapter.

7.1 Introduction

Motivation

Application 1. Suppose that a company $\mathcal{O}$ wants to achieve the following goals: (G1) $\mathcal{O}$ wants to reveal only part of its network map to third parties; (G2) $\mathcal{O}$ wishes to stop unwanted traffic passing through its network; and, (G3) when legitimate network traffic is intended for hidden parts of its network, $\mathcal{O}$ wants to route this

traffic to the right destination nodes. If it achieves these three goals, then $\mathcal{O}$ is able to perform traffic filtering and routing in a partially-hidden network. Moreover, the fact that $\mathcal{O}$'s network is partially hidden would increase $\mathcal{O}$'s resistance against various network-based attacks (e.g. intrusion and denial-of-service attacks) which stem from network mapping attacks. Remark that $\mathcal{O}$ could organize its network in hierarchically-structured subnets whose nodes are kept hidden from all but a designated router of the each subnet. Then, using a solution to the HAA problem, $\mathcal{O}$ could use each such router to direct network messages destined to members of its subnet. Using a solution to the HAA problem, $\mathcal{O}$ could also determine whether any network message is associated with a known subnet of $\mathcal{O}$'s network. Hence, any solution to the HAA problem can used to achieve G1 through G3.

Application 2. Another instance of the HAA problem surfaces in the context of *privacy-enhanced access control* (cf. 1). For instance, a hotel could provide its clients with privacy-enhancing keys which would enable these clients to access selected (and hierarchically-structured) sections of the hotel facility, in an anonymous fashion. Then, in case of law-mandated investigation, the anonymity of this hotel's clients could be revoked by a designated party. Furthermore, we note that the same concept could be applied to a company which offers payable online private services (cf. Chapter 1).

ID-based HGS. Recall that group signature schemes are the cryptographic primitive which have been specifically designed to support authentication with revocable anonymity. Recall also that hierarchical ID-based cryptographic scheme are suitable for large and distributed user communities. Hence, the aforementioned applications lead us to attempt to build a hierarchical ID-based group signature scheme which supports HAA. Moreover, since the property of hierarchical group signature (HGS) schemes which supports HAA is called *public subgroup identification* (PSI), we seek to devise an ID-based HGS scheme with PSI.

Contributions

The contributions of this chapter are twofold. First, we formalize the security of *HGS* schemes with PSI. Second, we construct a provably secure *HGS* scheme offering PSI.

We propose a hierarchical ID-based group signature (HIBGS) scheme, with the following group signature properties.

- First, the scheme allows each signature verifier to use a locally kept list of revoked-user tokens, in order to determine the revocation status of signers, at signature verification time. (Group signatures having this feature are said have *Verifier-Local Revocation* (VLR) capability [BS04].)

- Second, the scheme enables subgroup leaders to co-generate their subgroup members' keys, but prevents subgroup leaders from being able to sign on behalf of their subgroup members. This stands in contrast with Boneh and Shacham's recent group signature scheme [BS04].

- Third, the proposed scheme provides a mechanism for signers to be able to reuse the same private key to generate multiple group signatures (unlike Chen et al.'s ID-based GS scheme [CZK03].)

- Fourth, the scheme describes a signature verification procedure whose computational cost is only logarithmic with respect to the number $|SGL|$ of subgroups in a given hierarchy. This improves Boneh and Franklin's anonymous authentication scheme with subset queries [BF99], whose signing and signature verification costs grow linearly with respect to $|SGL|$.

- Fifth, the suggested scheme is provably secure in the the random oracle model [BR93], assuming the intractability of the bilinear Diffie-Hellman problem.

- Sixth, the proposed scheme enables signature verifiers to assess the validity of group signatures, without the need to obtain certified copies of group public keys. This stands in contrast to all certificate-based group signature schemes, including [BF99] and [BS04].

Additionally, our proposed scheme has the following features, as a *hierarchical* GS scheme.

- First, it prevents the ancestors of any subgroup leader *sgl* from impersonating *sgl* in her interactions with existing members of *sgm*'s subgroup. This is achieved by requiring each subgroup leader to run confidential and authenticated communications with her subgroup members, based on a secret which is unique to the pair (subgroup member, subgroup leader).

- Second, it is a *HGS* scheme with PSI, and thereby supports hierarchical anonymous authentication.

- Third, it is scalable, in the sense that it enables subgroup leaders to generate (on their own) their descendants' private keys. This stands in contrast with Kim et al.'s multigroup scheme [KPW97], which requires both subgroup leaders and signers to register with a central trusted party.

- Fourth, it is efficient, featuring constant-length keys and a constant-time signing procedure.

Noteworthy is the fact that our HIBGS scheme features constant-size signatures (even in the hierarchical setting). This property was obtained from the key generation algorithm of NewHIBE. However, let us underscore two steps and two conditions required to obtain our proposed HIBGS scheme, from this HIBE scheme.

- Leader (i.e. subgroup leader) keys had to be modified in such a way that they are different from subgroup member's signing keys. This is due to the fact that signing keys should be co-generated with subgroup members (so that leaders are not able to sign on behalf of subgroup members), while subgroup leaders should be able to generate the full key of each one of their children (i.e. sub-leaders). This difference between sub-leader keys and signer keys also implies that a new algorithm had to be designed for the generation of signing keys. Avoiding member impersonation by key-issuing leaders is not a trivial task: one must describe a mechanism which guarantees that any given signature was issued with a subgroup member's secret, without revealing this secret, yet showing that the secret is bound to the key given by the subgroup leader to the signer.

- The user key generation procedure had to be redesigned in such a way that one signing key enables its intended user to issue multiple group signatures. Note that this feature cannot be achieved using Chen et al's [CZK03] approach, since that approach requires each user to obtain a signing key for each issued group signature. Our approach is different, as shown in §7.3.

- The new key generation algorithm needed to be modified in such a way that it remains provably secure, in the random oracle model.

- Keys of the resulting scheme also had to remain of constant length, and signatures had to remain of constant size.

For our proposed scheme, it was also necessary to devise a threat model for *HGS* schemes with PSI, and to prove the security of our proposed scheme with respect to this model. To devise this model, we extended Boneh and Schacham's security model for group signature schemes with VLR capability in such a way that attempts of leaders to sign on behalf of their subgroup members are taken into consideration.

7.2 Preliminaries

In this section, we present fundamental definitions concerning HIBGS schemes.

ID-tuples

For the description of HIBGS schemes, subgroup leaders are assumed to be organized in a tree-shaped hierarchy whose root is called the root leader and is denoted by both *rLead* and $\overline{ID}_0$. Apart from *rLead*, every subgroup leader is identified with a tuple $\overline{ID}_t = (ID_1, ID_2, \cdots, ID_t)$ (called *ID-tuple*). Under such a notation, $\overline{ID}_t$'s ancestors consists of *rLead* and the $\overline{ID}_i$'s such that $1 \leq i < t$.

Hierarchical ID-based Group Signature Scheme

Each hierarchical ID-based group signature (HIBGS) scheme is composed of six algorithms. The function of each of these is described below.

1. `Root Setup` (k): Given a security parameter k, this algorithm is used by the root leader to return a tuple *params* of system parameters. *params* includes a description of the message space $\mathcal{M}$ and of the signature space $\mathcal{S}$, along with a secret piece of data $d_{\overline{ID}_0}$ called the root leader's private key. Other parameters are allowed, *provided they are not unique to any user.* Some parameters may be public (including those describing $\mathcal{M}$ and $\mathcal{S}$), while others remain secret (including the root secret key).

2. `Sub-Leader-Key Generation` (ID_{t+1}, $d_{\overline{ID}_t}$): Given the scheme's public parameters, an arbitrary identifier ID_{t+1}, and the private key $d_{\overline{ID}_t}$ of a leader identified by $\overline{ID}_t$, this algorithm computes $\overline{ID}_{t+1}$'s private key $d_{\overline{ID}_{t+1}}$.

3. `User-Key Generation` ($n_{\overline{ID}_t}$, $mt_{\overline{ID}_t}$): Given the number $n_{\overline{ID}_t}$ of members belonging to the leader $\overline{ID}_t$'s subgroup, and given an $n_{\overline{ID}_t}$-long vector $mt_{\overline{ID}_t} = \{mt_{\overline{ID}_t[1]}, mt_{\overline{ID}_t[2]}, \cdots, mt_{\overline{ID}_t[n_{\overline{ID}_t}]}\}$ of user membership tokens, this algorithm returns an $n_{\overline{ID}_t}$-long vector of certified user membership tokens $cmt_{\overline{ID}_t} = \{cmt_{\overline{ID}_t[1]}, cmt_{\overline{ID}_t[2]}, \cdots, cmt_{\overline{ID}_t[n_{\overline{ID}_t}]}\}$, an $n_{\overline{ID}_t}$-long vector of user secret keys $sk_{\overline{ID}_t} = \{sk_{\overline{ID}_t[1]}, sk_{\overline{ID}_t[2]}, \cdots, sk_{\overline{ID}_t[n_{\overline{ID}_t}]}\}$, and an $n_{\overline{ID}_t}$-long vector of user revocation tokens $rt_{\overline{ID}_0} = \{rt_{\overline{ID}_0[1]}, rt_{\overline{ID}_0[2]}, \cdots, rt_{\overline{ID}_0[n_{\overline{ID}_0}]}\}$.

4. `Signing` (m, $d_{\overline{ID}_t[i]}$): Given a message $m \in \mathcal{M}$, the private key $d_{\overline{ID}_t[i]}$ associated with a signer $\overline{ID}_t[i]$ (where $1 \leq i \leq n_{\overline{ID}_t}$), and the scheme's public parameters, this algorithm returns a signature $\sigma \in \mathcal{S}$ corresponding to m.

5. `Verifying` ($RL_{\overline{ID}_t}$, m, σ): Given a vector $RL_{\overline{ID}_t}$ of revocation tokens associated with a claimed subgroup leader $\overline{ID}_t$, a message $m \in \mathcal{M}$, a signature $\sigma \in \mathcal{S}$, and the scheme's public parameters, this algorithm returns "valid" or "invalid".

6. `Opening` ($RL_{\overline{ID}_t}$, m, σ): Given a signature $\sigma \in \mathcal{S}$ of a message $m \in \mathcal{M}$ which is valid with respect to both $\overline{ID}_t$ and $RL_{\overline{ID}_t}$, and given the scheme's public parameters, this algorithm returns either "Fail" or the identifier $\overline{ID}_t[i]$ of σ's issuer.

7. `Arbitrating` ($RL_{\overline{ID}_t}$, m, σ, $rt_{\overline{ID}_t}$, $cmt_{\overline{ID}_t[i]}$): Given a signature σ of a message $m \in \mathcal{M}$ such that σ is valid with respect to both $\overline{ID}_t$ and $RL_{\overline{ID}_t}$, and such that a valid subgroup member $\overline{ID}_t[i]$ refutes $\overline{ID}_t$'s claim that $\overline{ID}_t[i]$ issued σ, this algorithm uses both the user revocation vector $rt_{\overline{ID}_t}$ of $\overline{ID}_t$ and the certified membership token $cmt_{\overline{ID}_t[i]}$ of $\overline{ID}_t[i]$ to return either "$\overline{ID}_t[i]$ issued σ" or "$\overline{ID}_t[i]$ did not issue σ".

Double Discrete Logarithm

Let $\mathbb{G}_1$ be an additive Abelian group of prime order q. Let $P_0^{(1)} \in \mathbb{G}_1^*$ be a generator of $\mathbb{G}_1$. Let $\mathcal{A}$ be an attacker modelled as a probabilistic Turing machine.

The *Double Discrete Logarithm* (*DDL*) problem [CS97] is that in which $\mathcal{A}$ is to compute an unknown $c \in \mathbb{Z}_q^*$ given both $(a^c P_0^{(1)}, a, P_0^{(1)})$. The success (or *advantage*) of $\mathcal{A}$ in the *DDL* problem is then defined as $Pr[\mathcal{A}$ finds $c]$. The intractability of the *DDL* problem is assumed to prove the selfless full anonymity of our proposed HIBGS scheme (cf §7.5.) The DDL problem was used to prove the security of zero-knowledge proof of knowledge protocols [FFS87]. Such protocols were used to construct both Camenisch and Stadler's group signature scheme [CS97] and Lysyanskaya and Ramzan's blind group digital signature scheme [LR98]. The DDL problem was also used to construct Kula's encryption scheme [Kul]. Recently, Konama et al. [KMS05] proved that the DDL problem is as at least as hard as the discrete logarithm (DL) problem modulo a prime.

7.3 Proposed HIBGS Scheme

Since the design of a HIBGS scheme is complex, §7.3.1 presents an overview of our HIBGS scheme, while the actual scheme is described in §7.3.2.

7.3.1 Overview

Principals. The following parties are involved in our scheme: one root leader (*rLead*), one set of $|SGL|$ subgroup leaders, $|SGL|$ sets of subgroup members, one set of signature verifiers (i.e. third parties who can verify HGS signatures), and

one arbitrator (i.e. a trusted third party). The subgroups are organized in a tree-shaped hierarchy, and each subgroup is identified by the ID-tuple of its subgroup leader. For instance, any second-level subgroup is identified by an ID-tuple of the form $\overline{ID}_2 = (ID_1, ID_2)$, where $\overline{ID}_1 = (ID_1)$ refers to the ID-tuple of a first-level subgroup leader. Moreover, each member of a subgroup is identified by an indexed ID-tuple whose prefix is the ID-tuple of the subgroup. For example, the i^{th} member of the $\overline{ID}_2$ is identified by $\overline{ID}_2[i]$.

Key Generation. The proposed HIBGS scheme has three key generation algorithms. These are the *Root Setup* algorithm, the *Leader-Key Generation* algorithm, the *User-Key Generation* algorithm. ◇ The *Root Setup* algorithm defines various secret and public parameters which are used by others methods of the scheme. ◇The *Leader-Key Generation* algorithm is used by each subgroup leader to generate the private key of her children subgroups (i.e. the subgroups placed directly below the leader's subgroup). The private key of each t^{th}-level subgroup leader is a tuple $(S_{\overline{ID}_t}, E_{\overline{ID}_t}, \widetilde{S}_{\overline{ID}_t}, R_{\overline{ID}_{t+1}}, \widetilde{R}_{\overline{ID}_{t+1}}, s_t)$, where the first five entries are used to issue signing keys and the last entry is unique to the t^{th} hierarchical level. Moreover, the first five entries of a subgroup leader's private key are generated with random elements $\alpha_{\overline{ID}_t}$ and $\widetilde{\alpha}_{\overline{ID}_t}$ which make the private key unique to the branch and level of the corresponding subgroup. Furthermore, the first five entries of the subgroup leader's private key are generated in such a way that: none can be derived from the others; none can be derived from the private key of a subgroup leader of the same level; and none can be derived from the private key of a subgroup leader or a subgroup member located below the current subgroup leader. These conditions imply that subgroup leaders located on different branches of a subgroup hierarchy cannot derive one another's private keys. ◇ The third key generation algorithm of the proposed HIBGS scheme is the *User-Key Generation* algorithm. This procedure is used by each subgroup member (say $\overline{ID}_t[i]$) and her subgroup leader to co-generate $\overline{ID}_t[i]$'s private key. This process is initiated by $\overline{ID}_t[i]$ who generates a key $mgt_{\overline{ID}_t[i]}$ and hides it from $\overline{ID}_t$ by raising a known basis to the exponent $mgt_{\overline{ID}_t[i]}$. This hiding process yields a membership token $mt_{\overline{ID}_t[i]}$ which $\overline{ID}_t[i]$ gives to $\overline{ID}_t$. $\overline{ID}_t$ then verifies (in zero knowledge) that $\overline{ID}_t[i]$ knows $mgt_{\overline{ID}_t[i]}$, and computes both a revocation token and $\overline{ID}_t[i]$'s private key. The revocation token $rt_{\overline{ID}_t[i]}$ is given the value of $mt_{\overline{ID}_t[i]}$, but $rt_{\overline{ID}_t[i]}$ is not sufficient to identify $\overline{ID}_t[i]$. $\overline{ID}_t[i]$'s private key has the form $(S_{\overline{ID}_t[i]}, E_{\overline{ID}_t[i]}, \widetilde{S}_{\overline{ID}_t[i]}, s_{t+1}, \gamma_{\overline{ID}_t[i]})$, where the first four entries have the same properties as $\overline{ID}_t$'s private key entries, and where the last entry must be used by $\overline{ID}_t[i]$ when she issues group signatures.

Signing. The signing algorithm is composed of three steps. The first step consists of using the signer's hidden key $mgt_{\overline{ID}_t[i]}$ in order to obtain a new version of $S_{\overline{ID}_t[i]}$,

$E_{\overline{ID}_t[i]}$, and $\widetilde{S}_{\overline{ID}_t[i]}$ (namely $\underline{S}_{\overline{ID}_t[i]}$, $\underline{E}_{\overline{ID}_t[i]}$, and $\underline{\widetilde{S}}_{\overline{ID}_t[i]}$). These newly computed values are bound to $mgt_{\overline{ID}_t[i]}$. The second step of the signature algorithm consists of using $\gamma_{\overline{ID}_t}$ to set up a zero knowledge proof that $\overline{ID}_t[i]$ knows $\gamma_{\overline{ID}_t[i]}$. This proof is randomized (using the variable r) in such a way that third parties cannot determine whether two group signatures are issued by the same user. The third step of the signing algorithm mimics the *Leader-Key Generation* procedure, but uses a one-way function of the signed message in order to generate values which demonstrate that the signer has the required private key.

Signature Verification. The signature verification algorithm proceeds in two steps. In the first step, the verifiers computes the public key of the claimed signer (i.e. $A_{t+1}P_0^{(1)} + B_{t+1}P_0^{(2)}$ and $C_{t+1}P_0^{(1)} + D_{t+1}P_0^{(2)}$), and binds this public key with a one-way function of the signed message (thereby obtaining $\rho_t^{(1)}$ and $\rho_t^{(2)}$). Then, in the second step, the verifier determines whether the signer used coherent values (by checking whether $\hat{e}(U_1+U_2, P_0^{(5)}+P_0^{(6)}) = \hat{e}(U_5+U_6, P_0^{(1)}+P_0^{(2)})$ and $\hat{e}(U_1'+U_2', P_0^{(5)}+P_0^{(6)}) = \hat{e}(U_5'+U_6', P_0^{(1)}+P_0^{(2)})$), had the right signing key (by checking whether $\hat{e}(L_{t+2}^{(4)}, \rho_t^{(1)})\hat{e}(P_0^{(3)}, \widetilde{V}) = \hat{e}(P_0^{(4)}, V)\hat{e}(L_{t+2}^{(3)}, \rho_t^{(2)})$), had a hidden key that matches with his private key (by checking whether $\mathcal{H}_2(\varpi||(W_2^{\mathcal{H}_3(W_1)})^\theta U_1||\chi^\theta U_2') = \kappa$), and is not revoked (by checking whether $U_1' \neq rt_{\overline{ID}_t[j]}U_1$ and $(\beta-\gamma_{\overline{ID}_t[j]})U_1' = rt_{\overline{ID}_t[j]}P_0^{(1)}$, where $rt_{\overline{ID}_t[j]}$ is any revocation token given to the verifier by $\overline{ID}_t$).

Opening. The opening algorithm essentially consists of the very last part of the verification algorithm. Since the opener is the subgroup leader of a given signer, this leader is able to link a given revocation token with the identity (index) of each member of her subgroup.

Arbitrating. The arbitrating procedure is executed by a trusted third party (TTP). While the TTP does not have to hold any secret piece of information, it is trusted to follow to the following line of reasoning: since the private key used to issue a group signature is bound with the membership token of a signer, and since the revocation token of a user is equal to the membership token of this user, then the possession of a valid group signature which matches with a given revocation token uniquely identifies the corresponding user. Thus, the author of group signature can be identified.

7.3.2 Scheme

- `Instance Generator` (k). This procedure, denoted by IG, is a randomized algorithm which takes a security parameter $k > 0$, runs in $O(k)$ steps, and outputs $(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, SG)$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are two Abelian Gap-Diffie-Hellman groups of prime order $q \geq 2^k$, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is an admissible pairing with

respect to which the BDH problem is intractable, and SG is a prime order cyclic subgroup of $\mathbb{Z}_q^*$ in which the DL problem is intractable.

- `Root Setup` (k). Given a security parameter $k > 0$, the root PKG:

  1. runs IG with input k and obtains $(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, SG)$.
  2. picks, randomly and uniformly[1], $P_0^{(1)}, P_0^{(2)}, P_0^{(3)}, P_0^{(4)}, P_0^{(5)}, P_0^{(6)} \in \mathbb{G}_1$;
  3. picks $g \in_R SG$, and $s_0, I_0 \in_R \mathbb{Z}_q^*$, and sets $d_{\overline{ID}_0} = (s_0)$;
  4. computes $n = poly_1(k)$, $\ell = \log(k)$, $n_\tau = poly_2(k)$, and $n_{gs} = poly_3(k)$, where $poly_i$ is a polynomial over the positive integers, for $i = 1, 2, 3$;
  5. chooses cryptographic hash functions:
     $\mathcal{H}_1 : \{0,1\}^* \to (\mathbb{Z}_q^*)^{3\ell}$, $\mathcal{H}_2 : \{0,1\}^* \to \mathbb{Z}_q^*$, $\mathcal{H}_3 : \{0,1\}^{n_\tau} \to \mathbb{Z}_q^*$
     $\mathcal{H}_4 : \{0,1\}^n \to \mathbb{Z}_q^*$, $\mathcal{H}_5 : \mathbb{Z}_q^* \to \mathbb{Z}_q^*$,
     where $\mathcal{H}_1$ is defined as in NewHIBE.
  6. computes, $(s_i = \mathcal{H}_5(s_{i-1}))_{i=1}^{\ell-1}$, $L_1^{(j)} = s_0 P_0^{(j)}$ for $j = 3, 4$, and $(L_i^{(j)} = s_{i-1} L_{i-1}^{(j)})_{i=2}^{\ell}$ for $j = 3, 4$.

  The message space is $\mathcal{M} = \{0,1\}^n$ and the signature space is $\mathcal{S} = \mathbb{G}_1^6 \times \{0,1\}^{n_\tau} \times \mathbb{Z}_q^{*5}$. The system's public parameters (which must be certified) are $pubParams = (g, q, n, \hat{e}, I_0, P_0^{(1)}, P_0^{(2)}, P_0^{(3)}, P_0^{(4)}, P_0^{(5)}, P_0^{(6)}, \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4, \mathcal{H}_5, ((L_i^{(j)})_{i=1}^{\ell})_{j=3}^{4})$. The *root PKG* keeps s_0 secret, so that $params = (pubParams, s_0)$.

- `Sub-Leader-Key Generation` $(ID_{t+1}, d_{\overline{ID}_t})$: For each child-leader $\overline{ID}_{t+1} = (ID_1, \cdots, ID_{t+1})$ of a leader $\overline{ID}_t$, the following takes place:

  1. $\overline{ID}_t$ picks $\alpha_{\overline{ID}_{t+1}}, \widetilde{\alpha}_{\overline{ID}_{t+1}} \in_R \mathbb{Z}_q^*$.
  2. – If $t = 0$, then $rLead$ computes $J_1 = (I_1, p_{(1,1)}, p_{(1,3)})$ (with ID_1 and $\mathcal{H}_1$), $p_{(1,2)} = \mathcal{H}_7(p_{(1,3)})$, $p_{(1,4)} = \frac{p_{(1,2)} p_{(1,3)}}{p_{(1,1)}}$, $I_1' = -p_{(1,2)}$, $I_1'' = -\frac{p_{(1,1)}}{p_{(1,2)}} I_1'$, $\widetilde{I}_1 = I_1 + p_{(1,1)}$, $\widetilde{I}_1' = I_1' + p_{(1,2)}$, $\widetilde{I}_1'' = I_1'' + p_{(1,4)}$, $R_{\overline{ID}_1} = p_{(1,1)} \alpha_{\overline{ID}_1} + p_{(1,2)} \widetilde{\alpha}_{\overline{ID}_1}$, $\widetilde{R}_{\overline{ID}_1} = p_{(1,3)} \alpha_{\overline{ID}_1} + p_{(1,4)} \widetilde{\alpha}_{\overline{ID}_1}$, $s_1 = \mathcal{H}_5(s_0)$, and the following:

$$S_{\overline{ID}_1} = s_0(I_1 P_0^{(1)} + I_1' P_0^{(2)} + \alpha_{\overline{ID}_1} P_0^{(3)}),$$
$$E_{\overline{ID}_1} = s_0(\tfrac{I_1 I_1''}{I_1'} P_0^{(1)} + I_1'' P_0^{(2)} + \widetilde{\alpha}_{\overline{ID}_1} P_0^{(3)}),$$
$$\widetilde{S}_{\overline{ID}_1} = s_0(\alpha_{\overline{ID}_1} P_0^{(4)} + \widetilde{I}_1 P_0^{(5)} + \widetilde{I}_1' P_0^{(6)}),$$
$$\widetilde{E}_{\overline{ID}_1} = s_0(\widetilde{\alpha}_{\overline{ID}_1} P_0^{(4)} + \tfrac{\widetilde{I}_1 \widetilde{I}_1''}{\widetilde{I}_1'} P_0^{(5)} + \widetilde{I}_1'' P_0^{(6)});$$

[1] In the sequel, we shall use the notation $x \in_R X$ to indicate that the element x is chosen uniformly at random from the set X.

– Otherwise (i.e. if $t \geq 1$), $\overline{ID}_t$ computes: $J_t = (I_t, p_{(t,1)}, p_{(t,3)})$ and $J_{t+1} = (I_{t+1}, p_{(t+1,1)}, p_{(t+1,3)})$ (with ID_t, ID_{t+1} and $\mathcal{H}_1$); $I'_{t+1} = -p_{(t,2)}$, $I''_{t+1} = -\frac{p_{(t,2)}p_{(t,3)}}{p_{(t,1)}}$, $\widetilde{I}_{t+1} = I_{t+1} + p_{(t,1)}$, $\widetilde{I}'_{t+1} = I'_{t+1} + p_{(t,2)}$, $\widetilde{I}''_{t+1} = I''_{t+1} + p_{(t,4)}$; $p_{(t+1,2)} = -\frac{p_{(t,1)}}{p_{(t,3)}}p_{(t+1,1)}$, $p_{(t+1,4)} = \frac{p_{(t+1,2)}p_{(t+1,3)}}{p_{(t+1,1)}}$, $\alpha_{\overline{ID}_{t+1}}, \widetilde{\alpha}_{\overline{ID}_{t+1}} \in_R \mathbb{Z}_q^*$, $R_{\overline{ID}_{t+1}} = p_{(t+1,1)}\alpha_{\overline{ID}_{t+1}} + p_{(t+1,2)}\widetilde{\alpha}_{\overline{ID}_{t+1}}$, $\widetilde{R}_{\overline{ID}_{t+1}} = p_{(t+1,3)}\alpha_{\overline{ID}_{t+1}} + p_{(t+1,4)}\widetilde{\alpha}_{\overline{ID}_{t+1}}$; $s_{t+1} = \mathcal{H}_5(s_t)$, and the following:

$$
\begin{aligned}
S_{\overline{ID}_{t+1}} &= s_t(I_{t+1}S_{\overline{ID}_t} + I'_{t+1}E_{\overline{ID}_t}) + \alpha_{\overline{ID}_{t+1}}L^{(3)}_{t+1},\\
E_{\overline{ID}_{t+1}} &= s_t(\tfrac{I_{t+1}I''_{t+1}}{I'_{t+1}}S_{\overline{ID}_t} + I''_{t+1}E_{\overline{ID}_t} + \widetilde{\alpha}_{t+1}L^{(3)}_{t+1},\\
\widetilde{S}_{\overline{ID}_{t+1}} &= s_t(\widetilde{I}_{t+1}\widetilde{S}_{\overline{ID}_t} + \widetilde{I}'_{t+1}\widetilde{E}_{\overline{ID}_t}) + (\alpha_{\overline{ID}_{t+1}} - R_{\overline{ID}_t})L^{(4)}_{t+1}, \text{ and}\\
\widetilde{E}_{\overline{ID}_{t+1}} &= s_t(\tfrac{\widetilde{I}_{t+1}\widetilde{I}''_{t+1}}{\widetilde{I}'_{t+1}}\widetilde{S}_{\overline{ID}_t} + \widetilde{I}''_{t+1}\widetilde{E}_{\overline{ID}_t} + (\widetilde{\alpha}_{t+1} - \widetilde{R}_{\overline{ID}_t})L^{(4)}_{t+1};
\end{aligned}
$$

3. Then, $\overline{ID}_t$ secretly gives $d_{\overline{ID}_{t+1}} = (S_{\overline{ID}_{t+1}}, E_{\overline{ID}_{t+1}}, \widetilde{S}_{\overline{ID}_{t+1}}, R_{\overline{ID}_{t+1}}, \widetilde{R}_{\overline{ID}_{t+1}}, s_{t+1})$ to $\overline{ID}_{t+1}$.

- `User-Key Generation` ($mt_{\overline{ID}_t}$): For each member $\overline{ID}_t[i]$ of the group leader identified by $\overline{ID}_t$ (where $1 \leq i \leq n_{gs}$), the following takes place:

1. $\overline{ID}_t[i]$ picks $mgt_{\overline{ID}_t[i]}, \lambda \in_R \mathbb{Z}_q^*$, computes $mt_{\overline{ID}_t[i]} = (g^{\mathcal{H}_3(T)})^{mgt_{\overline{ID}_t[i]}} \in SG$, $\kappa = \mathcal{H}_2(||(g^{\mathcal{H}_3(T)})^{\lambda}||)$, and $\theta = \lambda - \kappa \cdot mgt_{\overline{ID}_t[i]}$, and secretly sends $(mt_{\overline{ID}_t[i]}, T, \kappa, \theta)$ to $\overline{ID}_t$, where $T \in \{0,1\}^{\tau}$ is the validity period of $mt_{\overline{ID}_t[i]}$.
2. Upon reception of $(mt_{\overline{ID}_t[i]}, T, \kappa, \theta)$, $\overline{ID}_t$ checks whether $\mathcal{H}_2(||(g^{\mathcal{H}_3(T)})^{\theta}mt^{\kappa}_{\overline{ID}_t[i]}||) = \kappa$. If this condition is not satisfied, $\overline{ID}_t$ sends "Invalid", and interrupts the key generation process. Otherwise:
3. $\overline{ID}_t$ picks $\alpha_{\overline{ID}_t[i]}, \widetilde{\alpha}_{\overline{ID}_t[i]} \in_R \mathbb{Z}_q^*$, sets $rt_{\overline{ID}_t[i]} = mt_{\overline{ID}_t[i]}$, and stores $(\overline{ID}_t[i], rt_{\overline{ID}_t[i]})$.
4. – If $t = 0$, the root leader computes $I'_0 = \mathcal{H}_7(I_0)$, $I''_0 = \mathcal{H}_7(I'_0)$, $\widetilde{I}_0 = \mathcal{H}_7(I''_0)$, $\widetilde{I}'_0 = \mathcal{H}_7(\widetilde{I}_0)$, $\widetilde{I}''_0 = \mathcal{H}_7(\widetilde{I}'_0)$, $s_1 = \mathcal{H}_5(s_0)$, and the following:

$$
\begin{aligned}
S_{\overline{ID}_0[i]} &= s_0(I_0P_0^{(1)} + I'_0mt_{\overline{ID}_0[i]}P_0^{(2)} + \alpha_{\overline{ID}_0[i]}P_0^{(3)}),\\
E_{\overline{ID}_1} &= s_0(\tfrac{I_0I''_0}{I'_0}P_0^{(1)} + I''_0mt_{\overline{ID}_0[i]}P_0^{(2)} + \widetilde{\alpha}_{\overline{ID}_0[i]}P_0^{(3)}),\\
\widetilde{S}_{\overline{ID}_0[i]} &= s_0(\alpha_{\overline{ID}_0[i]}P_0^{(4)} + \widetilde{I}_0P_0^{(5)} + \widetilde{I}'_0mt_{\overline{ID}_0[i]}P_0^{(6)}),\\
\widetilde{E}_{\overline{ID}_0[i]} &= s_0(\widetilde{\alpha}_{\overline{ID}_0[i]}P_0^{(4)} + \tfrac{\widetilde{I}_0\widetilde{I}''_0}{\widetilde{I}'_0}P_0^{(5)} + \widetilde{I}''_0mt_{\overline{ID}_0[i]}P_0^{(6)});
\end{aligned}
$$

– Otherwise (i.e. if $t \geq 1$):

(a) $\overline{ID}_t$ sends $mt_{\overline{ID}_t[i]}$ to its parent $\overline{ID}_{t-1}$;

(b) $\overline{ID}_{t-1}$ computes $I'_0 = \mathcal{H}_7(I_0)$, $I''_0 = \mathcal{H}_7(I'_0)$, $\widetilde{I}_0 = \mathcal{H}_7(I''_0)$, $\widetilde{I}'_0 = \mathcal{H}_7(\widetilde{I}_0)$, $\widetilde{I}''_0 = \frac{I''_0\widetilde{I}'_0}{I'_0}$, $R'_{\overline{ID}_t} = (\widetilde{I}_0 - I_0)\alpha_{\overline{ID}_t} + mt_{\overline{ID}_t[i]}(\widetilde{I}'_0 - I'_0)\widetilde{\alpha}_{\overline{ID}_t}$, $\widetilde{R}'_{\overline{ID}_t} = (\frac{\widetilde{I}_0\widetilde{I}''_0I'_0 - I_0I''_0I'_0}{\widetilde{I}'_0I'_0})\alpha_{\overline{ID}_t} + mt_{\overline{ID}_t[i]}(\widetilde{I}''_0 - I''_0)\widetilde{\alpha}_{\overline{ID}_t}$ (where $\alpha_{\overline{ID}_t}$ and

$\widetilde{\alpha}_{\overline{ID}_t}$ were used to compute $\overline{ID}_t$ private key components), and returns $(R'_{\overline{ID}_t}, \widetilde{R}'_{\overline{ID}_t})$ to $\overline{ID}_t$;

(c) $\overline{ID}_t$ computes $I'_0 = \mathcal{H}_7(I_0)$, $I''_0 = \mathcal{H}_7(K'_0)$, $\widetilde{I}_0 = \mathcal{H}_7(I''_0)$, $\widetilde{I}'_0 = \mathcal{H}_7(\widetilde{I}_0)$, $\widetilde{I}''_0 = \frac{I''_0 \widetilde{I}'_0}{I'_0}$, $s_{t+1} = \mathcal{H}_5(s_t)$, and the following:

$$
\begin{aligned}
S_{\overline{ID}_{t+1}} &= s_t(I_0 S_{\overline{ID}_t} + I'_0 mt_{\overline{ID}_t[i]} E_{\overline{ID}_t}) + \alpha_{\overline{ID}_t[i]} L^{(3)}_{t+1}, \\
E_{\overline{ID}_{t+1}} &= s_t(\tfrac{I_0 I''_0}{I'_0} S_{\overline{ID}_t} + I''_0 mt_{\overline{ID}_t[i]} E_{\overline{ID}_t}) + \widetilde{\alpha}_{\overline{ID}_t[i]} L^{(3)}_{t+1}, \\
\widetilde{S}_{\overline{ID}_{t+1}} &= s_t(\widetilde{I}_0 \widetilde{S}_{\overline{ID}_t} + \widetilde{I}'_0 mt_{\overline{ID}_t[i]} \widetilde{E}_{\overline{ID}_t}) + (\alpha_{\overline{ID}_t[i]} - R'_{\overline{ID}_t}) L^{(4)}_{t+1}, \text{ and} \\
\widetilde{E}_{\overline{ID}_{t+1}} &= s_t(\tfrac{\widetilde{I}_0 \widetilde{I}''_0}{\widetilde{I}'_0} \widetilde{S}_{\overline{ID}_t} + \widetilde{I}''_0 mt_{\overline{ID}_t[i]} \widetilde{E}_{\overline{ID}_t}) + (\alpha_{\overline{ID}_t[i]} - \widetilde{R}'_{\overline{ID}_t}) L^{(4)}_{t+1};
\end{aligned}
$$

5. $\overline{ID}_t$ picks $\gamma_{\overline{ID}_t[i]} \in_R \mathbb{Z}^*_q$. (This value is used by $\overline{ID}_t$ to verify $\overline{ID}_t[i]$'s signatures.)
6. $\overline{ID}_t$ secretly gives $sk_{\overline{ID}_t[i]} = (S_{\overline{ID}_t[i]}, E_{\overline{ID}_t[i]}, \widetilde{S}_{\overline{ID}_t[i]}, \widetilde{E}_{\overline{ID}_t[i]}, s_{t+1}, \gamma_{\overline{ID}_t[i]})$ to $\overline{ID}_t[i]$.
7. Then, $\overline{ID}_t[i]$ sets her private key $d_{\overline{ID}_t[i]} = (mgt_{\overline{ID}_t[i]}, sk_{\overline{ID}_t[i]})$.

   Note that, for robustness purposes, $\overline{ID}_t[i]$ can, via the *Signing* and *Verifying* algorithms, make sure that $(S_{\overline{ID}_t[i]}, E_{\overline{ID}_t[i]}, \widetilde{S}_{\overline{ID}_t[i]}, \widetilde{E}_{\overline{ID}_t[i]}, s_{t+1})$ matches $mt_{\overline{ID}_t[i]}$. Therefore, $\overline{ID}_t$ needs not compute $cmt_{\overline{ID}_t[i]}$, nor does this value need to be stored by $\overline{ID}_t[i]$.
8. Finally, $\overline{ID}_t[i]$ and $\overline{ID}_t$ co-generate a random *authentication key* to be used (e.g. via a message authentication code) for the authentication of $\overline{ID}_t$ in subsequent communications.

- `Signing` $(m, d_{\overline{ID}_t}[i])$: Given a message $m \in \mathcal{M}$, the membership generating token $mgt_{\overline{ID}_t}[i]$ of a signer $\overline{ID}_t[i]$, and the system's public parameters, this algorithm:

  1. picks $r, \lambda \in_R \mathbb{Z}^*_q$, computes $\beta_r = (r^{\mathcal{H}_3(T)})^{mgt_{\overline{ID}_t[i]}}$, and derives the following:

$$
\begin{array}{lll}
\underline{S}_{\overline{ID}_t[i]} = \beta_r S_{\overline{ID}_t[i]}, & \underline{\widetilde{S}}_{\overline{ID}_t[i]} = \beta_r \widetilde{S}_{\overline{ID}_t[i]}, & U_1 = \beta_r P^{(1)}_0, \\
\underline{E}_{\overline{ID}_t[i]} = \beta_r E_{\overline{ID}_t[i]}, & \underline{\widetilde{E}}_{\overline{ID}_t[i]} = \beta_r \widetilde{E}_{\overline{ID}_t[i]}, & U'_1 = \beta_r mt_{\overline{ID}_t[i]} P^{(1)}_0, \\
U_2 = \beta_r P^{(2)}_0, & U'_2 = \beta_r mt_{\overline{ID}_t[i]} P^{(2)}_0, & U_5 = \beta_r P^{(5)}_0, \\
U'_5 = \beta_r mt_{\overline{ID}_t[i]} P^{(5)}_0, & U_6 = \beta_r P^{(6)}_0, & U'_6 = \beta_r mt_{\overline{ID}_t[i]} P^{(6)}_0, \\
\theta = \lambda - mgt_{\overline{ID}_t[i]}, & \beta = \beta_r^{-1} + \gamma_{\overline{ID}_t[i]}, & \chi = (r \cdot g)^{\mathcal{H}_3(T)}, \\
W_1 = T, & W_2 = r, & \varpi = \mathcal{H}_2(mgt_{\overline{ID}_t[i]}),
\end{array}
$$

  $\kappa = \mathcal{H}_2(\varpi || (r^{\mathcal{H}_3(T)})^{\lambda} P^{(1)}_0 || \chi^{\lambda} P^{(2)}_0)$;

  2. picks $\alpha_m \in_R \mathbb{Z}^*_q$, and computes:

$I_m = \mathcal{H}_4(m)$, $I'_m = \mathcal{H}_7(I_m)$, $V = s_{t+1}(I_m \underline{S}_{\overline{ID}_t[i]} + I'_m \underline{E}_{\overline{ID}_t[i]}) + \alpha_m L^{(3)}_{t+2}$, $\widetilde{V} = s_{t+1}(I_m \underline{\widetilde{S}}_{\overline{ID}[i]} + I'_m \underline{\widetilde{E}}_{\overline{ID}_t[i]}) + \alpha_m L^{(4)}_{t+2}$;

3. outputs $\sigma = (U_1, U_1, U_2, U'_2, U_5, U'_5, U_6, U'_6, V, \widetilde{V}, W_1, W_2, \beta, \kappa, \theta, \varpi)$

- `Verifying` $(RL_{\overline{ID}_t}, m, \sigma)$: Given a message m, a signature $\sigma = (U_1, U_1, U_2, U'_2, U_5, U'_5, U_6, U'_6, V, \widetilde{V}, W_1, W_2, \beta, \kappa, \theta, \varpi)$, and the system's public parameters, this algorithm:

1. computes $(J_1, \cdots, J_t, J_0, \cdots, J_0) = \mathcal{H}_1(\overline{ID}_t) \in (\mathbb{Z}^*_q)^{3\ell}$, $I_m = \mathcal{H}_4(m)$, $I'_m = \mathcal{H}_7(I_m)$, $\rho^{(1)}_t = A_{t+2}P^{(1)}_0 + B_{t+2}P^{(2)}_0$, and $\rho^{(2)}_t = C_{t+2}P^{(5)}_0 + D_{t+2}P^{(6)}_0$ where:
   - If $t = 0$, then
     $I'_0 = \mathcal{H}_7(I_0)$, $I''_0 = \mathcal{H}_7(I'_0)$, $\widetilde{I}_0 = \mathcal{H}_7(I''_0)$, $\widetilde{I}'_0 = \mathcal{H}_7(\widetilde{I}_0)$, $\widetilde{I}''_0 = \mathcal{H}_7(\widetilde{I}'_0)$,
     $A_2P^{(1)}_0 = (I_mI_0 + I'_m\frac{I_0I''_0}{I'_0})U_1$, $B_2P^{(2)}_0 = (I_mI'_0 + I'_mI''_0)U'_2$,
     $C_2P^{(5)}_0 = (I_m\widetilde{I}_0 + I'_m\frac{\widetilde{I}_0\widetilde{I}''_0}{\widetilde{I}'_0})U_5$, $D_2P^{(6)}_0 = (I_m\widetilde{I}'_0 + I'_m\widetilde{I}''_0)U'_6$,
   - Otherwise :
     $p_{(1,2)} = \mathcal{H}_7(p_{(1,3)})$, $p_{(1,4)} = \frac{p_{(1,2)}p_{(1,3)}}{p_{(1,1)}}$, $I'_1 = -p_{(1,2)}$,
     $I''_1 = -\frac{p_{(1,1)}}{p_{(1,2)}}I'_1$, $\widetilde{I}_1 = I_1 + p_{(1,1)}$, $\widetilde{I}'_1 = I'_1 + p_{(1,2)}$, $\widetilde{I}''_1 = I''_1 + p_{(1,4)}$
     $A_1 = I_1$, $B_1 = I'_1$, $C_1 = \widetilde{I}_1$, $D_1 = \widetilde{I}'_1$,
     $\widetilde{A}_1 = \frac{I_1I''_1}{I'_1}$, $\widetilde{B}_1 = I''_1$, $\widetilde{C}_1 = \frac{\widetilde{I}_1\widetilde{I}''_1}{\widetilde{I}'_1}$, $\widetilde{D}_1 = \widetilde{I}''_1$,
     and, for $1 \le i < t$,
     $I'_{i+1} = -p_{(i,2)}$, $I''_{i+1} = -\frac{p_{(i,2)}p_{(i,3)}}{p_{(i,1)}}$, $\widetilde{I}_{i+1} = I_{i+1} + p_{(i,1)}$,
     $\widetilde{I}'_{i+1} = I'_{i+1} + p_{(i,2)}$, $\widetilde{I}''_{i+1} = I''_{i+1} + p_{(i,4)}$;
     $p_{(i+1,2)} = -\frac{p_{(i,1)}}{p_{(i,3)}}p_{(i+1,1)}$, $p_{(i+1,4)} = \frac{p_{(i+1,2)}p_{(i+1,3)}}{p_{(i+1,1)}}$,
     $A_{i+1} = I_{i+1}A_i + I'_{i+1}\widetilde{A}_i$, and $\widetilde{A}_{i+1} = \frac{I_{i+1}I''_{i+1}}{I'_{i+1}}A_i + I''_{i+1}\widetilde{A}_i$,
     $B_{i+1} = I_{i+1}B_i + I'_{i+1}\widetilde{B}_i$, and $\widetilde{B}_{i+1} = \frac{I_{i+1}I''_{i+1}}{I'_{i+1}}B_i + I''_{i+1}\widetilde{B}_i$,
     $C_{i+1} = \widetilde{I}_{i+1}C_i + \widetilde{I}'_{i+1}\widetilde{C}_i$, and $\widetilde{C}_{i+1} = \frac{\widetilde{I}_{i+1}\widetilde{I}''_{i+1}}{\widetilde{I}'_{i+1}}C_i + \widetilde{I}''_{i+1}\widetilde{C}_i$,
     $D_{i+1} = \widetilde{I}_{i+1}D_i + \widetilde{I}'_{i+1}\widetilde{D}_i$, and $\widetilde{D}_{i+1} = \frac{\widetilde{I}_{i+1}\widetilde{I}''_{i+1}}{\widetilde{I}'_{i+1}}D_i + \widetilde{I}''_{i+1}\widetilde{D}_i$.
     Then,
     $I'_0 = \mathcal{H}_7(I_0)$, $I''_0 = \mathcal{H}_7(I'_0)$, $\widetilde{I}_0 = \mathcal{H}_7(I''_0)$, $\widetilde{I}'_0 = \mathcal{H}_7(\widetilde{I}_0)$, $\widetilde{I}''_0 = \frac{I''_0\widetilde{I}'_0}{I'_0}$,
     $A_{t+1}P^{(1)}_0 = I_0A_tU_1 + I'_0\widetilde{A}_tU'_1$, $\widetilde{A}_{t+1}P^{(1)}_0 = \frac{I_0I''_0}{I'_0}A_tU_1 + I''_0\widetilde{A}_tU'_1$,
     $B_{t+1}P^{(2)}_0 = I_0B_tU_2 + I'_0\widetilde{B}_tU'_2$, $\widetilde{B}_{t+1}P^{(2)}_0 = \frac{I_0I''_0}{I'_0}B_tU_2 + I''_0\widetilde{B}_tU'_2$,
     $C_{t+1}P^{(5)}_0 = \widetilde{I}_0C_tU_5 + \widetilde{I}'_0\widetilde{C}_tU'_5$, $\widetilde{C}_{t+1}P^{(5)}_0 = \frac{\widetilde{I}_0\widetilde{I}''_0}{\widetilde{I}'_0}C_tU_5 + \widetilde{I}''_0\widetilde{C}_tU'_5$,
     $D_{t+1}P^{(6)}_0 = \widetilde{I}_0D_tU_6 + \widetilde{I}'_0\widetilde{D}_tU'_6$, $\widetilde{D}_{t+1}P^{(6)}_0 = \frac{\widetilde{I}_0\widetilde{I}''_0}{\widetilde{I}'_0}D_tU_6 + \widetilde{I}''_0\widetilde{D}_tU'_6$,
     $A_{t+2}P^{(1)}_0 = I_mA_{t+1}P^{(1)}_0 + I'_m\widetilde{A}_{t+1}P^{(1)}_0$,
     $B_{t+2}P^{(2)}_0 = I_mB_{t+1}P^{(2)}_0 + I'_m\widetilde{B}_{t+1}P^{(2)}_0$,
     $C_{t+2}P^{(5)}_0 = I_mC_{t+1}P^{(5)}_0 + I'_m\widetilde{C}_{t+1}P^{(5)}_0$,
     $D_{t+2}P^{(6)}_0 = I_mD_{t+1}P^{(6)}_0 + I'_m\widetilde{D}_{t+1}P^{(6)}_0$.

2. – outputs "valid" if the following conditions are satisfied:
   * $\hat{e}(U_1 + U_2, P_0^{(5)} + P_0^{(6)}) = \hat{e}(P_0^{(1)} + P_0^{(2)}, U_5 + U_6)$ and $\hat{e}(U_1' + U_2', P_0^{(5)} + P_0^{(6)}) = \hat{e}(P_0^{(1)} + P_0^{(2)}, U_5' + U_6')$;
   * $\hat{e}(L_{t+2}^{(4)}, \rho_t^{(1)})\hat{e}(P_0^{(3)}, \widetilde{V}) = \hat{e}(P_0^{(4)}, V)\hat{e}(L_{t+2}^{(3)}, \rho_t^{(2)})$;
   * $\mathcal{H}_2(\varpi||(W_2^{\mathcal{H}_3(W_1)})^{\theta}U_1||\chi^{\theta}U_2') = \kappa$, where $\chi = (W_2 \cdot g)^{\mathcal{H}_3(W_1)}$;
   * $U_1' \neq rt_{\overline{ID}_t[i]}U_1$, for each revocation token $rt_{\overline{ID}_t[i]}$ included in $RL_{\overline{ID}_t}$.
   * $(\beta - \gamma_{\overline{ID}_t[i]})U_1' = rt_{\overline{ID}_t[i]}P_0^{(1)}$ for some unrevoked member $\overline{ID}_t[i]$ of a $\overline{ID}_t$'s group, whenever the *Verifying* algorithm is run by $\overline{ID}_t$.

   – outputs "invalid" otherwise.

Note that the identifier $\overline{ID}_t[i]$ may be kept secret from the verifier, by including, in $RL_{\overline{ID}_t}$, only the revocation tokens of $\overline{ID}_t$'s revoked group members. Note also that the fifth condition of acceptance is a zero-knowledge proof that the signer used (in the computation of β_r) the same discrete logarithm that was used to generate the key sent by $\overline{ID}_t$ to $\overline{ID}_t[i]$, where i is unknown to the verifier.

- `Opening` $(RL_{\overline{ID}_t}, m, \sigma)$: Given a signature $\sigma = (U_1, U_1, U_2, U_2', U_5, U_5', U_6, U_6', V, \widetilde{V}, W_1, W_2, \beta, \kappa, \theta, \varpi)$ of m which is valid with respect to both $\overline{ID}_t$ and $RL_{\overline{ID}_t}$, this algorithm:

  1. looks for the user identifier $\overline{ID}_t[i]$ $(1 \leq j \leq n_{gs})$ such that
  $$U_1' = rt_{\overline{ID}_t[i]}U_1.$$
  2. If no identifier $\overline{ID}_t[i]$ can be found such that the equality holds, the algorithm returns "Fail". Otherwise, $\overline{ID}_t$ outputs $\overline{ID}_t[i]$ as the claimed issuer of σ.

- `Arbitrating` $(RL_{\overline{ID}_t}, m, \sigma)$: Given a signature $\sigma = (U_1, U_2, U_3, U_4, V, \widetilde{V}, W_1, W_2, \beta, \kappa, \theta, \varpi)$ of m such that σ is valid with respect to both $\overline{ID}_t$ and $RL_{\overline{ID}_t}$, and such that a valid group member $\overline{ID}_t[i]$ refutes $\overline{ID}_t$'s claim that $\overline{ID}_t[i]$ issued σ, the algorithm proceeds as follows:

  1. A trusted third party (TTP) requests $rt_{\overline{ID}_t[i]}$ from $\overline{ID}_t$.
  2. – If $U_1' = rt_{\overline{ID}_t[j]}U_1$, then the TTP outputs "$\overline{ID}_t[i]$ issued σ".
     – Otherwise, TTP outputs "$\overline{ID}_t[i]$ did not issue σ".

  Note that $\overline{ID}_t[i]$ needs not reveal $mgt_{\overline{ID}_t[i]}$ to the TTP.

7.4 Efficiency

	Our HIBGS	IDGS [CZK03]	GS with VLR [BS04]
Root Setup	$(\ell-1)H + 2\ell M_{\mathbb{G}_1}$ $+ 6R_{\mathbb{G}_1} + 3R_{\mathbb{Z}_q^*}$	$M_{\mathbb{G}_1} + R_{\mathbb{Z}_q^*}$	$\psi + M_{\mathbb{G}_1} + R_{\mathbb{G}_1} + R_{\mathbb{Z}_q^*}$
Sub-Leader Key Generation	$A_{\mathbb{G}_1} + 7A_{\mathbb{Z}_q^*} + 6H + 3Inv_{\mathbb{Z}_q^*}$ $+ 12M_{\mathbb{G}_1} + 22M_{\mathbb{Z}_q^*} + 2R_{\mathbb{Z}_q^*}$	(UNABLE)	(UNABLE)
User Key Generation	$8A_{\mathbb{G}_1} + 5A_{\mathbb{Z}_q^*} + 4Ex_{\mathbb{Z}_q}$ $+ 11H + 4Inv_{\mathbb{Z}_q^*} + 12M_{\mathbb{G}_1}$ $+ 35M_{\mathbb{Z}_q^*} + 5R_{\mathbb{Z}_q^*}$	$\lvert SIG\rvert(2H + 3M_{\mathbb{G}_1}$ $+ 2\mathbf{P} + R_{\mathbb{Z}_q^*})$ $+ H + M_{\mathbb{G}_1} + R_{\mathbb{Z}_q^*}$	$Inv_{\mathbb{Z}_q^*} + M_{\mathbb{G}_1} + R_{\mathbb{Z}_q^*}$
Signing	$4A_{\mathbb{G}_1} + A_{\mathbb{Z}_q^*} + 2Ex_{\mathbb{Z}_q^*}$ $+ 5H + Inv_{\mathbb{Z}_q^*} + 19M_{\mathbb{G}_1}$ $+ 8M_{\mathbb{Z}_q^*} + 3R_{\mathbb{Z}_q^*}$	$A_{\mathbb{G}_1} + A_{\mathbb{Z}_q^*} + 3H$ $+ 3M_{\mathbb{G}_1} + R_{\mathbb{Z}_q^*}$	$2A_{\mathbb{G}_1} + 3A_{\mathbb{Z}_q^*} + 2H$ $+ 3Ex_{\mathbb{G}_2} + Inv_{\mathbb{G}_1} + Inv_{\mathbb{G}_2}$ $+ 5M_{\mathbb{G}_1} + 2M_{\mathbb{G}_2} + 4M_{\mathbb{Z}_q^*}$ $+ 3\mathbf{P} + 5R_{\mathbb{Z}_q^*} + 2\psi$
Verifying	$16A_{\mathbb{G}_1} + (11t-7)A_{\mathbb{Z}_q^*}$ $+ (2t+11)H + (5t-1)Inv_{\mathbb{Z}_q^*}$ $+ 2Ex_{\mathbb{Z}_q^*} + (28 + \lvert RL\rvert)M_{\mathbb{G}_1}$ $+ 2M_{\mathbb{G}_2} + (26t+4)M_{\mathbb{Z}_q^*} + 8\mathbf{P}$	$2A_{\mathbb{G}_1} + 3H + M_{\mathbb{G}_1} + 4\mathbf{P}$	$(2 + \lvert RL\rvert)A_{\mathbb{G}_1} + 4Ex_{\mathbb{G}_2} + 2H$ $+ Inv_{\mathbb{G}_1} + Inv_{\mathbb{G}_2} + 4M_{\mathbb{G}_1}$ $+ 4M_{\mathbb{G}_2} + 2\lvert RL\rvert\mathbf{P} + 2\psi$
Opening	$\lvert RL\rvert M_{\mathbb{G}_1}$	$H + 4\mathbf{P}$	$\lvert RL\rvert A_{\mathbb{G}_1} + 2\lvert RL\rvert\mathbf{P}$
Arbitrating	$M_{\mathbb{G}_1}$	$H + 4\mathbf{P}$	(UNABLE)

Table 7.1: Efficiency Comparison of our HIBGS, with related schemes.

Table 7.1 compares the computational requirements of our HIBGS scheme with Chen et al.'s ID-based group signature scheme (CZK-IDGS [CZK03]), and Boneh and Shacham's group signature scheme with Verifier-Local Revocation (BS-GS with VLR [BS04].) (See also Table 6.2 for corresponding timing estimates, based on information obtained from [HPS02].) t denotes both the hierarchical level of a signer and the hierarchical level of a subgroup leader or subgroup member whose key is to be generated. It is assumed that all hash functions of the compared schemes have the same computational cost, denoted by H. M_X and A_X respectively denote computational costs of scalar multiplication and addition in the Abelian group X. R_X denotes the computational cost of uniformly selecting a random element in the set X. The computational cost of exponentiation in the group X is denoted by Ex_X, $\mathbf{P}$ denotes the computational cost of a bilinear pairing operation, Inv_X denotes the computational cost of inversion in X, and ψ denotes the computational cost of an efficiently computable function (cf. [BS04]). In order to provide a realistic computational cost average for the *Verifying* algorithm, it is assumed in Table 7.1 that the verifier is any party, except the signature issuer's subgroup leader. When the verifier is the issuer's subgroup leader, the computational cost of verification is at most $(2M_{\mathbb{G}_1} + A_{\mathbb{Z}_q^*})n_{gs}$ higher, and the information learned by the verifier (i.e. the fact that $\gamma_{\overline{ID}_t[i]}$ was used

		Computational Requirements					
		$t = 1$ $\ell = 1$ $RL = 10$	$t = 3$ $\ell = 4$ $RL = 10$	$t = 4$ $\ell = 5$ $RL = 10$	$t = 10$ $\ell = 11$ $RL = 10$	$t = 12$ $\ell = 13$ $RL = 10$	$t = 20$ $\ell = 21$ $RL = 10$
IDGS [CZK03]	Root Setup	21×10^3	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
	Sub-Leader Key Generation	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
	User Key Generation	252×10^3	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
	Signing	63×10^3	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
	Verifying	357×10^3	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
	Opening	336×10^3	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
	Arbitrating	336×10^3	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
GS with VLR [BS04]	Root Setup	21×10^3	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
	Sub-Leader Key Generation	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
	User Key Generation	21×10^3	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
	Signing	358×10^3	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
	Verifying	1.8×10^6	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
	Opening	1.7×10^6	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
	Arbitrating	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE	UNABLE
Our HIBGS	Root Setup	42×10^3	168×10^3	210×10^3	462×10^3	546×10^3	882×10^3
	Sub-Leader Key Generation	253×10^3	253×10^3	253×10^3	253×10^3	253×10^3	253×10^3
	User Key Generation	255×10^3	255×10^3	255×10^3	255×10^3	255×10^3	255×10^3
	Signing	400×10^3	400×10^3	400×10^3	400×10^3	400×10^3	400×10^3
	Verifying	1.5×10^6	1.5×10^6	1.5×10^6	1.5×10^6	1.5×10^6	1.5×10^6
	Opening	210×10^3	210×10^3	210×10^3	210×10^3	210×10^3	210×10^3
	Arbitrating	21×10^3	21×10^3	21×10^3	21×10^3	21×10^3	21×10^3

Table 7.2: Timing Estimates of our HIBGS and Related Schemes, in Microseconds.

to generate the signature) is only used to formally prove the traceability of signatures by the appropriate subgroup leaders (cf. the proof of Theorem 7.5.2). Since all verifiers except the appropriate subgroup leaders should not be able to identify the issuers of subgroup signatures, the assumption of Table 7.1 seems reasonable to us. We note that all algorithms have constant storage requirements, apart from the following two exceptions: (1) the *Verifying* procedures of both our HIBGS scheme and Boneh and Shacham's scheme have storage requirements which grow linearly with respect to the size of the revocation list; (2) the *Root Setup* method of our HIBGS scheme has space requirements which grow linearly with the depth of the underlying hierarchy; (3) the storage requirements of Chen et al.'s *User-Key Generation* algorithm grow linearly with the number of issued signatures.

By the *(UNABLE)* entries of its second row, Table 7.1 notes that, among the three analyzed schemes, ours is the only one that (efficiently[2]) handles hierarchical settings. Moreover, the last row's *(UNABLE)* entry show that, unlike our scheme, Boneh and Shacham's scheme allows key generating entities to impersonate subgroup members, which removes the possibility for trusted third parties to deny any subgroup member's claim that she has been the victim of an impersonation.

Table 7.1 also reveals that, while being executable in constant time, the *User-Key Generation* algorithm of our proposed scheme is more computationally demanding than Boneh and Shacham's, yet significantly more efficient than Chen et al.'s (which requires one signature certificate to be issued for each signature). The *Signing* procedure of Chen et al.'s scheme is the most efficient among the three compared ones. However, due to the very high storage and computational requirements of Chen et al.'s *User-Key Generation* procedure, our *Signing* procedure improves on Boneh and Shacham's due to the fact that it requires no pairing computation. The greatest advantage of our scheme concerns the verification of signatures. Our scheme requires only 8 pairings, compared to $2|RL|$ in the case of Boneh and Schacham's scheme. Hence, our scheme is advantageous when $|RL| > 8$ (using the computational costs presented [HPS02]). Moreover, this improvement is significant when revocation lists are either large or growing. The *Opening* procedure of our scheme is similarly more efficient than Boneh et Shacham's. Note that Chen et al.'s scheme did not present a mechanism to check the revocation status of signers at signature verification time. Such a mechanism could however be added, and would require $|RL|$ additional pairings. For Chen et al.'s *Opening* algorithm, a revocation bit could be associated with

[2]Note that the two other schemes could be used in hierarchal settings by having a central entity manage the key generation and secure key distribution/update of each subgroup leader. However, such an approach is not computationally scalable to large hierarchies. Moreover, such an approach would allow attackers to easily target the key distribution entity, and launch efficient *denial of service attacks.*

each signature certificate entry in the leader's member list. This would allow *Opening* procedures to be executed in constant time, but would not remove the $O(|SIG|)$ storage requirement, where $|SIG|$ is the number of signatures to be issued by all users of the system. Another interesting comparison can be made between our HIBGS scheme and Boneh and Franklin's anonymous authentication scheme with subset queries [BF99]. Compared to this scheme, our scheme has constant computational requirements for *Signing* (compared to $O(|GM|)$, where $|GM|$ is the number of subgroup leaders in the whole hierarchy.) Furthermore, the computational cost of our scheme's *Verifying* procedure grows only logarithmically with respect to the number of subgroup leaders in the underlying hierarchy. This stands in contrast with Boneh and Franklin's analogous algorithm, which grows linearly with respect to $|GM|$.

7.5 Security

This section presents the security guarantees of our proposed HIBGS scheme. We first present the threat model, and then state three security theorems.

7.5.1 Threat Model

We propose a threat model for *HGS* schemes with PSI, using Bellare et al.'s framework [BMW03]. The model is presented in terms of three fundamental properties which this class of HGS schemes may be expected to have.

The first property (*correctness*) ensures that the signing and verifying procedures of the HGS scheme are complementary (i.e. that each group signature is valid if and only if it was properly issued). The second property (*full traceability*) ensures that no coalition of subgroup members and subgroup leaders can generate a group signature which cannot be opened or traced back to some member of the coalition. The third property (*full selfless anonymity*) avoids the possibility that valid group signatures leak information about their signers. Following the approach of Boneh and Shacham [BS04], we consider full *selfless* anonymity (as opposed to *full anonymity*) in order to underline the fact that signers are assumed to be able to recognize (open) their own signatures. In other words, signatures are not assumed to be anonymous to their own issuers. The second and the third properties are described via two games in which an attacker attempts to break the desired property.

Note that, unlike Boneh and Shacham's threat model [BS04], our model assumes that key issuers can be compromised without subgroup members running the risk of being impersonated by key issuers. We believe that this subtle difference is significant, because, in practice, subgroup leaders (by whom key issuers are typically instantiated) cannot be assumed never to attempt to maliciously impersonate their

subgroup members. Each of the three security requirements is formally presented below.

Correctness:

A HGS scheme is said to satisfy the *correctness* property if:

$$\forall\, m \in \mathcal{M}, \forall\, i \in \{1, 2, \cdots, n_{\overline{ID}_t}\}$$

$$\text{Verifying}(RL,\, m,\, \text{Signing}(m,\, d_{\overline{ID}_t[i]}) = \text{valid} \iff rt_{\overline{ID}_t[i]} \notin RL$$

Full Traceability:

Let Ψ be a *HIBGS* scheme. The following game, between a challenger $\mathcal{C}h$ and an attacker $\mathcal{A}$, is considered:

Setup: Given a security parameter k, $\mathcal{C}h$ uses Ψ's *Root Setup* algorithm to generate the cryptosystem's public and private parameters – keeping the private parameters secret while giving the public ones to $\mathcal{A}$. For each subgroup leader $\overline{ID}_t$, $\mathcal{C}h$ also provides $\mathcal{A}$ with $\overline{ID}_t$'s revocation token vector $RL_{\overline{ID}_t}$ and sets $U_{\overline{ID}_t}$ to be the empty set, where $U_{\overline{ID}_t}$ is the set of $\overline{ID}_t$'s subgroup members who collude against $\mathcal{C}h$.

Queries: $\mathcal{A}$ makes a polynomially bounded number of queries of the following type:

- **Public Key**: Given a subgroup leader's ID-tuple $\overline{ID}_{(i,t_i)} = (ID_{(i,1)}, \cdots, ID_{(i,t_i)})$, $\mathcal{C}h$ must return the public key associated with $\overline{ID}_{(i,t_i)}$.
- **Signing**: Given a signer's identifier $\overline{ID}_{(i,t_i)}[j]$ (where $1 \leq j \leq n_{\overline{ID}_{(i,t_i)}}$) and an arbitrary message m, $\mathcal{C}h$ computes and returns to $\mathcal{A}$ the signature $\sigma = \text{Signing}(m,\, d_{\overline{ID}_t[j]})$ of m.
- **Compromise**:
  - *Leader Compromise*: Given a subgroup leader's ID-tuple $\overline{ID}_{(i,t_i)}$, $\mathcal{C}h$ responds with $(d_{\overline{ID}_{(i,t_i)}}, sk_{\overline{ID}_{(i,t_i)}}, RL_{\overline{ID}_{(i,t_i)}})$, where $d_{\overline{ID}_{(i,t_i)}}$ is $\overline{ID}_{(i,t_i)}$'s private key, $sk_{\overline{ID}_{(i,t_i)}}$ is $\overline{ID}_{(i,t_i)}$'s user secret key vector, and $RL_{\overline{ID}_{(i,t_i)}}$ is $\overline{ID}_{(i,t_i)}$'s user revocation token vector.
  - *User Compromise*: Given a signer's identifier $\overline{ID}_{(i,t_i)}[j]$ (where $1 \leq j \leq n_{\overline{ID}_{(i,t_i)}}$), $\mathcal{C}h$ appends j to $U_{\overline{ID}_{(i,t_i)}}$ (the known coalition, from $\overline{ID}_{(i,t_i)}$'s subgroup, that is attempting to forge a signature) and responds with $d_{\overline{ID}_{(i,t_i)}[j]} = (mgt_{\overline{ID}_{(i,t_i)}[j]}, cmt_{\overline{ID}_{(i,t_i)}[j]}, sk_{\overline{ID}_{(i,t_i)}[j]})$, where $mgt_{\overline{ID}_{(i,t_i)}[j]}$ is

$\overline{ID}_{(i,t_i)}[j]$'s membership generating token, $cmt_{\overline{ID}_{(i,t_i)}[j]}$ is $\overline{ID}_{(i,t_i)}[j]$'s certified membership token, and $sk_{\overline{ID}_{(i,t_i)}[j]}$ is $\overline{ID}_{(i,t_i)}[j]$'s secret key.

Forgery: $\mathcal{A}$ chooses a subgroup leader's ID-tuple $\overline{ID}_t^*$, and sends to Ch the following: a message m^*, and a signature σ^*.

The attacker $\mathcal{A}$ wins the *Traceability* game if either one of the following conditions is satisfied: (A) $\mathcal{A}$ proves that Ch issued incoherent answers during the game; or (B) the following are true: (B.1) σ^* is accepted by the verification algorithm as a valid signature on m^*, with respect to $\overline{ID}_t^*$; (B.2) either σ^* traces to some user outside of the coalition $U_{\overline{ID}_t^*} \backslash RL^*_{\overline{ID}_t^*}$[3] (using the *Opening* procedure), or the *Opening* algorithm fails; and (B.3) σ^* is nontrivial, i.e., $\mathcal{A}$ did not obtain σ^* by making a signing query for m^* with any of $\overline{ID}_t^*$'s signers.

The following definition is predicated on the assumption that the security of HGS schemes is to be proved in the random oracle model.

Definition: A forger $\mathcal{A}$ $(\tau, n_{comp}, n_{sig}, n_{pk}, \epsilon)$-breaks traceability in a HGS scheme if: (1) $\mathcal{A}$ runs in time at most τ; (2) $\mathcal{A}$ makes at most n_{comp} *Compromise* queries, at most n_{sig} *Signing* queries, and at most n_{pk} *Public Key* queries; and (3) the probability that $\mathcal{A}$ wins the game is at least ϵ, where the probability is taken over the coin tosses of $\mathcal{A}$ and the randomized key generation and signing algorithms.

Selfless Full Anonymity:

For *selfless full anonymity*, the following game, between a challenger Ch and an attacker $\mathcal{A}$, is considered:

Setup: Given a security parameter k, Ch uses Ψ's *Root Setup* algorithm to generate the cryptosystem's public and private parameters - keeping the private parameters secret while giving the public ones to $\mathcal{A}$.

Queries: $\mathcal{A}$ makes a polynomially bounded number of queries of the following type:

- **Public Key**: Given a subgroup leader's ID-tuple $\overline{ID}_{(i,t_i)} = (ID_{(i,1)}, \cdots, ID_{(i,t_i)})$, Ch must return the public key associated with $\overline{ID}_{(i,t_i)}$.
- **Signing**: Given a signer's identifier $\overline{ID}_{(i,t_i)}[j]$ (where $1 \leq j \leq n_{\overline{ID}_{(i,t_i)}}$) and an arbitrary message m, Ch computes and returns to $\mathcal{A}$ the signature $\sigma =$ Signing$(m, d_{\overline{ID}_t[j]})$ of m.

[3]Note that, without loss of generality, the revocation list $RL_{\overline{ID}_t^*}$ of $\overline{ID}_t^*$'s subgroup could be set to the empty set.

- **Compromise**:
  - *Leader Compromise*: Given a subgroup leader's ID-tuple $\overline{ID}_{(i,t_i)}$, $\mathcal{C}h$ responds with $(d_{\overline{ID}_{(i,t_i)}}, sk_{\overline{ID}_{(i,t_i)}}, rt_{\overline{ID}_{(i,t_i)}})$, where $d_{\overline{ID}_{(i,t_i)}}$ is $\overline{ID}_{(i,t_i)}$'s private key, $sk_{\overline{ID}_{(i,t_i)}}$ is $\overline{ID}_{(i,t_i)}$'s user secret key vector, and $rt_{\overline{ID}_{(i,t_i)}}$ is $\overline{ID}_{(i,t_i)}$'s user revocation token vector.
  - *User Compromise*: Given a signer's identifier $\overline{ID}_{(i,t_i)}[j]$ (where $1 \leq j \leq n_{\overline{ID}_{(i,t_i)}}$), $\mathcal{C}h$ appends j to $U_{\overline{ID}_{(i,t_i)}}$ (the known coalition from $\overline{ID}_{(i,t_i)}$'s subgroup) and responds with $d_{\overline{ID}_{(i,t_i)}[j]} = (mgt_{\overline{ID}_{(i,t_i)}[j]}, cmt_{\overline{ID}_{(i,t_i)}[j]}, sk_{\overline{ID}_{(i,t_i)}[j]})$, where $mgt_{\overline{ID}_{(i,t_i)}[j]}$ is $\overline{ID}_{(i,t_i)}[j]$'s membership generating token, $cmt_{\overline{ID}_{(i,t_i)}[j]}$ is $\overline{ID}_{(i,t_i)}[j]$'s certified membership token, and $sk_{\overline{ID}_{(i,t_i)}[j]}$ is $\overline{ID}_{(i,t_i)}[j]$'s secret key.
- **Revocation**: Given a signer's identifier $\overline{ID}_{(i,t_i)}[j]$ (where $1 \leq j \leq n_{\overline{ID}_{(i,t_i)}}$), $\mathcal{C}h$ responds with $\overline{ID}_{(i,t_i)}[j]$'s revocation token $rt_{\overline{ID}_{(i,t_i)}[j]}$.
- **Opening**: Given the ID-tuple $\overline{ID}_t$ of a claimed subgroup leader, and a signature $\sigma \in \mathcal{S}$ which is valid with respect to $\overline{ID}_t$, $\mathcal{C}h$ returns either "Fail" or the identifier $\overline{ID}_t[i]$ of σ's signer.

Challenge: $\mathcal{A}$ outputs a target subgroup leader's ID-tuple $\overline{ID}_t^*$, a message m^*, and two indices i_0 and i_1 (where $1 \leq i_0, i_1 \leq n_{\overline{ID}_t^*}$). With respect to $\overline{ID}_t^*$, $\mathcal{A}$ must have made no *Leader Compromise*, *User Compromise*, or *User Revocation* queries for either index. Moreover, no *Leader Compromise* query should have been issued on $\overline{ID}_t^*$'s parent. The challenger $\mathcal{C}h$ chooses a bit $b \in \{0,1\}$ uniformly at random, computes a signature $\sigma^* = \text{Signing}(m^*, d_{\overline{ID}_t^*[i_b]})$ on m^* by user $\overline{ID}_t^*[i_b]$, and provides $\mathcal{A}$ with σ^*.

Restricted Queries: After obtaining the challenge, algorithm $\mathcal{A}$ can make additional queries of the challenger, restricted as follows:

- **Public Key**: $\mathcal{A}$ issues *Public Key* queries as before.
- **Signing**: $\mathcal{A}$ issues *Signing* queries as before.
- **Compromise**: As before, but $\mathcal{A}$ cannot make *Leader Compromise* queries with $\overline{ID}_t^*$, or *User Compromise* queries for i_0 or i_1, with $\overline{ID}_t^*$.
- **Revocation**: As before, but $\mathcal{A}$ cannot make *Revocation* queries for i_0 and i_1, with $\overline{ID}_t^*$.
- **Opening**: As before, but $\mathcal{A}$ cannot make *Opening* queries for i_0 and i_1, with $\overline{ID}_t^*$.

Output: $\mathcal{A}$ outputs a bit b', its guess of b.

$\mathcal{A}$ wins the *Anonymity* game if either one of the following conditions is satisfied: (A) $\mathcal{A}$ proves that $\mathcal{C}h$ issued incoherent answers during the game; or (B) $b' = b$.

Definition: An exposer $\mathcal{A}$ $(\tau, n_{comp}, n_{sig}, n_{pk}, \epsilon)$-breaks selfless full anonymity if: (1) $\mathcal{A}$ runs in time at most τ; (2) $\mathcal{A}$ makes at most n_{comp} *Compromise* queries, at most n_{sig} *Signing* queries, and at most n_{pk} *Public Key* queries; and (3) the advantage $|Pr[b = b'] - \frac{1}{2}|$ is at least ϵ, where the probabilities are taken over the coin tosses of $\mathcal{A}$, of the randomized key generation and signing algorithms, and the choice of b.

Definition: A hierarchical group signature scheme is $(\tau, n_{comp}, n_{sig}, n_{pk}, \epsilon)$-secure if it is correct, if no polynomially-bounded adversary $(\tau, n_{comp}, n_{sig}, n_{pk}, \epsilon)$-breaks full traceability, and if no polynomially-bounded adversary $(\tau, n_{comp}, n_{sig}, n_{pk}, \epsilon)$-breaks selfless anonymity.

7.5.2 Impersonation Threats

Since there are many different classes of impersonation attacks, we specify in this section the assumptions made in our threat model regarding impersonation.

Consider the following question: what happens if a malicious party creates a new subgroup? This could be done either to maliciously study the behavior of joining subgroup members, or to control the security privileges of joining members. Our threat model does not address such a possibility, and our proposed scheme is therefore not intended to be protect against attacks of this nature. A related concern is the following: what happens if a malicious party impersonates a valid subgroup leader, in such a way that a legitimate user joins the impersonated leader's subgroup ? Once again, our threat model does not take into consideration such a possibility, and our HIBGS is therefore not designed to protect against this class of impersonation attacks.

The strength of our scheme, however, is to provide a mechanism which detects the impersonation of subgroup leaders in their relationship with existing subgroup members. Indeed, we require each joining subgroup member to co-generate, with her subgroup leader, an authentication key which does not solely depend on the leader's HIBGS-based private key. This authentication key should be unique to the pair formed by the joining subgroup member and the subgroup leader. The authentication key is then used, in subsequent communications, to authenticate the subgroup leader. This avoids the risk of impersonation of subgroup leaders which have not been compromised.

7.5.3 Security Theorems

Theorem 7.5.1. *Our proposed HIBGS scheme satisfies the correctness property.*

Practical Meaning of Theorem 7.5.1: In practice, this theorem indicates that each group signature is valid if and only if it was properly issued.

Theorem 7.5.2. *Let* T *be a tree-shaped hierarchy in which any proportion of ID-tuples* $(ID_1, \cdots, ID_t)$ *having a common* root branch ID_1 *is a non-negligible quantity with respect to a security parameter* k. *Assume that, in our proposed HIBGS scheme, all hash functions are random oracles. Suppose also that there exists an attacker* $\mathcal{A}$, *which* $(\tau, n_{comp}, n_{sig}, n_{pk}, \varepsilon(k))$*-breaks traceability in our scheme. Then, there exists an algorithm* $\mathcal{B}$ *which solves the BDH problem, in time* $O(\tau)$, *with non-negligible advantage at least*

$$\varepsilon(k)\frac{n_{rb}-1}{n_{rb}}(1-\delta)^{n_{comp}+n_{sig}}\frac{1}{pol(k)}, \text{ where}$$

n_{rb} *is the number of root branches in* T, δ *is the largest proportion of ID-tuples having a common root branch, and pol is a polynomial such that* $\frac{1}{pol(k)}$ *is a lower bound of all proportions of ID-tuples having a common root branch.*

Practical Meaning of Theorem 7.5.2: In practice, this theorem stipulates that each valid group signature can be opened by its signer's subgroup manager.

Theorem 7.5.3. *Let* k *be a security parameter. Assume that the hash functions of our proposed HIBGS scheme are random oracles. Suppose also that there exists an attacker* $\mathcal{A}$, *which* $(\tau, n_{comp}, n_{sig}, n_{pk}, \varepsilon(k))$*-breaks selfless full anonymity in our scheme. Then, there exists an algorithm* $\mathcal{B}$ *which solves the DDL problem, in time* $O(\tau)$, *with non-negligible advantage at least* $\varepsilon(k)$.

Practical Meaning of Theorem 7.5.3: In practice, this theorem guarantees that the issuer of any valid group signature remains anonymous for all except the signer's subgroup manager.

Proof of Theorem 7.5.1. We prove both implications of the *Correctness* property.

- First, assume that $\sigma = (U_1, U_2, U_3, U_4, U_5, U_6, U_7, U_8, V, \widetilde{V}, W_1, W_2, \beta, \kappa, \theta, \varpi)$ is a properly issued signature of a message m, issued by a member $\overline{ID}_t[i]$ of a group leader $\overline{ID}_t$'s group. Then, for[4] $t > 1$:

  1. $V = (\prod_{i=1}^{t+2} s_{i-1})(A_{t+2}P_0^{(1)} + B_{t+2}P_0^{(2)} + F_{t+2}P_0^{(3)})$ and $\widetilde{V} = (\prod_{i=1}^{t+2} s_{i-1})(F_{t+2}P_0^{(4)} + C_{t+2}P_0^{(5)} + D_{t+2}P_0^{(6)})$.

[4]Note that the case $t = 1$ is straightforward.

2. Consequently:
$$\begin{aligned}
&\hat{e}(P_0^{(4)}, V)\hat{e}(P_0^{(3)}, \widetilde{V})^{-1} \\
&= \hat{e}(P_0^{(4)}, (\textstyle\prod_{i=1}^{t+2} s_{i-1})(\rho_t^{(1)} + F_{t+2}P_0^{(3)})) \cdot \hat{e}(P_0^{(3)}, (\textstyle\prod_{i=1}^{t+2} s_{i-1})(F_{t+2}P_0^{(4)} + \rho_t^{(2)}))^{-1} \\
&= \hat{e}((\textstyle\prod_{i=1}^{t+2} s_{i-1})P_0^{(4)}, \rho_t^{(1)}) \cdot \hat{e}((\textstyle\prod_{i=1}^{t+2} s_{i-1})P_0^{(3)}, \rho_t^{(2)})^{-1} \\
&= \hat{e}(L_{t+2}^{(4)}, \rho_t^{(1)}) \cdot \hat{e}(L_{t+2}^{(3)}, \rho_t^{(2)})^{-1}.
\end{aligned}$$

3. Moreover, $\hat{e}(U_1 + U_2, P_0^{(5)} + P_0^{(6)}) = \hat{e}(\beta_r(P_0^{(1)} + P_0^{(2)}), P_0^{(5)} + P_0^{(6)}) = \hat{e}(P_0^{(1)} + P_0^{(6)}, \beta_r(P_0^{(5)} + P_0^{(6)})) = \hat{e}(P_0^{(1)} + P_0^{(2)}, U_5 + U_6)$, and the equality $\hat{e}(U_1' + U_2', P_0^{(5)} + P_0^{(6)}) = \hat{e}(P_0^{(1)} + P_0^{(2)}, U_5' + U_6')$ holds in the same way.

4. Furthermore, $(\beta - \gamma_{\overline{ID}_t[i]})U_1' = \beta_r^{-1}(\beta_r mt_{\overline{ID}_t[i]}P_0^{(1)}) = rt_{\overline{ID}_t[i]}P_0^{(1)}$.

5. Also, $\mathcal{H}_2(\varpi||(W_2^{\mathcal{H}_3(W_1)})^\theta U_1||\chi^\theta U_2') = \kappa$ is a zero-knowledge proof that $U_1 = \varrho P_0^{(1)}$ and $U_2 = \varrho P_0^{(2)}$, such that $\varrho = (r^{\mathcal{H}_3(T)})^\xi$, where ξ is the discrete logarithm which was used by $\overline{ID}_t[i]$ to generate $mt_{\overline{ID}_t[i]} = (g^{\mathcal{H}_3(T)})^\xi$ and which was used by $\overline{ID}_t$ to generate $\overline{ID}_t[i]$'s private key. In other words, $\overline{ID}_t[i]$ must have issued σ. ($\overline{ID}_t[i]$ could not have been impersonated by $\overline{ID}_t$.)

6. Thus, the only way for σ to be invalid is that $U_1' = rt_{\overline{ID}_t[i]}U_1$ for some revocation token $rt_{\overline{ID}_t[i]}$ of $RL_{\overline{ID}_t}$. If no such revocation token exists, then σ is valid.

- For the other implication of the *Correctness* property, assume now that σ is valid. Then, steps (3) and (4) above show that the signer used the discrete logarithm of her revocation token, in order to issue σ. Moreover, step (5) above implies that the signer is not revoked.

QED

Proof of Theorem 7.5.2. In this proof, we show how to construct $\mathcal{B}$ using $\mathcal{A}$. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two Gap-Diffie-Hellman groups of prime order q, where $\mathbb{G}_1$ is additive and $\mathbb{G}_2$ is multiplicative. Let $P_0^{(1)} \in \mathbb{G}_1^*$ be a generator of $\mathbb{G}_1$, and ν_2, ν_3, ν_4, ν_5, and ν_6 be positive integers such that $P_0^{(2)} = \nu_2 P_0^{(1)}$, $P_0^{(3)} = \nu_3 P_0^{(1)}$, $P_0^{(4)} = \nu_4 P_0^{(1)}$, $P_0^{(5)} = \nu_5 P_0^{(1)}$, and $P_0^{(6)} = \nu_6 P_0^{(1)}$ are other generators of $\mathbb{G}_1$ (i.e. distinct from one another and from $P_0^{(1)}$). Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be an admissible bilinear pairing with respect to which the BDH problem is assumed to be intractable, and let $(\mathbb{G}_1, q, P_0^{(1)}, aP_0^{(1)}, bP_0^{(1)}, cP_0^{(1)})$ be a tuple for which a, b, and c are unknown to $\mathcal{B}$. $\mathcal{B}$'s goal is to solve the BDH Problem by computing $\hat{e}(P_0^{(1)}, P_0^{(1)})^{abc}$ in polynomial time. To achieve this goal, $\mathcal{B}$ initiates a *Traceability* game with $\mathcal{A}$, using an arbitrary security parameter k.

$\mathcal{B}$ sets $n = poly_1(k)$, $\ell = \log(k)$, $n_\tau = poly_2(k)$, $n_{gs} = poly_3(k)$, $I_0 \in_R \mathbb{Z}_q^*$, $L_1^{(3)} = \mu_3(aP_0^{(1)})$, $L_1^{(4)} = \mu_4(aP_0^{(1)})$, $s_1 \in_R \mathbb{Z}_q^*$, $s_i = \mathcal{H}_5(s_{i-1})$ for $2 \leq i \leq \ell - 1$, and $L_i^{(j)} = s_{i-1}L_{i-1}^{(j)}$ for $2 \leq i \leq \ell$ and $j = 3, 4$. $\mathcal{B}$ also defines a hash function $\mathcal{H}_1^{sim} : \{0,1\}^* \rightarrow \mathbb{G}_2$, over which it has full control (i.e. $\mathcal{A}$ controls the output of $\mathcal{H}_1^{sim}$, given any input). $\mathcal{B}$ defines $\mathcal{M} = \{0,1\}^n$, $\mathcal{S} = \mathbb{G}_1^6 \times \{0,1\}^{n_\tau} \times \mathbb{Z}_q^{*5}$, $pubParams = (g, q, n, \hat{e}, P_0^{(1)}, P_0^{(2)}, P_0^{(3)}, P_0^{(4)}, P_0^{(5)}, P_0^{(6)}, \mathcal{H}_1^{sim}, \mathcal{H}_2, \mathcal{H}_3, \mathcal{H}_4, \mathcal{H}_5, ((L_i^{(j)})_{i=1}^{\ell})_{j=3}^{4})$, and $params = (pubParams, s_0)$, where s_0 is unknown to $\mathcal{B}$, and where $\mathcal{H}_2$ through $\mathcal{H}_5$ are arbitrary one-way functions with appropriate domains and codomains (cf. the *Root Setup* algorithm). $\mathcal{B}$ sends $pubParams$ to $\mathcal{A}$, and keeps $(s_i)_{i=1}^{\ell}$ secret.

The rest of the proof consists of three sections. The *Hash Simulation* section shows how $\mathcal{B}$ simulates $\mathcal{H}_1^{sim}$. The *Attack Games* section explains how $\mathcal{B}$ handles the queries of the attack game. Finally, the *Complexity and Probability* section derives the complexity and probability results of Theorem 7.5.2.

Hash Simulation:

In order to store information about the queries of the *Traceability* game, $\mathcal{B}$ maintains two lists Λ_{mt} and $\Lambda_{\mathcal{H}_1^{sim}}$.

A. *Obtaining Values from* Λ_{mt}:

In order to answer the signature-related queries of the *Traceability* game, $\mathcal{B}$ also maintains a list Λ_{mt}, whose entries are of the form $(\overline{ID}_{(i,t_i)}, z, t, mgt, mt)$.

Given T, $mt_{\overline{ID}_{(i,t_i)}[z]}$ is obtained from Λ_{mt} as follows: if Λ_{mt} contains an entry of the form $(\overline{ID}_{(i,t_i)}, z, t, mgt_{\overline{ID}_{(i,t_i)}[i]}, mt_{\overline{ID}_{(i,t_i)}[i]})$ with $t = T$, then $\mathcal{B}$ returns $mt_{\overline{ID}_{(i,t_i)}[i]}$. Otherwise, $\mathcal{B}$ requests $mt_{\overline{ID}_{(i,t_i)}[z]}$ from $\mathcal{A}$ (unless $\mathcal{A}$ should not know $mgt_{\overline{ID}_{(i,t_i)}[z]}$, in which case $\mathcal{B}$ picks $mgt_{\overline{ID}_{(i,t_i)}[z]} \in_R \mathbb{Z}_q^*$ and computes $mt_{\overline{ID}_{(i,t_i)}[z]} = \left(g^{\mathcal{H}_3(T)}\right)^{mgt_{\overline{ID}_{(i,t_i)}[z]}}$); $\mathcal{B}$ then adds $(\overline{ID}_{(i,t_i)}, z, T, mgt_{\overline{ID}_{(i,t_i)}[z]}, mt_{\overline{ID}_{(i,t_i)}[z]})$ to Λ_{mt}, and returns $mt_{\overline{ID}_{(i,t_i)}[i]}$.

B. *Obtaining Values from* $\Lambda_{\mathcal{H}_1^{sim}}$:

$\Lambda_{\mathcal{H}_1^{sim}}$ consists of entries of the form: $(\overline{ID}_{(i,t_i)}, (p_{(i,d,1)})_{d=1}^{t_i}, (p_{(i,d,3)})_{d=1}^{t_i}$, $(I_{(i,d)})_{d=1}^{t_i}, (I'_{(i,d)})_{d=1}^{t_i}, (I''_{(i,d)})_{d=1}^{t_i}, (I'''_{(i,d)})_{d=1}^{t_i}, (\widetilde{I}_{(i,d)})_{d=1}^{t_i}, (\widetilde{I}'_{(i,d)})_{d=1}^{t_i}, (\alpha_{(i,d)})_{d=1}^{t_i}, (\widetilde{\alpha}_{(i,d)})_{d=1}^{t_i}$, $(S_{\overline{ID}_{(i,d)}})_{d=1}^{t_i}, (E_{\overline{ID}_{(i,d)}})_{d=1}^{t_i}, (\widetilde{S}_{\overline{ID}_{(i,d)}})_{d=1}^{t_i}, (\widetilde{E}_{\overline{ID}_{(i,d)}})_{d=1}^{t_i}, ((\alpha_{\overline{ID}_{(i,d)}[z]})_{d=1}^{t_i})_{z=1}^{n_{gs}}$, $((\widetilde{\alpha}_{\overline{ID}_{(i,d)}[z]})_{d=1}^{t_i})_{z=1}^{n_{gs}}, ((S_{\overline{ID}_{(i,d)}[z]})_{d=1}^{t_i})_{z=1}^{n_{gs}}, ((E_{\overline{ID}_{(i,d)}[z]})_{d=1}^{t_i})_{z=1}^{n_{gs}}, ((\widetilde{S}_{\overline{ID}_{(i,d)}[z]})_{d=1}^{t_i})_{z=1}^{n_{gs}})$, $((\widetilde{E}_{\overline{ID}_{(i,d)}[z]})_{d=1}^{t_i})_{z=1}^{n_{gs}}), ((mgt_{\overline{ID}_{(i,d)}[z]})_{d=1}^{t_i})_{z=1}^{n_{gs}}, ((mt_{\overline{ID}_{(i,d)}[z]})_{d=1}^{t_i})_{z=1}^{n_{gs}}, ((rt_{\overline{ID}_{(i,d)}[z]})_{d=1}^{t_i})_{z=1}^{n_{gs}}$, $((\gamma_{\overline{ID}_{(i,d)}[z]})_{d=1}^{t_i})_{z=1}^{n_{gs}})$, where each $\overline{ID}_{(i,t_i)} = (ID_{(i,0)}, ID_{(i,1)}, \cdots, ID_{(i,t_i)})$ is a leader's ID-tuple in the underlying hierarchy, and where, for $1 \leq d \leq t_i$, $\alpha_{\overline{ID}_{(i,d)}}, p_{(i,d,1)}, p_{(i,d,3)}, I_{(i,d)}$, $I'_{(i,d)}, I''_{(i,d)}, I'''_{(i,d)}, \widetilde{I}_{(i,d)}, \widetilde{I}'_{(i,d)}, \alpha_{\overline{ID}_{(i,d)[z]}}, mgt_{\overline{ID}_{(i,d)}[z]} \in \mathbb{Z}_q^*$, $mt_{\overline{ID}_{(i,d)}[z]}, rt_{\overline{ID}_{(i,d)}[z]} \in SG$, $\gamma_{\overline{ID}_{(i,d)}[z]} \in \mathbb{Z}_q^*$, and $S_{\overline{ID}_{(i,d)}}, S_{\overline{ID}_{(i,d)}[z]}, E_{\overline{ID}_{(i,d)}}, E_{\overline{ID}_{(i,d)[z]}}, \widetilde{S}_{\overline{ID}_{(i,d)}}, \widetilde{S}_{\overline{ID}_{(i,d)}[z]}, \widetilde{E}_{\overline{ID}_{(i,d)}}$, $\widetilde{E}_{\overline{ID}_{(i,d)}[z]} \in \mathbb{G}_1$ are simulated key component values.

Note that, throughout the proof, ID-tuples include ID_0. Throughout the proof, it is also assumed that, before any (non-$\mathcal{H}_1^{sim}$) query is issued by $\mathcal{A}$ with an associated ID-tuple, $\mathcal{B}$ first performs a $\mathcal{H}_1^{sim}$-query on the same ID-tuple. Such a $\mathcal{H}_1^{sim}$-query is answered using $\Lambda_{\mathcal{H}_1^{sim}}$. Let $n_{\mathcal{H}_1}$ denote the number of $\mathcal{H}_1$ queries. Every $\mathcal{H}_1$-query is issued on an ID-tuple whose second component we call the *root branch*, except when the $\mathcal{H}_1^{sim}$-query is issued on ID_0, in which case ID_0 is called the root branch. Let then n_{rb} denote the number of distinct root branches appearing in $\mathcal{H}_1$-query ID-tuples. Every distinct root branch is indexed using an integer between 1 and n_{rb}. Let μ be a uniformly random element of $\{1, \cdots, n_{rb}\}$. If μ is the root branch index of ID_0, then $\mathcal{B}$ terminates the *Traceability* game, and fails to solve the BDH problem. Otherwise, the root branch index of the ID-tuple associated with the forged signature will need to be μ, for $\mathcal{B}$ to win the *Traceability* game. On the contrary, the root branch index of ID-tuples associated with *Signing* and *Compromise* queries preceding the *Forgery* phase must be different from μ, for $\mathcal{B}$ to win the *Traceability* game.

To answer the queries of the *Traceability* game, $\mathcal{B}$ builds $\Lambda_{\mathcal{H}_1^{sim}}$ as follows. Let $\overline{ID}_{(i,t_i)} = (ID_{(i,0)}, ID_{(i,1)}, \cdots, ID_{(i,t_i)})$ be the ID-tuple associated with the i^{th} $\mathcal{H}_1$-query and let y be maximal such that $(ID_{(i,0)}, ID_{(i,1)}, \cdots, ID_{(i,y)}) = (ID_{(j,0)}, ID_{(j,1)}, \cdots, ID_{(j,y)})$, where $(ID_{(j,0)}, ID_{(j,1)}, \cdots, ID_{(j,t_j)})$ is an ID-tuple which is already in $\Lambda_{\mathcal{H}_1^{sim}}$. Then:

- For $0 \leq d \leq y$:

  $p_{(i,d,1)} = p_{(j,d,1)}$, $p_{(i,d,3)} = p_{(j,d,3)}$, $I_{(i,d)} = I_{(j,d)}$, $I'_{(i,d)} = I'_{(j,d)}$, $I''_{(i,d)} = I''_{(j,d)}$, $I'''_{(i,d)} = I'''_{(j,d)}$, $\widetilde{I}_{(i,d)} = \widetilde{I}_{(j,d)}$, $\widetilde{I}'_{(i,d)} = \widetilde{I}'_{(j,d)}$, $\alpha_{\overline{ID}_{(i,d)}} = \alpha_{\overline{ID}_{(j,d)}}$, $\widetilde{\alpha}_{\overline{ID}_{(i,d)}} = \widetilde{\alpha}_{\overline{ID}_{(j,d)}}$, $S_{\overline{ID}_{(i,d)}} = S_{\overline{ID}_{(j,d)}}$, $E_{\overline{ID}_{(i,d)}} = E_{\overline{ID}_{(j,d)}}$, $\widetilde{S}_{\overline{ID}_{(i,d)}} = \widetilde{S}_{\overline{ID}_{(j,d)}}$, $\widetilde{E}_{\overline{ID}_{(i,d)}} = \widetilde{E}_{\overline{ID}_{(j,d)}}$, $\alpha_{\overline{ID}_{(i,d)}[z]} = \alpha_{\overline{ID}_{(j,d)}[z]}$, $\widetilde{\alpha}_{\overline{ID}_{(i,d)}[z]} = \widetilde{\alpha}_{\overline{ID}_{(j,d)}[z]}$, $S_{\overline{ID}_{(i,d)}[z]} = S_{\overline{ID}_{(j,d)}[z]}$, $E_{\overline{ID}_{(i,d)}[z]} = E_{\overline{ID}_{(j,d)}[z]}$, $\widetilde{S}_{\overline{ID}_{(i,d)}[z]} = \widetilde{S}_{\overline{ID}_{(j,d)}[z]}$, $\widetilde{E}_{\overline{ID}_{(i,d)}[z]} = \widetilde{E}_{\overline{ID}_{(j,d)}[z]}$, $mgt_{\overline{ID}_{(i,d)}[z]} = mgt_{\overline{ID}_{(j,d)}[z]}$, $mt_{\overline{ID}_{(i,d)}[z]} = mt_{\overline{ID}_{(j,d)}[z]}$, and $rt_{\overline{ID}_{(i,d)}[z]} = rt_{\overline{ID}_{(j,d)}[z]}$, $\gamma_{\overline{ID}_{(i,d)}[z]} = \gamma_{\overline{ID}_{(j,d)}[z]}$, for $1 \leq z \leq n_{gs}$.

- For $y < d \leq t_i$:

  1. Whatever the value of d, $\mathcal{B}$ performs the following: obtain $mt_{\overline{ID}_{(i,d)}[z]}$ from Λ_{mt}, set $rt_{\overline{ID}_{(i,d)}[z]} = mt_{\overline{ID}_{(i,d)}[z]}$, and pick $\gamma_{\overline{ID}_{(i,d)}[z]} \in_R \mathbb{Z}_q^*$.

     If the membership generating token of $\overline{ID}_{(i,d)}[z]$ is given, then $\mathcal{B}$ sets $mgt_{\overline{ID}_{(i,d)}[z]}$ to this value. Such a membership token is needed to answer *Signing* queries. In the *Traceability* game, $mgt_{\overline{ID}_{(i,d)}[z]}$ must be sent by $\mathcal{A}$.

  2. If $d = 0$: $I_{(i,0)} = I'_{(i,0)} = I''_{(i,0)} = I'''_{(i,0)} = \widetilde{I}_{(i,0)} = \widetilde{I}'_{(i,0)} = \alpha_{(i,0)} = 1$ and $S_{(\overline{ID}_{(i,0)}} = E_{(\overline{ID}_{(i,0)}} = \widetilde{S}_{\overline{ID}_{(i,0)}} = Id_{\mathbb{G}_1}$ (where 1 and the neutral element $Id_{\mathbb{G}_1}$ of $\mathbb{G}_1$ are default values), and, for $1 \leq z \leq n_{gs}$, $\alpha_{\overline{ID}_{(i,0)}[z]} \in_R \mathbb{Z}_q^*$, $mt_{\overline{ID}_{(i,0)}[z]}$

is obtained from Λ_{mt}, $S_{\overline{ID}_0[z]} = I_0(aP_0^{(1)}) + mt_{\overline{ID}_{(i,0)}[z]} I'_0 \nu_2(aP_0^{(1)}) + \alpha_{\overline{ID}_0[z]} L_1^{(3)}$, $E_{\overline{ID}_1} = \frac{I_0 I''_0}{I'_0}(aP_0^{(1)}) + mt_{\overline{ID}_{(i,0)}[z]} I''_0 \nu_2(aP_0^{(1)}) + \widetilde{\alpha}_{\overline{ID}_0[z]} L_1^{(3)}$, $\widetilde{S}_{\overline{ID}_0[z]} = \alpha_{\overline{ID}_0[z]} \nu_4(aP_0^{(1)}) + \widetilde{I}_0 \nu_5(aP_0^{(1)}) + \widetilde{I}_0 mt_{\overline{ID}_{(i,0)}} \nu_6(aP_0^{(1)})$, $\widetilde{E}_{\overline{ID}_0[z]} = \widetilde{\alpha}_{\overline{ID}_0[z]} \nu_4(aP_0^{(1)}) + \frac{\widetilde{I}_0 \widetilde{I}''_0}{\widetilde{I}'_0} mt_{\overline{ID}_{(i,0)}[z]} \nu_5(aP_0^{(1)}) + \widetilde{I}''_0 \nu_6(aP_0^{(1)})$.

3. If $d = 1$:
   - If $\overline{ID}_{(i,t_i)}$ is the first ID-tuple whose root branch index is μ, then, for $1 \leq z \leq n_{gs}$, $I'_{(i,1)}, I''_{(i,1)}, I'''_{(i,1)}, \widetilde{I}_{(i,1)}, \widetilde{I}'_{(i,1)} \in_R \mathbb{Z}_q^*$, $I_{(i,1)} = \alpha_{\overline{ID}_{(i,1)}} = \alpha_{\overline{ID}_{(i,1)}[z]} = 1$, and $S_{\overline{ID}(i,1)} = E_{\overline{ID}(i,1)} = \widetilde{S}_{\overline{ID}(i,1)} = S_{\overline{ID}(i,1)[z]} = E_{\overline{ID}(i,1)[z]} = \widetilde{S}_{\overline{ID}(i,1)[z]} = Id_{\mathbb{G}_1}$ (where 1 and the neutral element $Id_{\mathbb{G}_1}$ of $\mathbb{G}_1$ are default values). If the (public) value $I_{(i,1)}$ is queried, $\mathcal{B}$ assumes that $I_{(i,1)} = b$ and returns the ancillary value $I_{(i,1)}^{(1)} = bP_0^{(1)} = I_1 P_0^{(1)}$, instead of I_1.
     Note that it is possible to construct the public key of any subgroup leader whose root branch index is μ. This is done using $I_{(i,1)}^{(1)}$ with a variant of the recursive algorithm meant to compute A_{t+2}. Under this variant algorithm: $A_1 = bP_0^{(1)} \in \mathbb{G}_1$; $\widetilde{A}_1 = \frac{I''_{(i,1)}}{I'_{(i,1)}} A_1 \in \mathbb{G}_1$; for $1 \leq i < t$, $A_{i+1} = I_{i+1} A_i + I'_{i+1} \widetilde{A}_i$ and $\widetilde{A}_{i+1} = \frac{I_{i+1} I''_{i+1}}{I'_{i+1}} A_i + I''_{i+1} \widetilde{A}_i$.
     This procedure can be extended to construct the public key of any group member $\overline{ID}_t[i]$ whose root branch index is μ. However, $\mathcal{B}$ (and $\mathcal{A}$) cannot use these public keys to verify the validity of signatures which are claimed to have been issued by group members whose root branch index is μ. This is due to the fact that the verifying procedure requires to have $A_t \in \mathbb{Z}_q^*$ (instead of $A_t \in \mathbb{G}_1$, as in the above modified recursive algorithm). This limitation will not impede the effectiveness of our argument, whose success is based on the issuing (by the attacker $\mathcal{A}$) of a valid signature, for an ID-tuple whose root branch index is μ (whether or not this signature is provably valid *using the data provided by* $\mathcal{B}$).
   - Otherwise, $I_{(i,1)}, p_{(i,1,1)}, p_{(i,1,3)} \alpha_{(i,1)}, \widetilde{\alpha}_{(i,1)} \in_R \mathbb{Z}_q^*$, and the other components (except $S_{(i,1)}$, $E_{(i,1)}$, $\widetilde{S}_{(i,1)}$, and $\widetilde{E}_{(i,1)}$) are computed according to the method described in the *Key Extraction* algorithm. Moreover, $S_{\overline{ID}_1} = I_{(i,1)}(aP_0^{(1)}) + I'_{(i,1)} \nu_2(aP_0^{(1)}) + \alpha_{\overline{ID}_1} L_1^{(3)}$, $E_{\overline{ID}_1} = \frac{I_{(i,1)} I''_{(i,1)}}{I'_{(i,1)}}(aP_0^{(1)}) + I''_{(i,1)} \nu_2(aP_0^{(1)}) + \widetilde{\alpha}_{\overline{ID}_1} L_1^{(3)}$, and $\widetilde{S}_{\overline{ID}_1} = \alpha_{\overline{ID}_1} \nu_4(aP_0^{(1)}) + \widetilde{I}_{(i,1)} \nu_5(aP_0^{(1)}) + \widetilde{I}'_{(i,1)} \nu_6(aP_0^{(1)})$, $\widetilde{E}_{\overline{ID}_1} = \widetilde{\alpha}_{\overline{ID}_1} \nu_4(aP_0^{(1)}) + \frac{\widetilde{I}_{(i,1)} \widetilde{I}''_{(i,1)}}{\widetilde{I}'_{(i,1)}} \nu_5(aP_0^{(1)}) + \widetilde{I}''_{(i,1)} \nu_6(aP_0^{(1)})$.

4. Otherwise (i.e. if $2 \leq d \leq t_i$), $I_{(i,d)}, p_{(i,d,1)}, p_{(i,d,3)}\alpha_{(i,d)}, \widetilde{\alpha}_{(i,d)} \in_R \mathbb{Z}_q^*$, and the other components (including the I_i's, $S_{(i,d)}$, $E_{(i,d)}$, $\widetilde{S}_{(i,d)}$, and $\widetilde{E}_{(i,d)}$) are computed according to the method described in the *Key Extraction* algorithm.

- $\mathcal{B}$ adds the entry $(\overline{ID}_{(i,t_i)}, (p_{(i,d,1)})_{d=1}^{t_i}, (p_{(i,d,3)})_{d=1}^{t_i}$, $(I_{(i,d)})_{d=1}^{t_i}, (I'_{(i,d)})_{d=1}^{t_i}, (I''_{(i,d)})_{d=1}^{t_i}, (I'''_{(i,d)})_{d=1}^{t_i}, (\widetilde{I}_{(i,d)})_{d=1}^{t_i}, (\widetilde{I}'_{(i,d)})_{d=1}^{t_i}, (\alpha_{(i,d)})_{d=1}^{t_i}, (\widetilde{\alpha}_{(i,d)})_{d=1}^{t_i}$, $(S_{\overline{ID}_{(i,d)}})_{d=1}^{t_i}, (E_{\overline{ID}_{(i,d)}})_{d=1}^{t_i}, (\widetilde{S}_{\overline{ID}_{(i,d)}})_{d=1}^{t_i}, (\widetilde{E}_{\overline{ID}_{(i,d)}})_{d=1}^{t_i}, ((\alpha_{\overline{ID}_{(i,d)}[z]})_{d=1}^{t_i})_{z=1}^{n_{gs}}$, $((\widetilde{\alpha}_{\overline{ID}_{(i,d)}[z]})_{d=1}^{t_i})_{z=1}^{n_{gs}}, ((S_{\overline{ID}_{(i,d)}[z]})_{d=1}^{t_i})_{z=1}^{n_{gs}}, ((E_{\overline{ID}_{(i,d)}[z]})_{d=1}^{t_i})_{z=1}^{n_{gs}}, ((\widetilde{S}_{\overline{ID}_{(i,d)}[z]})_{d=1}^{t_i})_{z=1}^{n_{gs}})$, $((\widetilde{E}_{\overline{ID}_{(i,d)}[z]})_{d=1}^{t_i})_{z=1}^{n_{gs}}), ((mgt_{\overline{ID}_{(i,d)}[z]})_{d=1}^{t_i})_{z=1}^{n_{gs}}, ((mt_{\overline{ID}_{(i,d)}[z]})_{d=1}^{t_i})_{z=1}^{n_{gs}}, ((rt_{\overline{ID}_{(i,d)}[z]})_{d=1}^{t_i})_{z=1}^{n_{gs}}$, $((\gamma_{\overline{ID}_{(i,d)}[z]})_{d=1}^{t_i})_{z=1}^{n_{gs}})$ to the list $\Lambda_{\mathcal{H}_1^{sim}}$.

Attack Game:

- By generating or reusing the appropriate entries of $\Lambda_{\mathcal{H}_1^{sim}}$, $\mathcal{B}$ answers the *Public Key*, the *Signing*, and *Compromise* queries of the *Traceability* game, under the following constraint: if the root branch index of the i^{th} ID-tuple of any *Signing* or *Compromise* query is μ, then $\mathcal{B}$ terminates the *Traceability* game and fails to solve the BDH problem.

- Once the *Query* phase is over, the *Forgery* phase starts. $\mathcal{A}$ issues a target ID-tuple $\overline{ID}_t^* = (ID_0, ID_1, \cdots, ID_t)$, a message m^*, and a signature σ^*, such that: either (a) the *Opening* algorithm fails on input $(RL_{\overline{ID}_t^*}, m^*, \sigma^*, rt_{\overline{ID}_t^*})$, or (b) on input $(RL_{\overline{ID}_t^*}, m^*, \sigma, rt_{\overline{ID}_t^*})$, the *Opening* algorithm traces σ^* to a user identifier $\overline{ID}_t^*[i]$, such that: (b1) σ^* is valid with respect to $RL_{\overline{ID}_t^*}$ and m^*; (b2) $\overline{ID}_t^*[i]$ is not revoked; (b3) $\mathcal{A}$ did not issue a *User Key Compromise* query on $\overline{ID}_t^*[i]$; (b4) $\mathcal{A}$ did not issue a *Signing* query on m^* with $\overline{ID}_t^*[i]$. (Note that condition (b2) is implied by (b1), since the verification algorithm checks the revocation token of each revoked user.)

  If $\overline{ID}_t^*$'s root branch index is not μ, then $\mathcal{B}$ terminates the *Traceability* game and fails to solve the BDH problem. Otherwise, let $\sigma^* = (U_1, U'_1, U_2, U'_2, U_3, U'_3, U_4, U'_4, V, \widetilde{V}, W_1, W_2, \beta, \kappa, \theta, \varpi)$.

  If σ^* is valid, then note that the zero-knowledge proof component of the verification algorithm ensures that one of the revocation tokens associated with $\overline{ID}_t$'s group satisfies the condition stipulated by the *Opening* algorithm. Therefore, the *Opening* algorithm does not fail on input $(RL_{\overline{ID}_t^*}, m^*, \sigma^*, rt_{\overline{ID}_t^*})$. If σ^* is valid, note also that $V = (\prod_{i=1}^{t+2} s_{i-1})(A_{t+2}P_0^{(1)} + B_{t+2}P_0^{(2)} + F_{t+2}P_0^{(3)})$ and $\widetilde{V} = (\prod_{i=1}^{t+2} s_{i-1})(F_{t+2}P_0^{(4)} + C_{t+2}P_0^{(5)} + D_{t+2}P_0^{(6)})$, where $A_{t+2}, B_{t+2}, C_{t+2}, D_{t+2}$,

$F_{t+2} \in \mathbb{Z}_q^*$ and $A_{t+2} = I_1\beta_r\Upsilon + \Xi$, such that Υ, Ξ, B_{t+2}, C_{t+2}, D_{t+2} are multivariate polynomials which have no term with I_1 and I_1^{-1}. Consequently, $\mathcal{B}$ is able to construct these multivariate polynomials. Moreover, $\mathcal{B}$ knows s_i for $1 \leq i \leq \ell - 1$. Furthermore, β enables $\mathcal{B}$ to compute β_r^{-1}, where $U_1 = \beta_r P_0^{(1)}$ (since σ^* is valid and $\mathcal{B}$ knows $\gamma_{\overline{ID}_t^*}$.) Hence, $\mathcal{B}$ can compute $\hat{e}(\beta_r^{-1}cP_0^{(1)}, \beta_r I_1 s_0 P_0^{(1)}) = \Theta$, and, if $Z = \beta_r^{-1}cP_0^{(1)}$, $X = \hat{e}(L_{t+2}^{(4)}, (\Xi Z + B_{t+2}\nu_2 Z)^{-1}$,
$Y = \hat{e}(L_{t+2}^{(3)}, (C_{t+2}\nu_5 Z + D_{t+2}\nu_6 Z)$, then

$$\Theta = \left(\hat{e}(\nu_4 Z, V) \cdot \hat{e}(\nu_3 Z, \widetilde{V})^{-1} \cdot X \cdot Y\right)^{(\Upsilon)^{-1}}.$$

But $s_0 = a$ and $I_1 = b$. So $\Theta = \hat{e}(P_0^{(1)}, P_0^{(1)})^{abc}$ is the solution of the BDH problem. Therefore, $\mathcal{B}$ can (and does) output Θ, as the solution of the BDH problem.

Thus, $\mathcal{B}$ solves the BDH problem via the *Traceability* game if and only if the following conditions are satisfied: (1) μ is not the root branch index of ID_0; (2) $\mathcal{A}$ issues no *Compromise* or *Signing* queries with an ID-tuple whose root branch index is μ; (3) $\mathcal{A}$ chooses a target ID-tuple whose root branch index is μ.

Complexity and Probability:
In order to compute the probability that $\mathcal{B}$ solves the BDH problem, let δ_μ be the proportion of ID-tuples whose root branch index is μ (among all possible ID-tuples, and *without* considering ID_0 as a possible root branch). Assume that, in the *Traceability* game, $\mathcal{A}$ issues a total of n_{pk} *Public Key* queries, n_{sig} *Signing* queries, and n_{comp} *Compromise* queries.

The probability that $\mathcal{A}$ asks to be challenged on an ID-tuple whose root branch index is μ is δ_μ. If $\varepsilon(k)$ is the advantage of $\mathcal{A}$ in the *Traceability* game, then the probability that $\mathcal{B}$ solves the BDH problem is $\varepsilon(k)\frac{n_{rb}-1}{n_{rb}}(1-\delta_\mu)^{n_{sig}+n_{comp}}\delta_\mu$, where $\frac{n_{rb}-1}{n_{rb}}$ is the probability that μ is not the root branch index of ID_0. Now, $\delta_\mu \leq \delta$, since δ is the largest proportion of ID-tuples having a common root branch. Moreover, since all proportions of ID-tuples having a common root branch are assumed to be non-negligible quantities, there exists a polynomial pol over $\mathbb{N}$ such that $\delta_\mu \geq \frac{1}{pol(k)}$. Hence, $\varepsilon(k)\frac{n_{rb}-1}{n_{rb}}(1-\delta_\mu)^{n_{sig}+n_{comp}}\delta_\mu \geq \varepsilon(k)\frac{n_{rb}-1}{n_{rb}}(1-\delta)^{n_{sig}+n_{comp}}\frac{1}{pol(k)}$. Now, since $\varepsilon(k)$ is non-negligible, so is $\varepsilon(k)\frac{n_{rb}-1}{n_{rb}}(1-\delta)^{n_{sig}+n_{comp}}\frac{1}{pol(k)}$. Consequently, $\mathcal{B}$ solves the BDH problem with non-negligible probability at least

$$\varepsilon(k)\frac{n_{rb}-1}{n_{rb}}(1-\delta)^{n_{sig}+n_{comp}}\frac{1}{pol(k)}.$$

Moreover, since each query of the *Traceability* game requires $\mathcal{B}$ to make a polynomial number of both point selections in $\mathbb{G}_1$ and $\mathbb{Z}_q^*$, and point multiplications in $\mathbb{G}_1$,

it follows that $\mathcal{B}$ solves the BDH problem in time $O(\tau)$ where τ is the running time of $\mathcal{A}$ in the *Traceability* game.

QED.

Proof of Theorem 7.5.3. Let b be randomly chosen in $\{0,1\}$. b is fixed for the remainder of the proof. Let a be a random generator of a subgroup SG of $\mathbb{Z}_q^*$ in which the discrete logarithm problem is intractable. An entity $\mathcal{B}$ is asked to find an unknown $c \in_R \mathbb{Z}_q^*$, given $(a^c P_0^{(1)}, a, P_0^{(1)})$. The proof of this theorem consists of showing how $\mathcal{A}$ can be used to help $\mathcal{B}$ solve the *DDL* problem.

$\mathcal{B}$ runs the *Root Setup*, as specified by the HIBGS scheme, except for the following two changes: (1) $P_0^{(2)} = \nu_2 P_0^{(1)}$, $P_0^{(5)} = \nu_5 P_0^{(1)}$, and $P_0^{(6)} = \nu_6 P_0^{(6)}$, for some $\nu_2, \nu_5, \nu_6 \in_R \mathbb{Z}_q^*$; and (2) a simulated hash function $\mathcal{H}_2^{sim}$ is used instead of $\mathcal{H}_2$ ($\mathcal{B}$ controls the output of $\mathcal{H}_2^{sim}$.) $\mathcal{B}$ then computes the message space $\mathcal{M} = \{0,1\}^n$, the signature space $\mathcal{S} = \mathbb{G}_1^6 \times \{0,1\}^{n_\tau} \times \mathbb{Z}_q^{*5}$, $pubParams = (g, q, n, \hat{e}, P_0^{(1)}, P_0^{(2)}, P_0^{(3)}, P_0^{(4)}, P_0^{(5)}, P_0^{(6)}, \mathcal{H}_1, \mathcal{H}_2^{sim}, \mathcal{H}_3, \mathcal{H}_4, \mathcal{H}_5, ((L_i^{(j)})_{i=1}^{\ell})_{j=3}^{4})$, and $params = (pubParams, s_0)$. Without loss of generality, a validity period T is fixed for all private keys. Let $\widetilde{a} \in_R \mathbb{Z}_q^*$, $Y_1 = a^c P_0^{(1)}$, and $Y_2 = \widetilde{a} Y_1$.

Simulation of $\mathcal{H}_2$:

$\mathcal{H}_2^{sim}$ is used to simulate $\mathcal{H}_2$, during an *Anonymity* game initiated by $\mathcal{B}$ against $\mathcal{A}$. To answer $\mathcal{H}_2$ queries, $\mathcal{B}$ uses a list $\Lambda_{\mathcal{H}_2^{sim}}$, whose entries have the form (x, h_x), where $x \in \{0,1\}^*$ and $h_x \in \mathbb{Z}_q^*$. More precisely, x is either of the form "y", of the form "$||y||$", or of the form $(\varphi||y||\widetilde{y})$, where $\varphi, y, \widetilde{y} \in \{0,1\}^*$. At the beginning of the game, $\mathcal{B}$ places (F, h_F) in $\Lambda_{\mathcal{H}_2^{sim}}$, where $\varpi^* \in_R \mathbb{Z}_q^*$, $\xi^* \in_R \mathbb{Z}_q^*$, $F =$ "$\varpi^* || (a^{\mathcal{H}_3(T)})^{\xi^*} Y_1 || ((a\widetilde{a})^{\mathcal{H}_3(T)})^{\xi^*} \nu_2 Y_2$", and $h_F \in_R \mathbb{Z}_q^*$. During the game, $\mathcal{B}$ proceeds as follows, given an input X: (1) If $\Lambda_{\mathcal{H}_2^{sim}}$ contains an entry of the form (X, h_X), then $\mathcal{B}$ returns h_X has the output of $\mathcal{H}_2(X)$; (2) Otherwise, $\mathcal{B}$ picks $h_X \in_R \mathbb{Z}_q^*$, adds (X, h_X) to $\Lambda_{\mathcal{H}_2^{sim}}$, and returns h_X as the output of $\mathcal{H}_2(X)$.

Attack Game:

$\mathcal{B}$ then initiates an *Anonymity* game against $\mathcal{A}$, and proceeds as follows.

- *Query Phase*:

  $\mathcal{B}$ uses the appropriate algorithms of the HIBGS scheme, in order to answer the *Public Key*, the *Compromise*, the *Signing*, and the *Opening* queries. For the *Revocation* queries, $\mathcal{B}$ adds the corresponding revocation token to the subgroup's revocation vector.

- *Challenge Phase*:

  Once the *Query* phase is over, the *Challenge* phase starts. $\mathcal{A}$ outputs a target subgroup leader's ID-tuple $\overline{ID}_t^*$, a message m^*, and two indices i_0 and i_1

(where $1 \leq i_0, i_1 \leq n_{gs}$). With respect to $\overline{ID}_t^*$, $\mathcal{A}$ must have issued no *Leader Compromise*, *User Compromise*, or *User Revocation* queries for either index.

Recall that $Y_1 = a^c P_0^{(1)}$ and $Y_2 = \widetilde{a} Y_1$. $\mathcal{B}$ proceeds as follows:

1. Using $(s_i)_{i=0}^t$, $U_1 = Y_1$, $U_1' = Y_2$, $U_2 = \nu_2 U_1$, $U_2' = \nu_2 U_1'$, $U_5 = \nu_5 U_1$, $U_5' = \nu_5 U_1'$, $U_6 = \nu_6 U_1$, $U_6' = \nu_6 U_1'$, and the procedure presented in the *Verifying* algorithm to construct both $\rho_t^{(1)}$ and $\rho_t^{(2)}$ (instead of the mechanism described in the *Signing* algorithm), $\mathcal{B}$ computes $\underline{S}_{\overline{ID}_t[i]}$, $\underline{E}_{\overline{ID}_t[i]}$, $\underline{\widetilde{S}}_{\overline{ID}_t[i]}$, and $\underline{\widetilde{E}}_{\overline{ID}_t[i]}$.
2. Then, $\mathcal{B}$ picks $\beta^* \in_R \mathbb{Z}_q^*$, and computes $W_1 = T$, $W_2 = a$, $\kappa^* = \mathcal{H}_2(F)$, and $\theta^* = \xi^*$;
3. picks $\alpha_{m^*} \in_R \mathbb{Z}_q^*$, and computes:
   $I_{m^*} = \mathcal{H}_4(m^*)$, $I'_{m^*} = \mathcal{H}_7(I_{m^*})$,
   $V = s_{t+1}(I_{m^*} \underline{S}_{\overline{ID}_t[i]} + I'_{m^*} \underline{E}_{\overline{ID}_t[i]}) + \alpha_{m^*} L_{t+2}^{(3)}$,
   $\widetilde{V} = s_{t+1}(I_{m^*} \underline{\widetilde{S}}_{\overline{ID}_t[i]} + I'_{m^*} \underline{\widetilde{E}}_{\overline{ID}_t[i]}) + \alpha_{m^*} L_{t+2}^{(4)}$;
4. $\mathcal{B}$ sends $\sigma^* = (U_1, U_1', U_2, U_2', U_5, U_5', U_6, U_6', V, \widetilde{V}, W_1, W_2, \beta^*, \kappa^*, \theta^*, \varpi^*)$ to $\mathcal{A}$. By definition, σ^* is set to be issued on behalf of i_b, where b is the bit (unknown to $\mathcal{B}$) that was fixed at the beginning of the proof.

- *Restricted Phase*:

  This part of the game is handled as in the *Queries* phase, under the restrictions stipulated by the game.

- *Output Phase*:

  $\mathcal{A}$ finally issues a guess bit b', which $\mathcal{B}$ compares to b to determine whether $\mathcal{A}$ wins the game.

Complexity and Probability:
The only way for $\mathcal{A}$ to have a non-negligible advantage against $\mathcal{B}$ in the game is that $\mathcal{A}$ finds out that the value κ^* in σ^* was not properly generated (according to the *Signing* algorithm). This happens if and only if: (1) $\mathcal{A}$ computes c from σ^*; (2) $\mathcal{A}$ recomputes the valid value of σ^*. But the valid σ^* is computed if and only if $\mathcal{A}$ computes $\mathcal{H}_2^{sim}(\varphi||y||\widetilde{y})$, where $\varphi = \mathcal{H}_2(c)$ and $y, \widetilde{y} \in \{0,1\}^*$. Thus, $\mathcal{A}$ has non-negligible advantage against $\mathcal{B}$ if and only if $\mathcal{A}$ computes $\mathcal{H}_2(c)$ and $\mathcal{B}$ thereby learns c by examining the entries of $\Lambda_{\mathcal{H}_2^{sim}}$.

Hence, if $\varepsilon(k)$ is $\mathcal{A}$'s advantage in the *Anonymity* game, then $\mathcal{B}$ solves the DDL problem with advantage at least $\varepsilon(k)$.

Moreover, if $\mathcal{B}$ learns the solution of the DDL problem, then this solution is found in $O(\tau)$ steps, since each query of the *Query*, the *Challenge*, and the *Restricted*

Query phases requires $\mathcal{B}$ to make a polynomial number of point selections, point multiplications in $\mathbb{G}_1$, and pairings.

QED.

7.6 Conclusion

The aim of this chapter was to propose a cryptographic scheme suitable for hierarchical anonymous authentication (HAA). HAA enables routing and filtering, in large and hierarchically-structured partially-hidden networks. Moreover, HAA supports privacy-enhanced access control in settings in which the hierarchical subgroup structure of a large group is public, and one seeks to determine whether a user belongs to a given subgroup without compromising the anonymity of this user.

We devised a threat model for a class of hierarchical group signature schemes that are suitable for HAA, and we described a hierarchical ID-based group signature (HIBGS) scheme which is provably secure, in the random oracle model, under the proposed threat model. Important features of the proposed scheme include its efficiency and its security. In terms of computational efficiency: the scheme exempts signature verifiers from the need to obtain certified copies of subgroup public keys; moreover, the proposed scheme requires the least number of pairing computations, for signing and verifying procedures (compared with other pairing-based group signature schemes); furthermore, the scheme's signing and verifying costs grow only logarithmically with respect to the number of subgroups in the hierarchy. In terms of security: the scheme provides a mechanism for signature verifiers to locally assess the revocation status of signers (and thereby the validity of group signatures); moreover, the scheme offers the guarantee that non-compromised subgroup members cannot be impersonated by their subgroup leaders.

One limitation of the proposed scheme is the fact that the computational cost of signature verification grows linearly with respect to the size of revocation lists. This is therefore an area of future research. Extensions of our scheme include the design of threshold and forward-secure ID-based group signature/signcryption schemes.

Chapter 8

Conclusion

Our primary motivation for this dissertation was to investigate the use of identity-based cryptography (IBC) in large user communities. In particular, we focused on the following three fundamental questions.

- *Is it possible to generate efficiently-usable private keys from non-repeatable (e.g. biometric) information ?*
- *Is it possible to authenticate the sender of encrypted information ?*
- *Is it possible to determine whether a user belongs to a publicly known subset of a group, while preserving the anonymity of this user from all but a designated party ?*

In this concluding chapter, the dissertation's main contributions are summarized, and extensions of our work are discussed.

8.1 Contributions

Efficiently-Usable Keys From Non-Repeatable Information

Our interest in the first fundamental question is motivated by its relevance to both biometric-based access control and biometric-based authentication. However, any solution to this question can be applied to role-based access control.

Since biometric readings are non-repeatable, the cryptographic keys which are derived from biometric features are not easy to generate. Moreover, even if such keys are generated, they should enable fast cryptographic operations (e.g. fast decryption and fast digital signing). Furthermore, the encryption algorithm associated with these keys should handle the empirically-varying precision of reading devices.

In order to address these requirements, we proposed a threshold attribute-based encryption (thABE) providing the following benefits. First, the proposed thABE scheme (New-thABE) features an efficient decryption procedure, regardless of the number d of attributes required for decrypting a given ciphertext. (Our scheme requires only two pairings, compared to d pairings in the case of the previously most efficient thABE scheme.) Second, New-thABE enables encryptors to specify, at encryption time, any threshold number of attributes required to decrypt computed ciphertexts. (This provides flexibility with respect to the precision of biometric reading devices, but was not possible with previous schemes.) Third, New-thABE allows to use multiple ad-hoc sets of attributes. (This feature was not provided by previous thABE schemes.) Moreover, when all the attributes of a given set are required to decrypt a given ciphertext, New-thABE can be optimized to produce constant-size ciphertexts.

When thABE schemes are used support Daugman's iris recognition algorithm, 249 attributes must be handled. Hence, compared with Sahai and Waters'thABE scheme, New-thABE saves 249-2=247 pairings at each decryption.

Encrypted Message Authentication

Our interest in encrypted message authentication (EMA) is motivated by the threats of denial of service attacks in settings where untrusted users flood networks or public servers with encrypted random data. We sought to design a scheme that allows network routers to determine whether encrypted messages are sent by trusted users without having to compromise the confidentiality of these messages.

In order to support EMA in large user communities, we presented a hierarchical ID-based signcryption (HIBSC) scheme featuring both forward-security (FS) and public ciphertext authenticity (PCA). No such scheme had previously been described. The proposed HIBSC scheme is based on a proposed HIBE and a proposed HIBS scheme some of whose applications were also investigated. These application include forward-secure hierarchical ID-based encryption (fsHIBE), reverse fsHIBE, and append-only signature (AOS). fsHIBE limits the impact of decryption key compromise, and can be used to support both user-based and role-based access control. Reverse fsHIBE can be used to make restricted information quickly available to a large and distributed user community, after a predetermined time. AOS supports efficient and convenient resource sharing in large and distributed user communities, but AOS scheme can also be used to provide path authenticity in intra-domain Internet routing.

Anonymous Authentication

The third issue which this dissertation deals with is hierarchical anonymous authentication (HAA). HAA enables routing and filtering in large and hierarchically-structured partially-hidden networks. Moreover, HAA increase resistance against network mapping (NM) attacks, and various NM-derived network-based attacks. Alternatively, HAA can be used to support privacy-enhanced access control.

To support HAA, we proposed a provably-secure hierarchical ID-based group signature (HIBGS) scheme. No such scheme had previously been described.

8.2 Future Work

Many extensions of our work are conceivable.

Threshold ABE

With regards to thABE, it remains an open question whether it is possible to design a thABE scheme featuring constant-size ciphertexts regardless of both the number of target attributes and the specified threshold values. Likewise, no thABE scheme is known which simultaneously provides constant-time encryption and dynamically-specifiable threshold values. (While our thABE scheme's *Encryption* procedure requires only one pairing and a constant number multiplications in $\mathbb{G}_1$, its required number of multiplications and exponentiations in $\mathbb{Z}_q^*$ is linear with respect to threshold values.)

Another possibility is to investigate the topic of threshold attribute-based signature (thABS). (It should be possible to derive an efficient signature scheme from the key generation scheme of our proposed thABE.)

Empirical studies should also be conducted to determine whether threshold attribute-based cryptographic schemes are more suitable than other fuzzy ID-based schemes. In other words, is the set-overlap metric empirically better or worse than other metrics?

Furthermore, concerning the use of thABE to support RBAC, it remains an open question whether one can design a thABE scheme which allows equally-privileged users from two different domains to both be able to decrypt the same ciphertext. This would allow companies who have common identifiers for their employees'roles to issue ciphertexts which employees from both companies could decrypt.

HIBE and HIBS

With regards to HIBE and HIBS, one could build a mediated version of NewHIBE and NewHIBS. One could also investigate the applicability of NewAOS to BGP, in comparison with BBG_2-AOS. Another possibility is to investigate the use of NewAOS to applications in which users make decisions based on trust claims issued by other users. While, in the context of human relationships, trust is not necessarily transitive, converging evidence of the reliability of trust claims is often sufficient to make informed decisions. Suppose, then that AOS signatures are used to issue claims on the trust of users by other users, then HIBS-based AOS signatures would enable efficient verification of these trust claims (since HIBS-based AOS signature verification only requires the public key of one user).

Hierarchical ID-Based Signcryption

Concerning HIBSC, an open question is whether one can design a hierarchical ID-based signcryption scheme featuring ciphertext anonymity (i.e. the property whereby no information concerning both the sender and the recipient of any given signcryptext is revealed to an active third party.) NewHIBSC is the first HIBSC scheme which does not reveal the hierarchical level of intended recipients of signcryptexts. However, due to its PCA feature, NewHIBSC does not hide the hierarchical level of the signer. When signcryption is used in applications for which denial of service attacks are not a practical threat, then a HIBSC scheme with ciphertext anonymity would be useful, because it would provide sender anonymity.

Hierarchical ID-Based Group Signature

Our proposed HIBGS scheme has a signature verification cost which grows linearly with respect to the size of revocation lists. When users are frequently revoked from subgroups, this feature is not desirable. Hence, one could investigate the possibility of designing a group signature scheme with VLR in which the search for revocation tokens would be sublinear (e.g. logarithmic) in the number of revoked users.

Other Directions

One could also investigate the formal security of our schemes in other frameworks than the random oracle model. Such an extension could address the growing concern related to the discovery of weaknesses in widely used hash functions. One could also explore the possibility of designing hierarchical ID-based blind and ring signature schemes [Cha82, ZK02]. Another research avenue consists of implementing our proposed schemes. This would raise various questions concerning the choice of finite

fields, elliptic curves, and bilinear functions which efficiently implement our theoretical protocols. Finally, it remains unclear whether one can build an encryption scheme whose decryption keys are not escrowed, and whose encryption algorithm only requires to obtain the identifier of an intended ciphertext recipient.

Appendix A

Review of Certificate-Based Encryption Schemes for Object-Based Access Control

This Appendix reviews research on certificate-based public-key encryption schemes proposed for object-based access control, in large user communities. These schemes are meant to support (variants of) *role-based access control* (RBAC).

§A.1 presents a basic implementation of RBAC, and §A.2 describes two classes of certificate-based encryption scheme which have proposed to support RBAC. §A.3 concludes the Appendix.

A.1 Basic RBAC Implementation

RBAC can be used as follows: users of a system interact with an access manager (*AM*) which enforces predefined access control policies. Thus, the *AM* verifies users' credentials and, when the credentials are both trustworthy and sufficient, the *AM* sends cleartext copies of confidential information to users. One disadvantage of such a pure-policy-based implementation of *RBAC* is that the *AM* must perform the role-based credential verification at each confidential information request. Another disadvantage is the absence of protection against eavesdropping of protected information sent by the *AM* to users[1].

[1]Note that the *AM* is fully trusted (since it can access and alter each message sent to users). In selected applications, this may be a disadvantage.

A.2 Certificate-based Support of RBAC

Public-key cryptography can be used to support RBAC. One way to do this is to use the aforementioned KRA scheme. Another way is to use hierarchical certificate-based cryptographic schemes. The latter approach consists of associating each role with a public-private key pair, and of enabling each user of a role to generate the private keys of its hierarchical role descendants only.

Suppose, for instance, that the users of an information system are divided in disjoint sets $U_1, U_2, \cdots, U_n$. Each set U_i $(1 \leq i \leq n)$ is called a *role*, and gathers users who have the same security privileges to access information managed by system. Assume also that roles are partially ordered, using a relation denoted by the symbol $\leq$. $U_i \leq U_j$ is written to denote the fact that U_j has a higher *security clearance* than U_i, i.e. that all the users of U_j have all the security privileges of U_i's users. The expressions "U_i *is a descendant of* U_j" and "U_j *is an ancestor of* U_i" are also used to denote the fact that $U_i \leq U_j$. In order to cryptographically support role-based access control, each role U_i is assigned a key K_i, and the problem of designing a scheme whereby each key K_i can be dynamically generated form a key K_j if and only if $U_i \leq U_j$ is known as the *multilevel data security key management* problem. To tackle this problem, two categories of certificate-based cryptographic schemes have been proposed.

A.2.1 Algebraic Schemes Supporting Hierarchies

The first class of certificate-based schemes which address the multilevel data security problem involves the use of algebraic means.

In 1983, Akl and Taylor [AT82, AT83] described the first solution to the said problem, based on an (algebraic) cryptosystem called RSA[2] [RSA78]. The idea was to assign a special public value to each node of a partially ordered set of users, in such a way that these values can be used to encrypt keys. More precisely, the public value of role U_j is used to encrypt the key K_i, whenever $U_i \leq U_j$. While very elegant, this initial scheme of Akl and Taylor has two practical shortcomings: first, the special public values are very large integers (which must be computed and stored); second, any addition of role in a hierarchy requires to change the public values of all its non-descendant roles. While MacKinnon et al. [MA83, MTMA85] optimized the assignment public values based on carefully chosen criteria, the two shortcomings of Akl and Taylor's scheme have remained mostly unaddressed.

In 1990, Harn and Lin [HL90] suggested a variant of Akl and Taylor's scheme which offers two improvements: first, the order of magnitude of public values associated

[2]Note that, in large and distributed user communities, RSA is used with public-key certificates.

with roles is reduced; second, the addition of a role only requires to recompute the public values of its ancestors. While the storage improvement is substantial, the computational cost of role addition remains high, especially for deep and shallow hierarchies.

In 1993, Liaw et al. [LWL93] proposed a novel scheme[3] whereby, one-way hash functions and polynomial interpolation allow to derive any role key from a combination of any of this role's parent's[4] key and all the public values associated with this parent's children[5]. However, this scheme suffered from a security weakness which was later fixed by Hwang [Hwa99].

In 1997, Lin [Lin97] devised a method (based on RSA) with the following characteristics: (a) one representative of each role can generate a key for this role (instead of being assigned a role key); (b) each user having the key K_j of a role U_j can derive the key K_i of any descendant role U_i of U_j, without any need to modify the keys or public values associated with any other role. However, this scheme requires that each user interacts with a central server which stores confidential information concerning each role. This shortcoming is cumbersome whenever user autonomy is desired.

In 2002, based on the difficulty of the discrete logarithm problem in certain algebraic groups, Shen and Chen [SC02] proposed a scheme[6] which improves Liaw et al.'s [LWL93] scheme, by reducing (though not drastically) the computational cost of role addition. The same year, Ray et al. [RRN02] devised a scheme (based on RSA) which handles the multilevel data security problem in special classes of hierarchies characterized by non-transitive exceptions. In other words, Ray et al.'s scheme allows to generate keys in such a way that any role U_j such that $U_i \leq U_k \leq U_j$ is not able – unlike U_k – to generate the key K_i of U_i. Moreover, the scheme was designed in such a way that role addition only affects the keys of its descendants, and role deletion has no key-related computational cost.

In 2003, Lin et al. [LHC03] suggested a scheme which handles hierarchies characterized by the fact that some roles and their children can, exceptionally, generate each other's keys. (This is also known as the *anti-symmetrical hierarchical exception* case.) Shortly after, Hwang and Yang [HY03] described a scheme (based on RSA) with significant computational and storage benefits, when partially ordered hierarchies are considered. However, the scheme handles neither non-transitive nor anti-symmetrical hierarchical exceptions.

---

[3] As its predecessors, this scheme would also require the use of public-key certificates in large and distributed user communities.

[4] By *parent* of a role U_i, we denote any ancestor U_j, such that the only ancestor U_k such that $U_i \leq U_k \leq U_j$ is either U_i or U_j.

[5] By *children* of a role U_i, we denote any role for whom U_i is a parent.

[6] In large and distributed user communities, this scheme would require the use of public-key certificates.

Recently, Huang and Chang [HC04] described a scheme (based on RSA) which handles partially ordered hierarchies, and requires users to obtain up-to-date role keys, from a central authority, in order to be able to decrypt ciphered documents. More precisely, the deployment time of a system is divided into periods $t_1, \cdots, t_z$, and, in order to generate the key of any of its descendants, at time t of period $t_a \in \{t_1, \cdots, t_z\}$, each user must obtain the key corresponding to a role of his/hers which is valid during t_a. Such a requirement aims to reflect the fact that users are typically given privileges for a finite amount of time.

Finally, very recently, De Santis et al. [dSFM04] described an algebraic scheme which generalizes Akl and Taylor's method to arbitrary hierarchies. Such hierarchies can be represented by directed graphs whose edges correspond to roles, such that a role U_i is a descendant of another role U_j if and only if there is a path from U_j to U_i. However, the storage cost of the public values associated with each role is relatively high, and the scheme involves computationally intensive algebraic techniques.

A.2.2 One-Way Functional Schemes Supporting Hierarchies

The other class of certificate-based schemes which have been proposed to address the multilevel data security problem involves the use of one-way functions (and no heavy algebraic technique). One-way functions have been well studied in cryptography, and are generally faster than the aforementioned algebraic techniques.

In 1988, Sandhu [San88] proposed a very elegant and efficient scheme which handles tree-shaped hierarchies. If U_i is the child of role U_j, and if K_i and K_j are their respective private keys, then the scheme defines K_i as $f_{U_i}(K_j)$, where f is any one-way hash function. The descendant role keys of a given role can therefore be computed by successive applications of the one-way function along the path going from the given role to each of its descendants. However, the scheme does not handle general partially ordered hierarchies.

This limitation of Sandhu's scheme was addressed, in 1993, by Zheng et al. [HZS93] who used *sibling intractable function families* (SIFF) [ZHP91] in order to only allow any parent (and thereby all ancestors) of a given role to be able to compute this role's key. In fact, Zheng et al. [HZS93] showed how to use SIFF to allow the *direct* computation of a role's key from any of the role's ancestors' keys. However, the scheme is unable to handle hierarchies which have transitive or anti-symmetrical exceptions.

A.3 Conclusion

We have reviewed past research on certificate-based encryption schemes proposed to support object-based access control. Two such classes of schemes have been proposed, namely algebraic schemes and schemes which rely on one-way functions. The latter schemes are more efficient than the former ones.

Bibliography

[ACJT00] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and Provably Secure Coalition-Resistant Group Signature Scheme. In *Proceedings of CRYPTO'00 on Advances in cryptology*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270. Springer-Verlag, 2000.

[AL02] C. Adams and S. Lloyd. *Understanding PKI: Concepts, Standards, and Deployment Considerations, Second Edition.* Addison-Wesley, 2002.

[ARP03] S.S. Al-Riyami and K.G. Paterson. Certificateless Public Key Cryptography. In *Proceedings of Asiacrypt 03 on Advances in cryptology*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473. Springer-Verlag, 2003.

[AT82] S.G. Akl and P.D. Taylor. Crytographic Solution to a Multilevel Security Problem. In *Proceedings of CRYPTO'82 on Advances in cryptology*, pages 237–249. Plenum Press, NY, 1982.

[AT83] S.G. Akl and P.D. Taylor. Cryptographic solution to a problem of access control in a hierarchy. *ACM Trans. Comput. Syst.*, 1(3):239–248, 1983.

[AT99] G. Ateniese and G. Tsudik. Some Open Issues and New Directions in group signatures. In *Financial Cryptography 1999*, volume 1648 of *Lecture Notes in Computer Science*, pages 196–211. Springer-Verlag, 1999.

[BB04] D. Boneh and X. Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In *Proceedings of EUROCRYPT'04 on Advances in cryptology*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer-Verlag, 2004.

[BBG05] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In *Proceedings of EUROCRYPT'05 on Advances in cryptology*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456. Springer-Verlag, 2005.

[BDT02] D. Boneh, X. Ding, and G. Tsudik. Identity-based Mediated RSA. In *Proceedings of the third International Workshop on Information and Security Applications (WISA'02)*, Jeju Island, Korea, 2002.

[BDT04] D. Boneh, X. Ding, and G. Tsudik. Fine-grained Control of Security Capabilities. *ACM Trans. Inter. Tech.*, 4(1):60–82, 2004.

[BDTW01] D. Boneh, X. Ding, G. Tsudik, and C. Wong. A Method for Fast revocation of Public Key Certificates and Security Capabilities. In *Proceedings of the 10th USENIX Security Symposium*, pages 297–308. USENIX, 2001.

[BF99] D. Boneh and M.K. Franklin. Anonymous Authentication with Subset Queries. In *Proceedings of the* 6^{th} *ACM conference on Computer and Communications Security*, pages 113–119. ACM, 1999.

[BF01] D. Boneh and M.K. Franklin. Identity-Based Encryption from the Weil Pairing. In *Proceedings of CRYPTO'01 on Advances in cryptology*, volume 2139, pages 213–229. Springer-Verlag, 2001.

[BKLS02] P.S.L.M. Barreto, H.Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Proceedings of CRYPTO'02 on Advances in cryptology*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–368. Springer-Verlag, 2002.

[Bla79] G.R. Blakeley. Safeguarding Cryptographic Keys. In *Proceedings of AFIPS 1979 National Computer Conference*, volume 48, pages 313–317, 1979.

[BLS01] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, volume 2248, pages 514–532. Springer-Verlag, 2001.

[BLS02] P.S.L.M. Barreto, B. Lynn, and M. Scott. Constructing elliptic curves with prescribed embedding degrees. In *Proceedings of the Conference on Security in Communication Networks (SCN'02)*, volume 2576 of *Lecture Notes in Computer Science*, pages 263–273. Springer-Verlag, 2002.

[BMW03] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Proceedings of EUROCRYPT'03 on Advances in cryptology*, volume 2656, pages 614–629. Springer Verlag, 2003.

[Boy03] X. Boyen. Multipurpose Identity-Based Signcryption: A Swiss Army Knife for Identity-Based Cryptography. In *Proceedings of CRYPTO'03 on Advances in cryptology*, volume 2729 of *Lecture Notes in Computer Science*, pages 383–399. Springer Verlag, 2003.

[Boy04] X. Boyen. Reusable Cryptographic Fuzzy Extractors. In *ACM Conference on Computer and Communications Security (CCS'04)*, pages 82–91. ACM Press, 2004. Available at `http://www.cs.stanford.edu/~xb/ccs04/`.

[BR93] M. Bellare and P. Rogaway. Random Oracles are practical: a Paradigm for Deisigning Efficient Protocols. In *Proceedings of the* 1^{st} *ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[BS04] D. Boneh and H. Shacham. Group Signatures with Verifier-Local Revocation. In *Proceedings of the* 11^{th} *ACM Conference on Computer and Communications Security*, pages 168–177. ACM, 2004.

[BZ04] J. Baek and Y. Zheng. Identity-Based Threshold Decryption. In *Proceedings of the 7th International Workshop on Theory and Practice in Public Key Cryptography (PKC'04)*, volume 2947 of *Lecture Notes in Computer Science*, pages 262–276. Springer-Verlag, 2004.

[Cam97] J. Camenish. Efficient and Generalized Group Signatures. In *Proceedings of EUROCRYPT'97 on Advances in cryptology*, volume 1233 of *Lecture Notes in Computer Science*, pages 465–479. Springer-Verlag, 1997.

[Cas02] C. Castellucia. How to Convert Any ID-based Signature Scheme into a Group Signature Scheme. In *Cryptology ePrint Archive, Report 2002/116*, 2002.

[CC03] J.C. Cha and J.H. Cheon. An Identity-based signature from gap Diffie-Hellman groups. In *Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography (PKC'03)*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30. Springer-Verlag, 2003.

[CDS94] R. Cramer, I. Damgārd, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proceedings of CRYPTO'94 on Advances in cryptology*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer-Verlag, 1994.

[CG99] R. Canetti and S. Goldwasser. An efficient threshold public-key cryptosystem secure against adaptive chosen ciphertext attack. In *Proceedings of EUROCRYPT'99 on Advances in cryptology*, volume 1592 of *Lecture Notes in Computer Science*, pages 90–106. Springer-Verlag, 1999.

[Cha82] D. Chaum. Blind signatures for untraceable payments. In *Proceedings of CRYPTO'82 on Advances in cryptology*, pages 199–203. Plenum, 1982.

[CHK03] R. Canetti, S. Halevi, and J. Katz. A Forward-Secure Public-Key Encryption Scheme. In *Proceedings of EUROCRYPT'03 on Advances in cryptology*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271. Springer-Verlag, 2003.

[CHM+02] L. Chen, K. Harrison, A. Moss, N. P. Smart, and D. Soldera. Certification of Public Keys within an Identity Based System. In A. H. Chan and V. Gligor, editors, *Proceedings of the 5th Information Security Conference (ISC'02)*, volume 2433 of *Lecture Notes in Computer Science*, pages 322–333. Springer-Verlag, 2002.

[CHSS02] L. Chen, K. Harrison, N. Smart, and D. Soldera. Applications of Multiple Trust Authorities in Pairing Based Cryptosystems. In *Proceedings of the International Conference on Infrastructure Security (InfraSec'02)*, volume 2437 of *Lecture Notes in Computer Science*, pages 260–275. Springer-Verlag, 2002.

[CHYC04] S.S.M. Chow, L.C.K. Hui, S.M. Yiu, and K.P. Chow. Secure Hierarchical Identity-based Signature and Its Applications. In *Proceedings of the sixth Conference on Information Security and Cryptology (ICICS'04)*, volume 3269 of *Lecture Notes in Computer Science*, pages 480–494. Springer-Verlag, 2004.

[Coc01] C. Cocks. An Identity Based Encryption Scheme Based on Quadratic Residues. In *Proceedings of Cryptography and Coding*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer-Verlag, 2001.

[CP94] L. Chen and T.P. Pedersen. New Group Signature Schemes. In *Proceedings of EUROCRYPTO'94 on Advances in cryptology*, volume 950 of *Lecture Notes in Computer Science*, pages 171–181. Springer-Verlag, 1994.

[CS97] J. Camenisch and M. Stadler. Efficient Group Signature schemes for large groups. In *Proceedings of CRYPTO'97 on Advances in cryptology*, volume

1070 of *Lecture Notes in Computer Science*, pages 410–424. Springer-Verlag, 1997.

[CvH91] D. Chaum and E. van Heijst. Group Signature. In *Proceedings of EUROCRYPT'91 on Advances in cryptology*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer-Verlag, 1991.

[CYHC03] S. S.M. Chow, S.M. Yiu, L.C.K. Hui, and K.P. Chow. Efficient Forward and Provably Secure ID-based Signcryption Scheme with Public Verifiability and Public Ciphertext Authenticity. In *Proceedings of the sixth Conference on Information Security and Cryptology (ICISC'03)*, volume 2971 of *Lecture Notes in Computer Science*, pages 352–369. Springer-Verlag, 2003.

[CYHY04] S. S.M. Chow, T.H. Yuen, L.C.K. Hui, and S.M. Yiu. Signcryption in Hierarchical Identity Based Cryptosystem. In *Available at* `http://eprint.iacr.org/2004/244.pdf`, 2004.

[CZK03] X. Chen, F. Zhang, and K. Kim. A new id-based group signature scheme from bilinear pairings. 2003. `http://eprint.iacr.org/`.

[Dau04] J. G. Daugman. How Iris Recognition Works. *IEEE Transaction on Circuits and Systems for Video Technology*, 14(1):21–30, 2004.

[DBS04] R. Dutta, R. Barua, and P. Sarkar. Pairing-Based Cryptography: A Survey. In *Cryptology ePrint Archive*, volume 2004/064, 2004.

[Des94] Y. G. Desmedt. Threshold cryptography. *European Transaction on Telecommunications*, 5(4):449–457, 1994.

[Des97] Y. Desmedt. Some Recent Research Aspects of Threshold Cryptography. In *Proceedings of the First International Workshop on Information Security*, volume 1396 of *Lecture Notes in Computer Science*, pages 158–173. Springer-Verlag, 1997.

[DH76] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[DQ86] Y. Desmedt and J. Quisquater. Public-key systems based on the difficulty of tampering. In *Proceedings of CRYPTO'86 on Advances in cryptology*, volume 263, pages 111–117. Springer-Verlag, 1986.

[DRS04] Y. Dodis, L. Reyzin, and S. Smith. Fuzzy Extractor: How to generate string keys from biometrics and other noisy data. In *Proceedings of EUROCRYPT'04 on Advances in Cryptology*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540. Springer-Verlag, 2004.

[dSFM04] A. de Santis, A.L. Ferrara, and B. Masucci. Cryptographic key assignment schemes for any access control policy. *Information Processing Letters*, pages 199–205, 2004.

[FFS87] U. Fiege, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. In *Proceedings of the nineteenth annual ACM conference on Theory of computing (STOC'87)*, pages 210–217. ACM Press, 1987.

[FFS88] U. Feige, A. Fiat, and A. Sharair. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1:77–94, 1988.

[FK89] D. Feldmeier and P. Karn. UNIX password security–Ten years later. In *Proceedings of CRYPTO'89 on Advances in cryptology*, volume 435 of *Lecture Notes in Computer Science*. Springer Verlag, 1989.

[FO99] E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer-Verlag, 1999.

[FSG+01] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST Standard for Role-Based Access Control. *ACM Transaction in Information System Security*, 4(3):224–274, 2001.

[FW94] W. Ford and M. J. Wiener. A key distribution method for object-based protection. In *Proceedings of the 2nd ACM Conference on Computer and communications security (CCS'94)*, pages 193–197. ACM Press, 1994.

[Gag03] M. Gagne. Identity-Based Encryption: a Survey. volume 6 of *CryptoBytes*, pages 10–19. RSA Laboratories, 2003.

[Gal01] S.D. Galbraith. Supersingular curves in cryptography. In *Proceedings of AsiaCrypt'01 on Advances in cryptology*, volume 2248 of *Lecture Notes in Computer Science*, pages 495–513. Springer-Verlag, 2001.

[Gen03] C. Gentry. Certificate-Based Encryption and the Certificate Revocation Problem. In *Proceedings of EUROCRYPT'03 on Advances in cryptology*, volume 2656 of *Lecture Notes in Computer Science*, pages 272–293. Springer-Verlag, 2003.

[GHS02] S.D. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In *Proceedings of the 5th International Conference on Algorithmic Number Theory (ANTS-V)*, volume 2369 of *Lecture Notes in Computer Science*, pages 324–337. Springer-Verlag, 2002.

[Gir91] M. Girault. Self-Certified Public Keys. In *Proceedings of EUROCRYPT'91 on Advances in Cryptology*, volume 547 of *Lecture Notes in Computer Science*, pages 490–497. Springer-Verlag, 1991.

[GLZ99] C. Gamage, J. Leiwo, and Y. Zheng. Encrypted message authentication by firewalls. In *proceedings of the Second International Workshop on Practice and Theory in Public Key Cryptography (PKC'99)*, pages 69–81. Springer-Verlag, 1999.

[GQ88] L. Guillou and J.-J. Quisquater. A "Paradoxical" Identity-Based Signature Scheme Resulting From Zero-Knowledge. In *Proceedings of CRYPTO'88 on Advances in cryptology*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231. Springer-Verlag, 1988.

[GS02] C. Gentry and A. Silverberg. Hierarchical ID-Based Cryptography. In *Proceedings of ASIACRYPT'02 on Advances in cryptology*, volume 2501, pages 548–566. Springer-Verlag, 2002.

[Gut02] P. Gutmann. PKI: it's not dead, just resting. *IEEE Computer*, 35(8):41–49, 2002.

[HC04] H.-F. Huang and C.-C. Chang. A new cryptographic key assignment scheme with time-constraint access control in a hierarchy. *Computer Standards and Interfaces*, pages 159–166, 2004.

[Hes02] F. Hess. Efficient Identity-Based Signature Schemes Based on Pairings. In *Proceedings of SAC 02 on Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324. Springer-Verlag, 2002.

[HL90] L. Harn and H.-Y. Lin. A cryptographic key generation scheme for multilevel data security. *Computer & Security*, 9(6):539–546, 1990.

[HL02] J. Horwitz and B. Lynn. Towards Hierarchical Identity-Based Encryption. In *Proceedings of EUROCRYPT'02 on Advances in cryptology*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481. Springer-Verlag, 2002.

[HPS02] K. Harrison, D. Page, and N.P. Smart. Software Implementation of Finite Fields of Characteristic three, for use in pairing based cryptosystems. *London Mathematical Society Journal of Computing Mathemetics*, 5:181–193, 2002.

[Hwa99] M.-S. Hwang. An improvement of a dynamic cryptographic key assignment schemes in a tree hierarchy. *Computers and Mathematics with Applications*, pages 19–22, 1999.

[HY03] M.-S. Hwang and W.-P. Yang. Controlling access in large partially ordered hierarchies using cryptographic keys. *Journal of Systems and Software*, pages 99–107, 2003.

[HZS93] T. Hardjono, Y. Zheng, and J. Seberry. New solutions to the problem of access control in a hierarchy. Technical Report Preprint 93-2, University of Wollongong, 1993.

[JKL99] M. Joye, S. Kim, and N. Lee. Cryptanalysis of Two Group Signature Schemes. In *Information Security 1999*, volume 1729 of *Lecture Notes in Computer Science*, pages 271–275, Springer-Verlag, 1999.

[Jou00] A. Joux. A one-round protocol for tripartite Diffie-Hellman. In *Proceedings of ANTS-IV conference*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394, Springer-Verlag, 2000.

[Joy99] M. Joye. On the Difficulty of Coalition-Resistance in Group Signature Schemes (II). In *Technical Report*, LCIS-99-6B, 1999.

[JRP01] A. Jain, A. Ross, and S. Prabhakar. Fingerprint Matching Using Minutiae and Texture Features. In *Proceedings of the International Conference on Image Processing (ICIP)*, pages 282–285, 2001.

[JvO99] M. Just and P.C. van Oorschot. Addressing the Problem of Undetected Signature Key Compromise. In *Proceedings of the 6th Annual Network and Distributed System Security Symposium (NDSS'99)*, Internet Society, 1999.

[JW99] A. Juels and M. Wattenberg. A Fuzzy commitment scheme. In *Proceedings of the 6th ACM Conference on Computer and Communication Security*, pages 28–36, 1999.

[Kle90] D. Klein. Foiling the cracker: A survey of, and improvements to, password security. In *Proceedings of the 2nd USENIX Security Workshop*, pages 5–14, 1990.

[KMPR05] E. Kiltz, A. Mityagin, S. Panjwani, and B. Raghavan. Append-only signatures. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *Lecture Notes in Computer Science*, pages 434–445, Springer-Verlag, 2005.

[KMS05] C. Konoma, M. Mambo, and H. Shizuya. The Computational Difficulty of Solving Cryptographic Primitive Problems Related to the Discrete Logarithm Problem. *IEICE Trans Fundamentals*, E88-A(1):81–88, 2005. `http://ietfec.oupjournals.org/cgi/content/abstract/E88-A/1/81`.

[Koc98] P. Kocher. On certificate revocation and validation. In *Financial Cryptography (FC'98)*, volume 1465 of *Lecture Notes in Computer Science*, pages 172–177, Springer-Verlag, 1998.

[KP98] J. Kilian and E. Petrank. Identity Escrow. In *Proceedings of CRYPTO'98 on Advances in cryptology*, volume 1462 of *Lecture Notes in Computer Science*, page 169. Springer-Verlag, 1998.

[KPW97] S. Kim, S. Park, and D. Won. Group signatures for hierarchical multigroups. In *Information Security Workshop - ISW'97*, volume 1396 of *Lecture Notes in Computer Science*, pages 273–281, Springer-Verlag, 1997.

[Kul] M. Kula. A Cryptosystem Based on Double Discrete Logarithm. In *Draft available at www.ulb.ac.be/di/scsi/classicsys/sedfrob.pdf*.

[Lan93] S. Lang. *Algebra, Third Edition.* Addison-Wesley, 1993.

[LHC03] I.-C. Lin, M.-S. Hwang, and C.-C. Chang. A new key assignment scheme for enforcing complicated access control policies in hierarchy. *Future Generation Computer Systems*, pages 457–462, 2003.

[Lin97] C.H. Lin. Dynamic key management schemes for access control in a hierarchy. *Computer Communications*, 20:1381–1385, 1997.

[LP01] A. Lysyanskaya and C. Peikert. Adaptive security in the threshold setting: From cryptosystems to signature schemes. In *Proceedings of ASIACRYPT 01 on Advances in cryptology*, volume 2248 of *Lecture Notes in Computer Science*, pages 331–350. Springer-Verlag, 2001.

[LQ03a] B. Libert and J.-J. Quisquater. Efficient Revocation and Threshold Pairing Based Cryptosystems. In *Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 163–171. ACM Press, 2003.

[LQ03b] B. Libert and J.-J. Quisquater. New Identity Based Signcryption Schemes from Pairings. In *Proceedings of the IEEE Information Theory Workshop*, pages 155–158. Full version available at `http://eprint.iacr.org`, 2003.

[LQ04] B. Libert and J.-J. Quisquater. The Exact Security of an Identity Based Signature and its Applications. In *Cryptology ePrint Archive, Report 2004/102*, 2004.

[LR89] K. Lougheed and Y. Rekhter. A Border Gateway Protocol (BGP), RFC 1105, 1989.

[LR98] A. Lysyanskaya and Z. Ramzan. Group blind digital signatures: A scalable solution to electronic cash. In *Proceedings of the Second International Conference on Financial Cryptography (FC'98)*, pages 184–197. Springer-Verlag, 1998.

[LWL93] H.T. Liaw, S.J. Wang, and C.L. Lei. A dynamic cryptographic key assignment scheme in a tree structure. *Computers and Mathematics with Applications*, 25(6):109–114, 1993.

[MA83] S. MacKinnon and S.G. Akl. New Key Generation Algorithms for Multilevel Security. In *IEEE Symposium on Security and Privacy*, pages 72–78. IEEE Press, 1983.

[MAM+99] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. Internet public key infrastructure online certificate status protocol - OCSP. In *RFC 2560*, 1999.

[MB04] N. McCullagh and P.S.L.M. Barreto. Efficient and forward-secure identity-based signcryption. 2004. `http://eprint.iacr.org/`.

[MHS03] M. Casassa Mont, K. Harrison, and Ma. Sadler. The HP time vault service: exploiting IBE for timed release of confidential information. In *Proceedings of the twelfth international conference on World Wide Web*, pages 160–169. ACM Press, 2003.

[ML98] W. Mao and C.H. Lim. Cryptanalysis in Prime Order Subgroup of $\mathbb{Z}_n$. In *Proceedings of ASIACRYPT'98 on Advances in cryptology*, volume 1514 of *Lecture Notes in Computer Science*, pages 214–226. Springer-Verlag, 1998.

[ML02] J. Malone-Lee. Identity Based Signcryption. Cryptology ePrint Archive, Report 2002/098, 2002.

[Moy98] J. Moy. OSPF Version 2, RFC 2328, 1998.

[MPI+03] S. Miltchev, V. Prevelakis, S. Ioannidis, J. Ioannidis, A. Keromytis, and J. Smith. Secure and Flexible Global File Sharing. In *Proceedings of the USENIX Technical Annual Conference, Freenix Track. (2003)*. USENIX, 2003.

[MRL+02] F. Monrose, M. K. Reiter, Q. Li, D. Lopresti, and C. Shih. Towards voice generated cryptographic keys on ressource constrained devices. In *Proceedings of the 11th USENIX Security Symposium*, 2002.

[MRLW01] F. Monrose, M. K. Reiter, Q. Li, and S. Wetzel. Cryptographic key generation from voice. In *Proceedings of the IEEE Conference on Security and Privacy*, pages 202–213. IEEE Press, 2001.

[MRW99] F. Monrose, M. K. Reiter, and S. Wetzel. Password hardening based on key-stroke dynamics. In *Proceedings of the 6th ACM Conference on Computer and communications security (CCS'99)*, pages 73–82. ACM Press, 1999.

[MTMA85] S. MacKinnon, P.D. Taylor, H. Meijer, and S.G. Akl. An optimal algorithm for assigning cryptographic keys to access control in a hierarchy. *IEEE Transactions on Computers*, pages 797–802, 1985.

[MvOV97] A. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[MY91] U. Maurer and Y. Yacobi. Non-interactive public-key cryptography. In *Proceedings of CRYPTO'91 on Advances in cryptology*, volume 547 of *Lecture Notes in Computer Science*, pages 498–507. Springer-Verlag, 1991.

[NAM04] D. Nali, C. Adams, and A. Miri. Using Mediated Identity-Based Cryptography to Support Role-Based Access Control. In *Proceedings of the 7th Information Security Conference (ISC'04)*, volume 3225 of *Lecture Notes in Computer Science*, pages 245–256. Springer-Verlag, 2004.

[NAM05a] D. Nali, C. Adams, and A. Miri. An Efficient and Flexible Scheme to Support Biometric-Based and Role-Based Access Control. In *Proceedings of the 4th International Workshop for Applied PKI (IWAP'05)*, volume 128, pages 139–154. IOS Press, 2005.

[NAM05b] D. Nali, C. Adams, and A. Miri. Hierarchical Identity-Based Encryption with Constant Ciphertext and Key Length. In *Manuscript*, 2005.

[NAM05c] D. Nali, C. Adams, and A. Miri. Time-based Release of Confidential Information in Hierarchical Settings. In *Proceedings of the 8th Information Security Conference (ISC'05)*, volume 3650 of *Lecture Notes in Computer Science*, pages 29–43. Springer-Verlag, 2005.

[NAM05d] D. Nali, C. Adams, and A. Miri. Using Threshold Attribute-Based Encryption for Practical Biometric-Based Access Control. *International Journal of Network Security*, 1(3):173–182, 2005.

[NMA04] D. Nali, A. Miri, and C. Adams. Efficient Revocation of Dynamic Security Privileges in Hierarchically Structured Communities. In *Proceedings of the 2nd Annual Conference on Privacy, Security and Trust (PST 2004)*, pages 219–223, 2004.

[NMA06] D. Nali, A. Miri, and C. Adams. Traffic Filtering and Routing in Partially Hidden Networks. *International Journal of Network Security*, 2(2):91–104, 2006.

[NR03] D. Nalla and K.C. Reddy. Signcryption scheme for Identity-Based Cryptosystems. Cryptology ePrint Archive, Report 2003/066, 2003.

[oSN99] National Institute of Standards and Technology (NIST). Round 2 Discussion Issues for the AES Development Effort. 1999.

[Pat02] K.G. Paterson. ID-based signatures from pairings on elliptic curves. In *Electronics Letters*, volume 38 of *Lecture Notes in Computer Science*, pages 1025–1026. Springer-Verlag, 2002.

[Ped92] T. Pedersen. Non-interactive and Information theoretic Secure Verifiable Secret Sharing. In *Proceedings of CRYPTO'91 on Advances in cryptology*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer-Verlag, 1992.

[Pet97] H. Petersen. How to Convert Any Digital Signature Scheme into a Group Signature Scheme. In *Proceedings of the Security Protocols Workshop 1997*, volume 1361 of *Lecture Notes in Computer Science*, pages 177–190. Springer-Verlag, 1997.

[PKW97] S. Park, S. Kim, and D. Won. ID-based Group Signature. In *Electronics Letters*, volume 33, pages 1616–1617, 1997.

[Pop00] C. Popescu. Group Signature Schemes Based on the Difficulty of Computation of Approximate e-th roots. In *Proceedings of Protocols for Multimedia Systems (PROMS 2000)*, pages 325–331, Poland, 2000.

[PP03] K.G. Paterson and G. Price. A Comparison Between Traditional Public Key Infrastructures and Identity-Based Cryptography. *Information Security Technical Report*, 8(3):57–72, 2003.

[RBBK01] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption. *ACM Transactions on Information and System Security (TISSEC)*, 6(3):365–403, 2001.

[RRN02] I. Ray, I. Ray, and N. Narasimhamurthi. A cryptographic solution to implement access control in a hierarchy and more. In *Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 65–73. ACM Press, 2002.

[RSA78] R.L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[San88] R. S. Sandhu. Cryptographic implementation of a tree hierarchy for access control. *Inf. Process. Lett.*, 27(2):95–98, 1988.

[SC02] V.R.L. Shen and T.-S. Chen. A Novel Key Management Scheme Based on Discrete Logarithms and Polynomial Interpolations. *Computers and Security*, pages 164–171, 2002.

[SCPY94] A. De Santis, G. Di Crescenzo, G. Persiano, and M. Yung. On monotone formula closure of SZK. In *IEEE Foundations of Computer Science*, pages 454–465. IEEE Press, 1994.

[Sha79] A. Shamir. How to Share a Secret. *Communication of the ACM*, 22(11):612–613, 1979.

[Sha84] A. Shamir. Identity-Based Cryptosystems and Signature Schemes. In *Proceedings of CRYPTO'84 on Advances in cryptology*, pages 47–53. Springer-Verlag New York, Inc., 1984.

[Sim88] G. J. Simmons. How to (really) share a secret. In *Proceedings of CRYPTO'88 on Advances in cryptology*, volume 403 of *Lecture Notes in Computer Science*, pages 390–448. Springer-Verlag, 1988.

[SM95] G. J. Simmons and C. Meadows. The role of trust in information security protocols. *Journal of Computer Security*, 3:71–84, 1994/1995.

[Sma03a] N. Smart. Access control using pairing based cryptography. In *Proceedings CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 111–121. Springer-Verlag, April 2003.

[Sma03b] N. Smart. An identity based authenticated key agreement protocol based on the weil pairing. *Electronics Letters*, pages 630–632, 2003.

[SOK00] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems Based on Pairings. In *The 2000 Symposium on Cryptography and Information Security*, 2000.

[SW05] A. Sahai and B. Waters. Fuzzy identity based encryption. In *Proceedings of EUROCRYPT'05 on Advances in Cryptoglogy*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer-Verlag, 2005.

[SZ00] R. Steinfield and Y. Zheng. A Signcryption Scheme Based on Integer Factorization. In *Proceedings of ISW'00*, pages 308–322, 2000.

[Tan87] H. Tanaka. A realization scheme for the identity-based cryptosystem. In *Proceedings of CRYPTO'87 on Advances in cryptology*, volume 293 of *Lecture Notes in Computer Science*, pages 341–349. Springer-Verlag, 1987.

[TJ98] Y.M. Tseng and J.K. Jan. A Novel ID-Based Group Signature. In *Workshop on Cryptology and Information Security*, pages 159–164, Tainan, 1998.

[Tur37] A.M. Turing. On Computable Numbers, With an Application to the Entscheidungsproblem. In *London Mathematical Society*, volume 2, pages 230–265, 1937.

[Tur00] J. Turnbull. Cross-certification and pki policy networking, v 1.0, `http://www.entrust.com/resources/pdf/cross_certification.pdf`, 2000.

[TW05] M. Trolin and D. Wickström. Hierarchical Group Signature. In 446-458, editor, *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *Lecture Notes in Computer Science*, pages 434–445, Springer-Verlag, 2005.

[Wu99] T. Wu. A real-world analysis of Kerebros password security. In *Proceedings of the 1999 ISOC Symposium on Network and Distributed System Security*, volume (8) of *9*, pages 723–736, 1999.

[Y. 02] Y. Dodis and M. Yung. Exposure-Resilience for Free: the Hierarchical ID-based Encryption Case. In *Proceedings of IEEE International Security In Storage Workshop (SISW'02)*, pages 45–52, 2002.

[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 *Proceedings of the 11th ACM conference on Computer and communications security (CCS'04)*, pages 354–363. ACM Press, 2004.

[YL01] B.H. Yum and P.J. Lee. New Signcryption Schemes Based on KCDSA. In *Proceedings of ICISC'01*, volume 2288 of *Lecture Notes in Computer Science*, pages 305–317. Springer-Verlag, 2001.

[Zhe97] Y. Zheng. Digital Signcryption or How to Achieve Cost (Signature & Encryption) $\ll$ Cost(Signature) + Cost(Encryption). In *Proceedings of CRYPTO'97 on Advances in cryptology*, volume 1294 of *Lecture Notes in Computer Science*, pages 165–179. Springer-Verlag, 1997.

[Zhe98] Y. Zheng. Signcryption and its Applications in Efficient Public Key Solutions. In *Proceedings of ISW'97*, pages 291–312, 1998.

[Zhe01] Y. Zheng. Identification, Signature, and Signcryption using High Order Residues Modulo and RSA Composite. In *Proceedings of PKC'01*, volume 1992 of *Lecture Notes in Computer Science*, pages 48–63. Springer-Verlag, 2001.

[ZHP91] Y. Zheng, T. Hardjono, and J. Pieprzyk. Sibling Intractable Function Families and their Applications. In *Proceedings of ASIACRYPT'91 on Advances in cryptology*, volume 739 of *Lecture Notes in Computer Science*, pages 124–138. Springer-Verlag, 1991.

[ZI98] Y. Zheng and H. Imai. Efficient Signcryption Schemes on Elliptic Curves. In *Proceedings of IFIP/SEC'98*. Chapman & Hall, 1998.

[ZK02] F. Zhang and K. Kim. ID-Based Blind Signature and Ring Signature from Pairings. In *Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security (AsiaCrypt'02)*, volume 2501 of *Lecture Notes in Computer Science*, pages 533–547. Springer-Verlag, 2002.