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

Nicholas Mailloux

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.Sc. (Mathematics)

GRADE / DEGREE

Department of Mathematics and Statistics

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Group Key Agreement from Bilinear Pairings

TITRE DE LA THÈSE / TITLE OF THESIS

Ali Miri

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

Monica Nevins

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

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Sylvia Boyd

Daniel Panario

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Group Key Agreement from Bilinear Pairings

Nicholas J. Mailloux

Thesis Submitted to the Faculty of Graduate and Postdoctoral Studies
In partial fulfilment of the requirements for the degree of Master of Science in Mathematics [1]

Department of Mathematics and Statistics
Faculty of Science
University of Ottawa

[1] The M.Sc. program is a joint program with Carleton University, administered by the Ottawa-Carleton Institute of Mathematics and Statistics

Your file Votre référence
ISBN: 978-0-494-61164-7
Our file Notre référence
ISBN: 978-0-494-61164-7

Abstract

The use of bilinear pairings as a building block for cryptographic protocols, most notably in the construction of identity-based cryptosystems, is a very popular area of cryptographic research. In this thesis, we provide a novel classification of pairing-based group key agreement (GKA) from current literature. We propose a new framework for constructing secure and efficient computationally asymmetric authenticated GKA protocols from identity-based signcryption schemes and adapt this framework to construct a novel identity-based authenticated GKA protocol with perfect forward secrecy. To the best of our knowledge, our protocol is the first that maintains perfect forward secrecy in the presence of auxiliary key agreement protocols. We formally prove the security of our protocols in the random oracle model and show that they are communication and computationally efficient in comparison to the pairing-based protocols from the literature.

Acknowledgements

I would like to thank my supervisors, Isabelle Déchène, Ali Miri and Monica Nevins, for their guidance, their patience and their support both inside and outside of the department.

I would also like to thank the Department of Mathematics and Statistics and the National Science and Engineering Research Council for financial support throughout my Masters degree.

Finally, I would like to thank my family and friends for their patience and understanding throughout the writing of this thesis.

Dedication

To Isabelle, whose encouragement and enthusiasm will be greatly missed.

Contents

List of Figures

List of Acronyms

AGKA	Authenticated Group Key Agreement
AKA	Auxiliary Key Agreement
BDH	Bilinear Diffie-Hellman problem
CA	Certification Authority
CCA1	Chosen Ciphertext Attack
CCA2	Adaptive Chosen Ciphertext Attack
CMA	Chosen Message Attack
CPA	Chosen Plaintext Attack
DBDH	Decisional Bilinear Diffie-Hellman problem
DDH	Decisional Diffie-Hellman problem
DHP	Diffie-Hellman Problem
DLP	Discrete Logarithm Problem
EUF	Existentially Unforgeable
GC	Group Controller
GDH	Gap Diffie-Hellman
GKA	Group Key Agreement
GKD	Group Key Distribution
IBE	Identity-based Encryption
IBS	Identity-based Signature
ID-AGKA	Identity-based Authenticated Group Key Agreement
ID-PKI	Identity-based Public Key Infrastructure
IKA	Initial Key Agreement
IND	Indistinguishability
KEK	Key Encryption Key
KMA	Known Message Attack
KS	Key Server
NIKDS	Non-Interactive Key Distribution Scheme
NMA	No Message Attack
PKG	Private Key Generator
TA	Trusted Authority
TEK	Traffic Encryption Key
TTP	Trusted Third Party

List of Notation

ℓ	Security parameter
$\mathbb{G}$	A cyclic group of prime order q with generator g
$\mathbb{G}^{\times}$	$\mathbb{G} \setminus \mathcal{O}$ where $\mathcal{O}$ is the identity of $\mathbb{G}$
$\in_R \mathbb{G}$	An element taken at random from $\mathbb{G}$
e	The bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$
$\mathbb{Z}_q$	The group of integers mod q, $\mathbb{Z}_q = \{0, 1, 2, ..., q-1\}$
$\{0,1\}^*$	Binary string
$\oplus$	Bitwise addition of binary strings (i.e. XOR operation)
ID	An identity string (i.e. email address), $\mathsf{ID} \in \{0,1\}^*$
ID^*	The challenge identity in security game
$\mathcal{U}$	The multicast group
$\mathcal{R}$	The group of responders $\{U_1, ..., U_n\}$ in asymmetric protocols
n	The size of the multicast group $\mathcal{U}$
$A \xrightarrow{B} \Gamma : x$	Indicates that the message x is broadcasted to the set of entities Γ by the entity A
$A : x \leftarrow \mathcal{F}$	Indicates that the entity A executes the algorithm $\mathcal{F}$ to obtain the message x
$*$	The protocol does not guarantee authentication of all key material
$\Diamond$	The protocol uses some unspecified signature scheme
$\dagger$	The protocol does not maintain perfect forward secrecy when the proposed AKA protocols are employed

List of Protocols

Chapter 1

Introduction

In the past few years, group-oriented applications, such as video conferencing, data multicasting or multi-player gaming, have become more pervasive in dynamic multicast networks. Before such multicast applications are fully adopted, efficient cryptographic protocols must be designed to provide suitable levels of security for these specific communication models. The general approach is to encrypt the group communication with a traffic encryption key known only by members of the multicast group. The problem is to establish this shared group key in such a way that we provide the strongest notion of security.

In the search for suitable *group key establishment* protocols, we find that they generally fall under two broad categories: *group key distribution* and *group key agreement.* Group key distribution (GKD) protocols require a trusted authority (TA) to securely distribute the shared key to the members of the multicast group via some secure channel. These protocols effectively reduce the computational load of individual group members. However, they often suffer from performance bottlenecks and a single point of failure, due to the requirement of a TA. Furthermore, since the TA usually transports the group key using public key cryptography, GKD protocols do not often provide *perfect forward secrecy,* where the disclosure of the long-term private keys of all group members does not compromise previous session keys. Group key agreement (GKA) protocols require a group of members to collaboratively establish the shared key, ensuring that no member can precompute the key for a given session. Each member *must* be assured that their own contribution has been used to compute

the key to ensure that no other member can control the final outcome of the session key, known as *key control.* This property is one of the defining characteristics of GKA protocols and makes them particularly favourable for use in multicast group communication models.

A more recently proposed class of group key establishment protocols that combines the favourable qualities of GKD and GKA protocols is the *computationally asymmetric GKA protocols.* In this class of protocol, a multicast group consisting of a stationary host, which we will refer to as the *group leader*, and a cluster of client members, known as *responders*, collaborate to produce the shared group key. Each group member contributes equally to the group key, but most of the computational burden is shifted to the group leader. While this approach reduces the computational costs of the majority of the responders, the protocols must be designed with care to avoid potential drawbacks, such as performance bottlenecks and single point of failure associated with the group leader, while also dealing with the issues of communication bandwidth and authentication. These protocols are commonly employed in networks of low-power mobile devices with limited computational capabilites. They are exceptionally well suited for this model since we can delegate a large portion of the expensive computations to the more powerful group leader, which has sufficient computational resources and storage capacity.

The design of efficient *auxiliary key agreement* (AKA) protocols is essential for reducing communication complexity in dynamic multicast groups. Auxiliary protocols are used to *rekey* the group following membership changes. Requiring the execution of the *initial key agreement* (IKA) protocol after a single member joins or leaves the multicast group induces high communication bandwidth and extra computations for group members. It becomes increasingly more inefficient as the multicast group grows in size. Employing AKA protocols can effectively reduce both computational and communication costs. However, this introduces the problem of maintaining perfect forward secrecy in the auxiliary protocols. Group key agreement protocols without auxiliary key agreement often have inherent perfect forward secrecy, since new ephemeral values are chosen for each session, but suffer from high communication costs. Designing communication efficient protocols that provide perfect forward secrecy has proven to be a nontrivial problem.

More recently, cryptographers have used bilinear pairings on elliptic curves to construct efficient group key agreement protocols. Indeed, since Joux first used pairings to construct a tripartite version of the Diffie-Hellman key exchange in [Jou00], they have become a powerful building block in the design of cryptographic protocols. The most powerful application of pairings is in the realization of *identity-based public key cryptography.* An *identity-based public key infrastructure* (ID-PKI), first proposed by Shamir [Sha85] in 1984, allows a user's public key to be any unique, arbitrary string, such as a name with a social insurance number or an email address. A trusted authority, known as the *private key generator* (PKG), produces and securely transmits the private keys corresponding to each user's public key, implicitly ensuring its authenticity. ID-PKIs present an attractive alternative to certificate-based PKIs, which often induce high costs associated with authentication and key management. A practical solution to the problem of identity-based cryptography remained an open problem until 2001, when Boneh and Franklin [BF01] used bilinear pairings to construct the first practical and provably secure identity-based encryption scheme. As a result of this cryptographic breakthrough, pairing- and identity-based cryptography have become two of the most heavily researched areas in the field.

1.1 Structure and Contributions of this Thesis

The primary focus of this thesis revolves around group key agreement from bilinear pairings. Our goal is to analyze the use of pairings in group key agreement protocols from the literature and propose new protocols that provide increased efficiency and security over the current solutions.

To provide a thorough background for our work, we outline the issues related to group key establishment, and group key agreement in particular, in Chapter 2. To introduce the fields of pairing- and identity-based cryptography, we illustrate the bilinear pairings as a cryptographic building block in Chapters 3 and 4 by presenting some of the more influential works from the literature. In particular, we present the tripartite key exchange of Joux and the identity-based encryption scheme of Boneh and Franklin in Chapter 3 and the identity-based signature schemes of Cha-Cheon and Hess and the identity-based signcryption scheme of Chen and Malone-Lee in

Chapter 4. In Appendix A, we provide a thorough discussion of the mathematics behind bilinear pairings, including an overview of concepts from the theory of elliptic curves and divisors, which are used in the definition and computation of the Weil and Tate pairings.

Chapters 5, 6 and 7 are the central focus of this thesis and provide several new and intriguing contributions to the field of pairing-based group key agreement. We list the contribution of each chapter in more detail, as follows:

- Chapter 5 is devoted to analyzing the state of the art of pairing-based group key agreement. Although dozens of pairing-based GKA protocols have been proposed, a literature survey comprising the various protocols has yet to have been proposed. In this chapter, we not only amass the pairing-based GKA protocols from the literature, but also propose a thorough classification of these protocols. We found that every protocol that we encountered that satisfied the essential property of group key secrecy (i.e. that an adversary cannot determine the session key by eavesdropping on the communication), belonged to one of the following three classes of group key agreement protocols: the tree-based GKA protocols, the Burmester-Desmedt-based GKA protocol or the computationally asymmetric GKA protocols. We divide the protocols into their respective classes and examine the use of pairings in the key agreement process and the communication and computational efficiency of each individual protocol. For each class, we determine whether the use of pairings provides any advantages over the original GKA protocol and conclude which pairing-based version is best.

- In Chapter 6, we propose a general framework to transform any signature and signcryption scheme pair into a provably secure authenticated group key agreement protocol. This novel framework, which we call the COMPASS framework, achieves the lower bound for communication in group key agreement, requiring only a single broadcast message from each member in only one round of communication. While the overall idea builds upon the computationally asymmetric conference key agreement protocol of Boyd and González Nieto, our framework provides stronger security than their protocol. In addition, we show that by choosing a signcryption scheme that satisfies some desirable criteria, we can

provide a number of computational optimizations to the basic framework. We prove the semantic security of the optimized COMPASS framework in the random oracle model, by reducing the security of the COMPASS protocol to the security of the underlying signcryption scheme.

- In Chapter 7, we propose a novel identity-based authenticated group key agreement protocol that provides the oft sought property of perfect forward secrecy. Our protocol is adapted from the COMPASS framework, but relies on our observation that a collection of signatures from the identity-based signature schemes of Cha-Cheon and Hess may be efficiently batch verified together using the same algorithm. We prove the semantic security of our protocol under the decisional bilinear Diffie-Hellman assumption in the random oracle model. In addition, we propose auxiliary key agreement protocols to rekey the multicast group following a change in group membership and show that our protocol maintains perfect forward secrecy when these auxiliary protocols are employed. Furthermore, the auxiliary key agreement protocols provide this property while allowing members to discard ephemeral values in the same session in which they were created, which, as far as we know, is the first group key agreement protocol to achieve this feat.

Finally, in Chapter 8, we provide a summary of our work and suggest some areas of future research.

Chapter 2

Key Management in Group Key Agreement Protocols

The prevalence of group-oriented applications has triggered a demand for multicast communication. In the unicast model, group communication requires a member to send n copies of the same data packet, one to each member in the group. *Multicasting* allows a user to simultaneously transmit the data packet to all n group members, saving bandwidth and requiring minimal resources. Clearly, the multicast communication model is more suitable for group applications. However, as discussed in [RH03, CS05], the multicast communication model lacks suitable levels of security in the areas of *confidentiality*, *authentication* and *access control.* In particular, they note that any multicast-enabled host can join a given multicast group and gain access to the group communication. Before group-oriented applications are adapted for multicast communication on a grand scale, cryptographers must address these security issues.

The logical solution to provide secure group communication is to encrypt the data packets with a shared *group key* known only by the legitimate members of the multicast group. In this scenario, the problem becomes managing the group key in order to provide confidentiality, authentication and access control from one session to another.

In this chapter, we discuss the concept of *group key management*, a primary component in the security of multicast architecture, and how it relates to *group key*

agreement. In Section 2.1, we give a broad overview of key management in group key establishment protocols. In Section 2.2, we narrow in on the class of group key agreement protocols, as they are the primary focus of our work, and discuss issues specific to group key agreement. We provide a literature survey consisting of some of the more popular and creative group key agreement protocols in Section 2.3. This survey serves as motivation for Chapter 5, where we'll see how cryptographers applied the concept of pairings to these models to produce fast, efficient and secure pairing-based versions of these group key agreement protocols. Finally we summarize the results of the chapter in Section 2.4.

2.1 Group Key Management

In this section, we investigate the idea of key management in group key establishment protocols. We follow the presentation of Challal and Seba in [CS05], which is an invaluable source on the topic of group key management, and discuss the necessary concepts as defined by the authors. To begin, we outline the primary components of group key management and discuss the structure of our multicast group in Section 2.1.1. We outline the particular security and quality of service goals of group key management in Section 2.1.2 and look at a classification of the various types of group key establishment protocols in Section 2.1.3.

2.1.1 Components of Group Key Management

Informally speaking, a group key management scheme involves a group of members called *receivers* that are sent some encrypted data from a *source* member in a multicast session. As previously mentioned, in order to ensure the security of the session, we must address the issues of confidentiality, authentication and access control of group members. In group key management schemes, these tasks are performed by the Group Controller and the Key Server.

Multicast Group

The multicast group $\mathcal{U}$ consists of the receivers and the source member in the session. Upon creation of the group, an *initial key establishment* protocol is used to produce

the shared session key. As we are dealing with dynamic multicast groups, several members may leave or join the multicast group thoughout its lifetime. To ensure the confidentiality of previous sessions, we must *rekey* the group to establish a new session key. Executing the initial key establishment protocol after a single member joins the group can often induce high communication bandwidth and computational costs for the group members. We may often like to execute *auxiliary key establishment* protocols that reduce the communication complexity of the rekey operation.

Group Controller

The *Group Controller (GC)* is a logical entity responsible for the access control of group members and handling group membership changes. Before a member can join the multicast group, they must be authenticated and given authorization by the GC. Similarly, when a member leaves or is forcefully dismissed from the group, the GC processes these requests and notifies the remaining group members. This process ensures that only legitimate group members will be given access to group communication. As the role of the GC is largely administrative, group key establishment protocols commonly assume the existence of a GC and focus on the more challenging tasks of key management.

Key Server

The *Key Server (KS)* is a logical entity (such as an external entity or subset of group members) responsible for updating and managing the secret keys of the group. These keys are divided into two classes: *Traffic Encryption Keys* and *Key Encryption Keys.*

The *Traffic Encryption Key (TEK)* is the secret group key shared by all valid members of the group. To ensure confidentiality, the source member encrypts the message using the *TEK* and sends it to all receivers in the multicast session. Valid group members can then decrypt the message using the *TEK* and recover the original message.

The *Key Encryption Key* (KEK_i) is a unique secret key shared between each valid group member U_i and the KS.

After a membership change, the KS rekeys the multicast group by generating a new *TEK* and distributing it to group members in a secure manner. A naive approach

is to have the KS encrypt the new *TEK* using the *TEK* from the previous session. However, we see that this does not provide confidentiality if a member leaves the group. Another approach is to have the KS encrypt the new *TEK* for each member U_i, using their secret key KEK_i. The group members can then decrypt the message using their secret key KEK_i to obtain the new *TEK*. However, this induces a heavy computational load for the KS and drastically increases the length of the multicast message, especially for large multicast groups. As one can see, rekeying a highly dynamic multicast group in a manner that is computationally, communication and storage efficient can prove to be a difficult task. The maintenance and distribution of the *TEK*s and *KEK*s is a central role of group key management.

Remark 2.1 *In many protocols, the roles of the GC and the KS may be performed by a single entity or may be divided amongst different entities. For example, in the identity-based setting, the private key generator (PKG) performs the KEK distribution tasks of the KS and assumes the role of the GC. However, the management of TEKs is handled by some other entity, depending on the protocol. We discuss identity-based protocols in more detail in Chapter 3.*

2.1.2 Goals of Group Key Management

In this section, we discuss the particular goals of group key management in terms of security, quality of service and overhead costs, as defined in [CS05]. We discuss the security of a group key protocol in terms of two different types of adversaries. A *passive adversary* can eavesdrop on messages sent between session participants, including messages used to establish the group key and traffic encrypted with the group key. An *active adversary* can eavesdrop on the traffic, but also disrupt the communication by inserting, deleting or modifying the messages sent between session participants.

Security

A secure group key protocol must satisfy all of the following requirements if it is intended for multi-party applications.

- **Group Key Secrecy**: A protocol provides *group key secrecy* if it is computationally infeasible for a passive adversary to determine the group key.

- **Key Independence:** A protocol is said to provide *key independence* (or *known session key security*) if an adversary $\mathcal{A}$ in possession of a proper subset of session keys $\mathcal{K}' \subset \mathcal{K}$ cannot determine any other session key $\widehat{K} \in (\mathcal{K} - \mathcal{K}')$. Key independence is the combination of *forward secrecy*, where an adversary $\mathcal{A}$ in possession of a contiguous subset of past TEKs cannot determine subsequent session keys, and *backward secrecy*, where an adversary $\mathcal{A}$ in possession of a contiguous subset of TEKs cannot determine past TEKs. However, throughout this thesis, we will use the more common definition of forward secrecy, as given in Section 2.2.1.

- **Minimal Trust**: A protocol should not place trust in a high number of entities.

Quality of Service

The following attributes deal with the quality of service attributes of the group key protocol. These properties are desirable, but not necessarily required.

- **1-Affects-n**: A protocol suffers from the *1-affects-n phenomenon* when a single membership change affects all group members. Many protocols require a rekey of the TEK or KEKs and in some extreme cases, the execution of the original group setup algorithm, in response to a membership change.

- **Single Point of Failure**: A protocol suffers from a *single point of failure* when the failure of one group entity causes the entire multicast session to collapse.

Overhead Costs

We would like the group key protocol to be computationally, communication and storage efficient. In particular, we prefer that they do not induce high overhead costs.

- **Computation Overhead**: A protocol has high *computation overhead* if it induces a high computational load for either the key server or the group member.

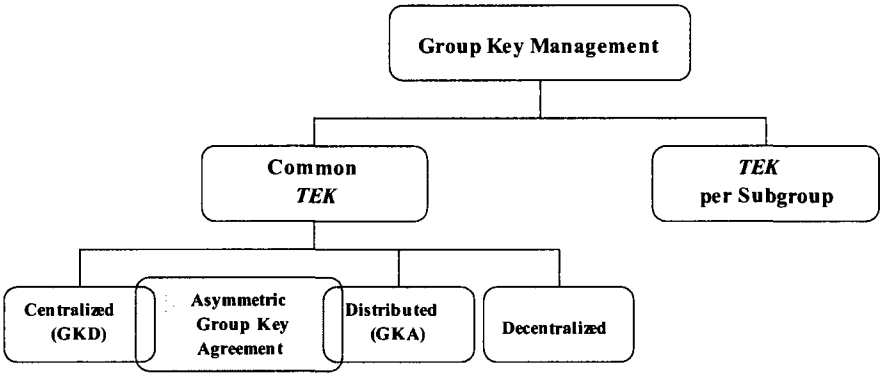

Figure 2.1: Taxonomy of Group Key Management Schemes

- **Bandwidth Overhead**: A protocol has high *bandwidth overhead*, when a rekey operation requires the transmission of a high number of messages.
- **Storage Overhead**: A protocol has high *storage overhead* if it requires that the key server or the group member store a high number of keys.

2.1.3 Classes of Group Key Management Protocols

In [CS05], Challal and Seba provided a taxonomy of the most popular group key management protocols at the time. The authors separate the protocols into two basic classes: the *common* TEK class, in which all group members share the same TEK, and the TEK *per subgroup* class, where the group is divided into subgroups, each with their own separate TEK, as seen in Figure 2.1.

Common TEK Class

The common TEK class can be subdivided into smaller classes based on the management of the shared TEK. These three subclasses are the *centralized*, *decentralized* and *distributed* protocols.

Centralized Approach In the *centralized common TEK* approach, the KS is solely responsible for the generation and distribution of the TEK. While this approach reduces the computational load of the group members, the protocols often suffer from performance bottlenecks and a single point of failure. In addition, placing all

responsibilities of group key management on the KS makes this entity an attractive target for attackers. We more commonly refer to this as the subclass of *group key distribution* (GKD) protocols.

Decentralized Approach In the *decentralized common TEK* approach, the role of the key server is divided amongst a hierarchy of key managers. These protocols are mostly concerned with organization and administration issues of group key management. The hierarchy of key managers helps to reduce bottlenecks and the single point of failure, but introduces the problem of placing trust in a high number of entities.

Distributed Approach In the *distributed common TEK* approach, the members of the group collaborate to construct a group key. The distributed approach eliminates the role of the central key server, reducing bottlenecks and avoiding a single point of failure, but increases the computational load of the group members. The collaborative nature of these protocols assures each entity that no other entity can predetermine the TEK or control the outcome of the TEK, ensuring *key freshness.* Protocols from this subclass are more commonly called *group key agreement* (GKA) protocols and are the central focus of our work.

On the boundary between GKD and GKA protocols, we find the *computationally asymmetric GKA protocols* (or simply asymmetric GKA protocols). Though clumped in with the GKA protocols in [CS05], this type of protocol combines the favourable properties of GKD and GKA protocols, making it particularly noteworthy. In an asymmetric GKA protocol, a group consisting of a stationary host, which we will refer to as the *group leader*, and a cluster of client members, known as *responders*, collaborate to produce the group TEK. Each group member contributes equally to the group TEK, but most of the computational burden is shifted to the group leader. However, we must be sure that they do not suffer from potential drawbacks associated with the group leader, such as performance bottlenecks and single point of failure.

In order to ensure key independence, the common TEK class requires that a new TEK be distributed following a membership change. As a result, many of the protocols discussed above suffer from the 1-affects-n phenomenon and are not as efficient as we would like. A solution to this problem is to arrange the group into a

hierarchy of subgroups, each with their own separate TEK, which is the basis of the TEK per subgroup class of protocol.

The TEK *per Subgroup* Class

In the TEK per subgroup class, the group members are divided into different subgroups, each with their own separate TEK. In order to send a message to the entire group, it must encrypted for each subgroup individually using their subgroup TEK. Using this approach, a membership change will only require a rekey of the TEK for a single subgroup, rather than the entire group, reducing the effects of the 1-affects-n phenomenon to 1-affects-t, where t is the number of members in a given subgroup. However, this introduces a new problem, as the data must be decrypted and re-encrypted at the borders using each respective group's TEK for inter-subgroup communication. This approach requires an elected group leader from each subgroup to perform decryption and re-encryption, increasing computational costs and requiring that the group place trust in a high number of entities. Thus the potential gains of the TEK per subgroup approach may be outweighed by these disadvantages.

2.2 Group Key Agreement

In this section, we elaborate on the concept of group key agreement. Group key agreement is a central area of research in cryptography and consequently, many different protocols satisfying many different properties have been proposed in the literature. In Section 2.2.1, we discuss the security and contributory goals that are specific to group key agreement. In Section 2.2.2, we define the formal communication and security models used for group key agreement. These models will be used to prove the security of our group key agreement protocols in Chapters 6 and 7.

2.2.1 Goals of Group Key Agreement

In general, we would like our protocol to satisfy the requirements of group key management as specified in Section 2.1.2. However, there are several goals that are exclusive to the design of group key agreement protocols. We address these goals in terms of security properties and contributory properties. For a more thorough

discussion, we refer the reader to [AST98], which is an excellent source on the subject of group key agreement.

Security Properties

The following security properties protect the GKA protocol from active adversaries with some additional abilities. They are preferable, but not all necessarily essential to the security of a group key agreement protocol.

- **Implicit Key Authentication:** A protocol is said to provide *implicit key authentication* if a session participant is assured that only the intended participants can possibly learn the value of the session key. In other words, no external party can learn the value of the session key unless aided by a dishonest session participant.
- **Key Confirmation**: A protocol is said to provide *key confirmation* if each participant is assured that they have computed the same session key as all other session participants. A protocol that provides both implicit key authentication and key confirmation is said to provide *explicit key authentication.*
- **Forward Secrecy:** A protocol provides *forward secrecy* (or *forward security*), not to be confused with the definition given in Section 2.1.2, if the disclosure of the long-term private keys of one or more group members does not compromise past session keys. To be thorough, we consider any key that is stored for an extended period of time to be a long-term private key. We have the following forms of forward secrecy, in order of increasing strength:
  - **Partial Forward Secrecy**: A protocol provides *partial forward secrecy* if the long-term private keys of one or more, but not all group members may be disclosed without compromising past session keys.
  - **Perfect Forward Secrecy**: A protocol provides *perfect forward secrecy* if the long-term private keys of all group members may be disclosed without compromising past session keys.
  - **TA Forward Secrecy**: For protocols that require a trusted authority (TA), we say the protocol provides *TA forward secrecy* if the long-term

private key of the TA (and thus the long-term private keys of all group members) may be disclosed without compromising past session keys. This property is particularly applicable in the case of identity-based cryptosystems, discussed in Chapter 5.

- **Key Control Resilience**: A protocol provides *key control resilience* if it is not possible for any entity to predict the value of the session key or force the session key to some predetermined value.

- **Resilience to Man-in-the-Middle Attacks**: A *man-in-the-middle attack* on a key agreement protocol is an attack in which an active adversary E intercepts the communication between entities A and B and modifies the messages so that A and B share a session key with E, rather than with each other. To protect against this type of attack, we must properly authenticate the contributions of the entities A and B using a digital signature scheme, to be discussed in Chapter 4.

- **Unknown Key Share Resilience**: An *unknown key share attack* on key agreement protocols is an attack in which an entity A correctly believes that she shares a session key with an entity B after the execution of the protocol, while the entity B mistakenly believes that he shares the session key with some other entity $E \neq A$. A possible application of this attack is mentioned in [Cho06]. Suppose Alice shares a session key with Bob, but Bob believes he shares the key with Eve. We note that Eve may not have knowledge of the session key. Alice encrypts some valuable information, such as an electronic money transfer, to Bob using the key. Bob believes the encrypted message is from Eve and thus Eve can claim credit for the money transfer. Kaliski describes other ways in which an adversary may exploit this attack in [Kal01].

Contributory Properties

The following properties are specific to contributory group key agreement protocols. A protocol is said to be *contributory* if each party contributes to the session key equally, guaranteeing its freshness.

- **Group Integrity**: A contributory protocol provides *group integrity* if each party is assured of every other party's participation.

- **Verifiable Contributiveness**: A contributory protocol is said to provide *verifiable contributiveness* if each session participant is assured of all other participants' contributions to the session key. Clearly verifiable contributiveness implies group integrity, since assurance of a participant's contribution to the key implies assurance of their participation in the session.

- **Key Integrity**: A contributory protocol is said to provide *key integrity* if each participant is assured that the session key is a function of only the contributions of valid participants.

2.2.2 Formal Model for Group Key Agreement

A common approach to proving the security of cryptographic protocols is to reduce the security of the protocol to some well-known difficult problem in mathematics, thereby requiring an adversary to solve this hard problem in order to break the protocol. The most common security model for two party key establishment is the ideal hash model, more commonly known as the *random oracle model*, proposed by Bellare and Rogaway in [BR93]. Subsequent variations of this model were proposed by Bresson *et al.* in [BR95, BPR00], with a stronger model being proposed by Canetti and Krawczyk in [CK01]. For an analysis of the similarities and differences of these models, we refer the reader to the PhD thesis of Choo in [Cho06].

An *oracle* may be thought of as a theoretical black box that, upon input of some chosen parameters, outputs a value in a single time step. As discussed in [KY03], the random oracle model assumes the existence of some public random function, or *random oracle*, that may be accessed by all participants, including the adversary. The random oracle responds with a truly random value for each new query that is asked, in that it does not reveal any information about the query, and responds with an identical value if that same query is asked again later. The security reduction in this model ultimately depends on the existence of these random oracles.

In recent years, some cryptographers have questioned the validity of security proofs in the random oracle model, as the proofs do not indicate how the random

oracle will be instantiated in the real world. Most often, the random oracle is implemented using a *cryptographic hash function,* which is a one-way function that maps a message of arbitrary length to a fixed length hash value in such a way that it is infeasible to find two messages that produce the same hash value. We note that the design of practical and secure cryptographic hash functions is an area of ongoing research. In particular, the US National Institute of Standards and Technology (NIST) has recently held an open competition for a new cryptographic hash function to replace the older SHA-1 and SHA-2 hash functions. For more information, we refer the reader to [NIS07].

However, standard hash functions do not behave like random oracles and thus an adversary could exploit some weakness or specific feature of the hash function to attack the protocol. This means that the adversary can distinguish between the hash function and the random oracle in the security proof. As a result, some cryptographers have designed their protocols so that they may be proven secure in a stronger model that requires fewer assumptions, known as the *standard model.*

A security proof in the standard model does not rely on the existence of random oracles. While the standard model provides a more realistic environment for security analysis, it is generally accepted that the random oracle model still provides a strong heuristic argument for the security of cryptographic protocol. As stated by Pointcheval in [CCD+05, §4.2.4], "...no one has ever been able to provide a convincing contradiction to its practical validity, but just theoretical counter-examples on either clearly wrong designs for practical purpose, or artificial security notions. Therefore, this model has been strongly accepted by the community, and is considered as a good one, in which security analyses give a good taste of the actual security level."

In this section, we present the model of Bresson *et al.* from [BCPQ01], adapted from the two-party key establishment model of [BPR00], which is the conventional model for authenticated group key establishment and has been widely used to prove the security of GKA protocols, as in [BN03, KLL04, BAA+07]. We use this model to prove the security of our protocols in Chapters 6 and 7.

Group Communication Model

We assume that there is a fixed set of potential participants $\mathcal{U} = \{U_0, U_1, ..., U_\eta\}$, where the number of users is polynomial in the security parameter ℓ. Each user $U_i \in \mathcal{U}$ is associated with a unique identity $\mathsf{ID}_i \in \{0,1\}^*$. We allow a user to execute the protocol several times with different users. The model accounts for this by allowing different *instances* of a user, known as oracles, involved in distinct, but possibly concurrent, executions of the protocol. We denote instance α of a user U_i as Π_i^α with $\alpha \in \mathbb{N}$. We refer to a participant in general as user U and denote the α^{th} instance by Π_U^α. Before the protocol is executed for the first time, the KS runs some key generation algorithm $\mathcal{G}(1^\ell)$ and distributes long-term public/private key pairs to each user. We note that the public key set of all users is known by all parties, including the adversary.

We assume that every message is broadcasted to all users in the session. As a result, we define the notion of partnering as it is given in [BCPQ01], using *session IDs* and *partner IDs*. A session ID for an oracle Π_U^α is denoted sid_U^α and is equal to the concatenation of all messages that are sent and received by instance Π_U^α in the execution of the protocol. A partner ID for an oracle Π_U^α is denoted pid_U^α and consists of the identities of all the users establishing a key in the α^{th} session. We say that oracles Π_i^α and Π_j^β are *partnered* if, and only if, $\mathsf{sid}_i^\alpha = \mathsf{sid}_j^\beta$ and $\mathsf{pid}_i^\alpha = \mathsf{pid}_j^\beta$.

Adversarial Model

The standard approach to proving the security of a cryptographic protocol is to prove that an adversary has no advantage in breaking a simulation of the protocol. In the two-party setting of [BR93, BPR00, CK01], we define the simulation in terms of a "security game" between an adversary and a challenger. However, since we are working in a group setting, as in [BCPQ01], we define the simulation in terms of a game between an adversary $\mathcal{A}$ and an infinite set of oracles $\{\Pi_U^\alpha\}$, $U \in \mathcal{U}$ and $\alpha \in \mathbb{N}$. The general concept behind this approach is as follows: the oracles (or challenger in the two-party case) run the protocol simulation for the adversary; the oracles embed some hard mathematical problem into the simulation so that the adversary must successfully solve the hard problem in order to break the protocol; if at any point, the adversary realizes that she is experiencing a simulation and not the real world execu-

tion of the protocol, the game aborts and the adversary wins; otherwise, we reduce the adversary's advantage in breaking the protocol to the adversary's advantage in solving the problem.

In the simulation, we allow the adversary to control all communication between instances of users. Since we want our simulation to be as close to the real world as possible, we allow the adversary to have certain capabilities that correspond to real world abilities or attacks. We model these capabilities by allowing the adversary to issue the following queries (discussed in [BCPQ01, KY03, Cho06]) to the oracles:

Send(Π_U^α, M) The Send query sends the message M to instance α of user U and returns the response generated by the oracle. The message $M =$"`init`" indicates that user U should initiate the protocol. For protocols in which members assume different roles, the message $M =$ "`init` $\|$ `role`" indicates that user U should initiate the protocol for the specified role.

The Send query models the adversary's ability to make an instance run the protocol normally. This captures the adversary's ability to perform active attacks by modifying or inserting messages during the execution of the protocol, allowing for man-in-the-middle or impersonation attacks.

Execute($U_{i_1}, ..., U_{i_n}$) The Execute query executes the protocol for a group of users $\{U_{i_1}, ..., U_{i_n}\}$, chosen by the adversary, and returns the transcript of the execution.

The Execute query models a passive adversary's ability to eavesdrop on an execution of the protocol.

Reveal(Π_U^α) If oracle Π_U^α has accepted the session, it returns the session key K.

The Reveal query models attacks which reveal the group session key and can be used to capture the idea of key independence (see Section 2.1.2) in the security model.

Corrupt(U) The Corrupt query returns the private key of user U, but does not reveal any internal data of any instance of U.

The Corrupt query models adversary attacks which reveal the long-term private key of a user $U \in \mathcal{U}$. As mentioned in [Cho06], this query captures the notion of unknown key share and insider attacks. Furthermore, this query can be used to model forward secrecy in the protocol, depending on when the adversary may issue the query in the security game.

Test(Π_U^α) The Test query generates a random bit $b \in \{0,1\}$; if $b = 1$, oracle Π_U^α returns the session key K to the adversary and if $b = 0$ it returns a random session key.

The Test query is the only query that does not correspond to any real world event or capture any capabilities of the adversary. This query models the semantic security of a session key. The adversary may only ask this query one time throughout the simulation.

The security model allows for both passive and active adversaries (see Section 2.1.2). A passive adversary is given access to the Execute, Reveal, Corrupt and Test queries. An *active adversary* is additionally given access to the Send query. We note that the Execute query can be simulated through multiple calls to the Send query. Before describing the security game in detail, we consider the following definition.

Definition 2.1 *An oracle Π_i^α is said to be* fresh *(or hold a fresh session key K) if it satisfies the following properties:*

(i) *Oracle Π_i^α has accepted the session key K and the adversary did not issue the queries* Reveal(Π_i^α) *or* Reveal(Π_j^β) *for any oracle Π_j^β partnered with Π_i^α.*

(ii) *The adversary has not issued a* Corrupt *query before a $Send(\Pi_i^\alpha, *)$ or $Send(\Pi_j^\beta, *)$ query, for any oracle Π_j^β partnered with Π_i^α.*

Security Game The security of a protocol $\mathcal{P}$ is defined in terms of the following game between the adversary $\mathcal{A}$ and an infinite set of oracles $\{\Pi_U^\alpha\}$, for $U \in \mathcal{U}$ and $\alpha \in \mathbb{N}$:

Phase 1: The adversary $\mathcal{A}$ issues queries to the challenger. Afterwards, $\mathcal{A}$ issues the Test query on a fresh oracle Π_i^α.

Challenge: The challenger C responds with the session key K_b as per the Test query.

Phase 2: $\mathcal{A}$ continues to query the oracles, but may not issue the Reveal query for oracle Π_i^α or any oracle Π_j^β partnered with Π_i^α.

Response: The adversary outputs a guess b' as to the value of the bit b. The adversary wins the game if $b' = b$.

We measure the adversary's advantage in the security game by her ability to distinguish between the session key and a random value. The advantage of $\mathcal{A}$ in attacking $\mathcal{P}$ is defined as

$$\mathrm{Adv}_{\mathcal{A},\mathcal{P}}(\ell) = |2 \cdot \Pr[b' = b] - 1|$$

for the security parameter ℓ. Given this definition, we can formally define the notion of secure authenticated group key agreement.

Definition 2.2 *We say that a protocol $\mathcal{P}$ is a* secure authenticated group key agreement (AGKA) protocol *if it satisfies the following properties:*

(i) Validity: *Partner oracles Π_i^α and Π_j^β accept the same session key in the presence of a passive adversary $\mathcal{A}$.*

(ii) Indistinguishability: *The advantage $\mathrm{Adv}_{\mathcal{A},\mathcal{P}}(\ell)$ of any probabilistic polynomial time (PPT) active adversary $\mathcal{A}$ is negligible.*

Remark 2.2 *We say that an algorithm is a* polynomial time algorithm *if its running time is bounded by a polynomial in the size of the input parameters. While some algorithms output a unique value for each set of input parameters,* probabilistic algorithms *may perform coin tosses in the computation of the output, which may not necessarily be unique. A* probabilistic polynomial time algorithm *is a probabilistic algorithm whose expected running time (as averaged over all coin tosses) is bounded by a polynomial in the size of the input parameters. For more information, we refer the reader to [Pom97].*

We will use the security game simulation approach described in this section to define the security model for encryption schemes in Section 3.3.2, signature schemes in Section 4.1.2 and signcryption schemes in Section 4.3.4. Furthermore, we use the group key agreement security model presented here to prove the security of the protocols proposed in Chapters 6 and 7.

2.3 A Survey of Group Key Agreement Protocols

The Diffie-Hellman (DH) key exchange protocol, proposed by Diffie and Hellman in [DH76], was a pioneering development in public key cryptography, allowing Alice and Bob to agree upon a shared session key without the use of a secure channel of communication. Many of the prominent group key agreement protocols from the literature are based on the now classic DH key exchange protocol. In this section, we provide a brief literature survey of what we deem to be some of the most influential and creative group key agreement protocols, as the basic structure of these protocols have been adopted by several other protocols from the literature.

In Section 2.3.1, we establish the notation used throughout the section and define the security problems which serve as the basis for the cryptographic protocols. In Section 2.3.2, we give a short summary of the DH key exchange protocol to provide the necessary background for the unfamiliar reader. For the remainder of the section, we provide a detailed examination of some of the more popular group key agreement protocols, several of which are included in the survey of Distributed Common *TEK* protocols in [RH03, CS05]. In particular, we look at the Group Diffie-Hellman protocol from [STW96] in Section 2.3.3, the Tree-based Group Diffie-Hellman protocol from [KPT00] in Section 2.3.4, the Burmester-Desmedt protocol from [BD94] in Section 2.3.5, the Conference Key Agreement protocol from [Boy97, BN03] in Section 2.3.6 and the Asymmetric Group Diffie-Hellman protocol from [BAA+07] in Section 2.3.7. This investigation will be helpful in Chapter 5, where we'll see how cryptographers applied pairings to produce new versions of several of these GKA protocols.

2.3.1 Preliminary Definitions

Throughout the rest of this thesis, we let $\mathbb{G}$ be a cyclic group of prime order q with generator g and for any group $\mathbb{G}$ (either additive or multiplicative), we let $\mathbb{G}^\times$ denote the set obtained by removing the identity element from $\mathbb{G}$. For example, $\mathbb{Z}_q = \{0, 1, 2, ..., q-1\}$ is a group under addition modulo q and $\mathbb{Z}_q^\times$ denotes the set $\{1, 2, ..., q-1\}$, which is a group of order $q-1$ under multiplication modulo q since q is prime. Furthermore, we use the notation $x \in_R \mathbb{G}$ to denote that the element x is chosen randomly from the set of elements in $\mathbb{G}$. We use this notation to express the following definitions.

Definition 2.3 *Suppose $\mathbb{G}$ is a cyclic group of prime order q with generator g. That is, for each $h \in \mathbb{G}$, there exists some $x \in \mathbb{Z}_q$ such that $h = g^x$. Given $g, g^x \in \mathbb{G}^\times$ for some $x \in_R \mathbb{Z}_q^\times$, the* discrete logarithm problem (DLP) *is to determine x.*

Definition 2.4 *Suppose $\mathbb{G}$ is a cyclic group of prime order q with generator g. Given $g, g^x, g^y \in \mathbb{G}^\times$ for some $x, y \in_R \mathbb{Z}_q^\times$, the* (computational) Diffie-Hellman problem (DHP) *is to determine g^{xy}.*

Definition 2.5 *Suppose $\mathbb{G}$ is a cyclic group of prime order q with generator g. Given $g, g^x, g^y, g^z \in \mathbb{G}^\times$ for some $x, y, z \in_R \mathbb{Z}_q^\times$, the* Decisional Diffie-Hellman problem (DDHP) *is to distinguish between the tuples (g, g^x, g^y, g^{xy}) and (g, g^x, g^y, g^z).*

It is understood that these are hard problems in mathematics for the average case, by which we mean that there exists no polynomial time algorithm that can solve the problem with non-negligible probability on random inputs. Consequently, they serve as the basis for many cryptosystems, including all of the group key agreement protocols in this literature survey, with the exception of the Conference Key Agreement protocol of Section 2.3.6.

Finally, as commonly practiced in the literature, we use the term hash function to refer to a cryptographic hash function and we will not concern ourselves with the issue of existence of these functions or their specific implementations in practical applications.

2.3.2 The Diffie-Hellman Key Exchange Protocol

Overview

Suppose we have the public parameters $\langle \mathbb{G}, g, q \rangle$ for which the DLP is hard. Alice and Bob execute the DH key exchange protocol in a single round as follows:

1. Alice chooses $a \in_R \mathbb{Z}_q^\times$ and computes $A = g^a$. The value A serves as Alice's public key, while a is her private key. Similarly, Bob chooses $b \in_R \mathbb{Z}_q^\times$ and computes $B = g^b$ to obtain the public/private key pair (B, b).

2. Alice sends her public key A to Bob, while Bob sends his public key B to Alice.

3. Upon receiving B from Bob, Alice computes $B^a = g^{ab}$. Similarly, Bob computes $A^b = g^{ab}$. Alice and Bob now share the secret key $K = g^{ab}$.

In order to compute the shared secret key K, an adversary Eve must recover the private key of either Alice or Bob, thus computing a from g^a or b from g^b, or compute K directly using the public tuple (g, g^a, g^b). We see that these methods correspond to the solving the DLP and DHP respectively, and, assuming that these problems are intractable, it is infeasible for Eve to recover K.

The MQV Protocol

We note that the original DH key exchange does not address the issue of authentication and is vulnerable to a man-in-the-middle attack. In [LMQ+98, LMQ+03], Law *et al.* proposed an efficient authenticated key agreement protocol, known as the MQV protocol, based on the DH key exchange. The general idea is that Alice and Bob have respective long-term public/private key pairs (A, a) and (B, b), as in the original DH key exchange, and also choose ephemeral public/private key pairs (X, x) and (Y, y), respectively, where $X = g^x$ and $Y = g^y$ for $x, y \in_R \mathbb{Z}_q^\times$. Using the long-term and ephemeral keys, Alice and Bob compute the session key as

$$K = (Y \cdot B^e)^{(x+ad)} = (X \cdot A^d)^{(y+be)} = g^{(x+da)(y+be)},$$

where d is a function of x and e is a function of y. For simplicity, we leave out specific details of the protocol, including discussion of the computation of d and e and the

cofactor h, and refer the reader to [LMQ+98, LMQ+03] for more information.

The ingenuity of the MQV protocol lies in the use of long-term and ephemeral secrets in the computation of the session key. This allows the protocol to provide authentication without increasing the communication costs of the original DH protocol. In addition, the MQV protocol provides partial forward secrecy, since the disclosure of the long-term private key of either party, but not both, does not compromise the session key.

For each of the remaining protocols, we assume that a group of $n > 2$ members, $\mathcal{U} = \{U_1, ..., U_n\}$ wish to agree upon a shared session key. With the exception of the protocol from 2.3.6, each protocol extends the DH key exchange to n parties by arranging the members in some unique structure. For simplicity, we do not address the issue of authentication for these DH-based protocols.

2.3.3 The Group Diffie-Hellman Protocol

Overview

The Group Diffie-Hellman (GDH) protocol, proposed by Steiner *et al.* in [STW96] extends the two party Diffie-Hellman key exchange to an n-party group by arranging the members in a ring structure, similar to the Ingemarsson *et al.* protocol from [ITW82]. In the protocol, we use the term *cardinal value* (not to be confused with cardinal numbers used to measure the size of a set) as given in [RH03, CS05], to refer to the last element of the set.

Each member chooses an ephemeral value $k_i \in_R \mathbb{Z}_q^\times$. Member U_1 computes g^{k_1} and sends $\{g, g^{k_1}\}$ to U_2. Member U_2 raises these intermediate values to his secret exponent k_2 and sends $\{g^{k_2}, g^{k_1}, g^{k_1 k_2}\}$ to U_3. Continuing in this manner, each member U_i receives a set of intermediate values from U_{i-1}, containing the cardinal value $g^{k_1 k_2 ... k_{i-1}}$. Member U_i generates a new set by raising all previous intermediate values to his exponent k_i and entering the cardinal value $g^{k_1 k_2 ... k_{i-1}}$ of the previous set in the position just before the current cardinal value $g^{k_1 k_2 ... k_{i-1} k_i}$. Member U_i then sends this new set to U_{i+1}. The last member U_n receives a set from U_{n-1} and raises all values to his exponent k_n to generate a new set. The cardinal value of this set is the group key $K = g^{k_1 k_2 ... k_n}$. Member U_n broadcasts the new set of intermediate values (excluding the group key) to the remaining group members. Upon receiving

this new set, each member U_i extracts the i^{th} intermediate value and raises it to his secret exponent k_i to obtain the shared session key, K. We present an example to further clarify this protocol.

Example

Suppose we have a group of 4 members, $\mathcal{U} = \{U_1, ..., U_4\}$. Each group member U_i chooses a value $k_i \in_R \mathbb{Z}_q^\times$.

1. Member U_1 sends $\{g, g^{k_1}\}$ to U_2.
2. Member U_2 sends $\{g^{k_2}, g^{k_1}, g^{k_1 k_2}\}$ to U_3.
3. Member U_3 sends $\{g^{k_2 k_3}, g^{k_1 k_3}, g^{k_1 k_2}, g^{k_1 k_2 k_3}\}$ to U_4.
4. Member U_4 computes $\{g^{k_2 k_3 k_4}, g^{k_1 k_3 k_4}, g^{k_1 k_2 k_4}, g^{k_1 k_2 k_3}, g^{k_1 k_2 k_3 k_4}\}$ and retrieves the key $K = g^{k_1 k_2 k_3 k_4}$.
5. Member U_4 broadcasts $\{g^{k_2 k_3 k_4}, g^{k_1 k_3 k_4}, g^{k_1 k_2 k_4}, g^{k_1 k_2 k_3}\}$ to the remaining group members.
6. Each member U_i extracts the i^{th} value and raises it to their k_i to compute the key. For example, member U_2 extracts the 2^{nd} value $g^{k_1 k_3 k_4}$ and raises it to k_2 to compute the key $K = g^{k_1 k_2 k_3 k_4}$.

Analysis

In the Ingemarsson *et al.* protocol, a member U_i receives a value from U_{i-1}, raises it to the exponent k_i and then passes it along to U_{i+1}. To obtain the group key $K = g^{k_1 k_2 \cdots k_n}$, the protocol requires $n - 1$ rounds, in which each member performs 1 exponentiation and sends 1 unicast message, for a total of n exponentiations per member and a group total of $n(n-1)$ unicast messages. The GDH protocol reduces the computational and communication complexity of the Ingemarsson *et al.* protocol by having member U_i raise the set of contributions received from U_{i-1} to the exponent k_i and sending this set, along with his own contribution, to member U_{i+1}. While the GDH protocol requires an extra round of communication (for n rounds in total), it requires only n broadcast and $n - 1$ unicast messages in total and each member to

compute *at most* n, but as little as 2 exponentiations, depending on their position in the ring.

While the GDH protocol offers a number of improvements over the Ingemarsson *et al.* protocol, both fail to provide forward and backward secrecy without re-executing the inital key agreement protocol and are thus unsuitable for dynamic broadcast groups.

2.3.4 The Tree-based Group Diffie-Hellman Protocol

Overview

The Tree-based Group Diffie-Hellman (TGDH) protocol was proposed by Kim *et al.* in [KPT00, KPT04] as a variant of the skinny tree (STR) protocol [SSDW90, KPT01]. The protocol extends the classic two-party Diffie-Hellman key exchange protocol to a group of n parties by arranging the group members into a balanced binary tree, with each member associated with a leaf of the tree. Each non-root node i of the tree is associated with a secret key k_i and a public *blinded key* $bk_i = g^{k_i}$, as seen in Figure 2.2. The root node is associated with the group *TEK*. The group *TEK* is built recursively from the bottom up, with each secret node key given by

$$k_i = \left(bk_{left(i)}\right)^{k_{right(i)}} = \left(bk_{right(i)}\right)^{k_{left(i)}} = g^{k_{left(i)}k_{right(i)}}, \tag{2.3.1}$$

where $left(i)$ and $right(i)$ represent the left and right child nodes of the i^{th} node. Each member in the group knows the secret node keys along the path from his own leaf node to the root node, known as the member's *key path.*

We note that Equation (2.3.1) used to compute the secret node keys is simply the DH key exchange protocol. As a result, we must have $bk_{left(i)}, bk_{right(i)} \in \mathbb{G}$ (i.e. blinded keys correspond to elements of $\mathbb{G}$) and $k_{left(i)}, k_{right(i)} \in \mathbb{Z}_q^\times$ (i.e. secret node keys correspond to elements of $\mathbb{Z}_q^\times$). However, using Equation (2.3.1) to compute the secret node key, we obtain an element of $\mathbb{G}$ rather than $\mathbb{Z}_q^\times$, which introduces a problem whenever $\mathbb{G} \neq \mathbb{Z}_q^\times$. As a solution, we use some hash function $H : \mathbb{G} \rightarrow \mathbb{Z}_q^\times$ in our computations whenever the input secret node key corresponds to an element of $\mathbb{G}$. For example, we would have blinded keys $bk_{12} = g^{H(g^{k_1 k_2})}$ and $bk_{34} = g^{H(g^{k_3 k_4})}$,

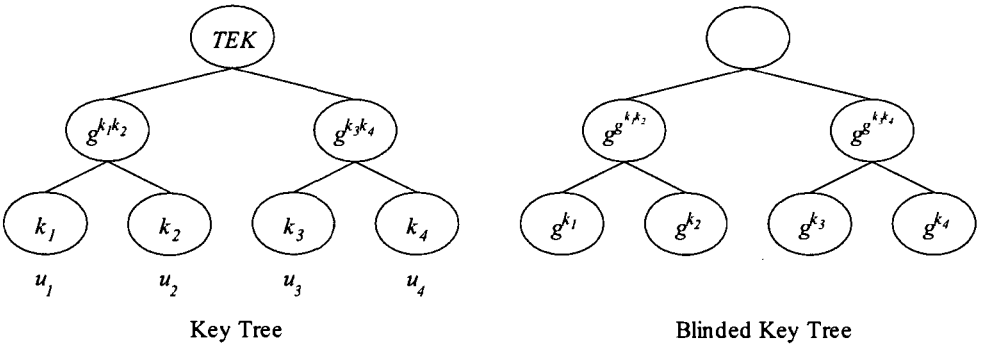

Figure 2.2: TGDH Key Trees

rather than $g^{g^{k_1k_2}}$ and $g^{g^{k_3k_4}}$, in Figure 2.2.

For simplicity, we explain how the protocol works with the following example.

Example

Suppose we have a group of 4 members, $\mathcal{U} = \{U_1, ..., U_4\}$. Each group member U_i chooses a value $k_i \in_R \mathbb{Z}_q^\times$. The members compute the secret key and blinded key trees from Figure 2.2 as follows:

1. Each member U_i computes and broadcasts their blinded key $bk_i = g^{k_i}$.

2. Member U_1 receives $bk_2 = g^{k_2}$ from U_2 and computes $k_{12} = \left(g^{k_2}\right)^{k_1} = g^{k_1k_2}$ using (2.3.1). Similarly, member U_2 computes $k_{12} = \left(g^{k_1}\right)^{k_2}$. Of $\{U_1, U_2\}$, the member associated with the left-most child node (i.e. U_1) computes and broadcasts the blinded key $bk_{12} = g^{H(g^{k_1k_2})}$. Similarly, members U_3 and U_4 compute k_{34} and U_3 broadcasts $bk_{34} = g^{H(g^{k_3k_4})}$.

3. Members U_1 and U_2 receive $bk_{34} = g^{H(g^{k_3k_4})}$ from U_3 and compute the group TEK as

$$K = (bk_{34})^{H(g^{k_1k_2})} = \left(g^{H(g^{k_3k_4})}\right)^{H(g^{k_1k_2})} = g^{H(g^{k_1k_2})\cdot H(g^{k_3k_4})}.$$

   Members U_3 and U_4 compute the group TEK similarly, using bk_{12} and k_{34}.

Analysis

By arranging the group members in the GDH protocol into a tree structure, the authors were able to reduce the key calculations from $O(n)$ to $O(\log_2 n)$. Assuming that n members are arranged in a balanced binary tree, the initial key agreement protocol requires a total of $2n - 2$ broadcast messages, sent throughout h rounds, where $h = \lceil \log_2 n \rceil$ is the height of the tree. In addition, the unique structure of the protocol allows for reduced communication and computational costs in auxiliary key agreement. Indeed, Kim *et al.* propose different auxiliary key agreement protocols for join, leave, multiple join and multiple leave operations with varying costs. For more information, we refer the reader to [KPT04].

Finally, we note that the TGDH protocol does not consider key authentication, but rather assumes that all messages are signed using some secure public key signature scheme.

2.3.5 The Burmester-Desmedt Protocol

Overview

In [BD94], Burmester and Desmedt proposed an efficient GKA protocol based on the DH key exchange, which we call the BD protocol. The BD protocol runs in the following two rounds:

Round 1 Each member $U_i \in$U chooses $k_i \in_R \mathbb{Z}_q^\times$ and broadcasts $Z_i = g^{k_i}$.

Round 2 Upon receiving Z_{i-1} and Z_{i+1} from U_{i-1} and U_{i+1}, respectively, each U_i broadcasts $X_i = (Z_{i+1}/Z_{i-1})^{k_i}$.

Key Computation Each member U_i can now compute the shared secret key

$$\begin{aligned} K_i &= (Z_{i-1})^{nk_i} \cdot \prod_{j=0}^{n-1} X_{i+j \bmod n}^{n-j-1 \bmod n} \\ &= (Z_{i-1})^{nk_i} X_i^{n-1} X_{i+1}^{n-2} \ldots X_{i-2}^{1} X_{i-1}^{0} \\ &= g^{k_1k_2 + k_2k_3 + \ldots + k_{n-1}k_n + k_nk_1}. \end{aligned}$$

We illustrate the correctness of the key computation phase with the following example.

Example

Suppose we have a group of 4 members $\mathcal{U} = \{U_1, ..., U_4\}$. Each member U_i chooses a value $k_i \in_R \mathbb{Z}_q^\times$. Member U_2 computes the final session key as

$$\begin{aligned}
K_2 &= (Z_1)^{4k_2} \cdot X_2^3 \cdot X_3^2 \cdot X_4 \\
&= (Z_1)^{4k_2} \cdot \left(\frac{Z_3}{Z_1}\right)^{3k_2} \cdot \left(\frac{Z_4}{Z_2}\right)^{2k_3} \cdot \left(\frac{Z_1}{Z_3}\right)^{k_4} \\
&= g^{4k_1k_2} \cdot \frac{g^{3k_2k_3}}{g^{3k_1k_2}} \cdot \frac{g^{2k_3k_4}}{g^{2k_2k_3}} \cdot \frac{g^{k_4k_1}}{g^{k_3k_4}} \\
&= \frac{g^{4k_1k_2}}{g^{3k_1k_2}} \cdot \frac{g^{3k_2k_3}}{g^{2k_2k_3}} \cdot \frac{g^{2k_3k_4}}{g^{k_3k_4}} \cdot g^{k_4k_1} \\
&= g^{k_1k_2} \cdot g^{k_2k_3} \cdot g^{k_3k_4} \cdot g^{k_4k_1} \\
&= g^{k_1k_2+k_2k_3+k_3k_4+k_4k_1}.
\end{aligned}$$

One can easily verify that members U_1, U_3 and U_4 compute the same session key.

Analysis

Although this protocol is quite computationally efficient, each member is required to send 2 broadcast messages for key agreement, for a total of $2n$ broadcast messages. In addition, since there are no protocols for auxiliary key agreement, the initial key agreement protocol must be executed following a membership change, inducing additional communication costs. In [KY03], Katz and Yung proposed a compiler to transform secure key establishment protocols to secure authenticated key establishment protocols. To illustrate their compiler, they produced a secure authenticated version of the BD protocol as an example.

2.3.6 The Conference Key Agreement Protocol

Overview

In [Boy97] and [BN03], Boyd (and González Nieto) proposed computationally asymmetric GKA protocols that are not based on the Diffie-Hellman key exchange, which they call Conference Key Agreement (CKA) protocols. We slightly alter the presentation of their protocol to reflect the structure of computationally asymmetric GKA protocols to be discussed in future sections.

The multicast group $\mathcal{U}$ consists of a group of responders $\mathcal{R} = \{U_1, ..., U_n\}$ and a group leader U_0. The protocol is associated with some public key encryption scheme $(\mathcal{E}, \mathcal{D})$, with encryption algorithm $\mathcal{E}$ and decryption algorithm $\mathcal{D}$, and signature scheme $(\mathcal{S}, \mathcal{V})$, with signing algorithm $\mathcal{S}$ and verification algorithm $\mathcal{V}$. Each member U_i possesses a public/private encryption key pair (e_i, d_i) and a public/private signature key pair (e_i', d_i') and chooses a random ephemeral value N_i that will serve as their key contribution. The CKA protocol (as given in [BN03]) runs in a single two-step round as follows:

Round 1

i) The group leader computes $\mathcal{E}_{e_i}(N_0)$ to encrypt her contribution N_0 for each responder U_i using their public key e_i. She computes and broadcasts

$$\left\langle \mathcal{U}, \mathcal{E}_{e_1}(N_0), ..., \mathcal{E}_{e_n}(N_0), \mathcal{S}_{d_0'}\left(\mathcal{U} \parallel \mathcal{E}_{e_1}(N_0) \parallel ... \parallel \mathcal{E}_{e_n}(N_0)\right)\right\rangle$$

to the group of responders $\mathcal{R}$.

ii) Each responder broadcasts their contribution N_i in the clear.

Key Computation Each responder $U_i \in \mathcal{R}$ verifies $\mathcal{V}_{e_0'}\left(\mathcal{U} \parallel \mathcal{E}_{e_1}(N_0) \| ... \| \mathcal{E}_{e_n}(N_0)\right)$. If the verification holds, they use their private key d_i to compute $\mathcal{D}_{d_i}\left(\mathcal{E}_{e_i}(N_0)\right)$ and recover the leader's contribution N_0.

Each member in U computes the group *TEK* as

$$K = H\left(N_0 \parallel N_1 \parallel ... \parallel N_n\right),$$

using some cryptographic hash function H.

Analysis

The CKA protocol is computationally efficient, requiring only n encryptions and 1 signature for the leader and 1 decryption and 1 verification for each responder. We note that the leader can send the $n-1$ encryptions $\mathcal{E}_{e_i}(N_0)$ and the signature in a single broadcast message. In addition, there is no need for the leader's messages to be sent before the responders', so they can all be sent in a single round. Thus the CKA protocol is very communication efficient, requiring a total of $n+1$ broadcast messages in a single broadcast round.

The biggest drawback of the CKA protocol is that it does not provide perfect forward secrecy. Indeed, if the long-term private key of a responder is disclosed to Eve, she can recover the leader's secret contribution, and thus the group TEK, for every session in which that responder was a participant. In addition, the protocol does not require authentication of the responder contributions. As a result, the group members cannot be assured that every member has contributed to the session key or that no extraneous contributions have been introduced, precluding the properties of verifiable contributiveness and key integrity, respectively.

Furthermore, the CKA protocol was shown to be vulnerable to an unknown key share attack by Choo in [Cho06]. Indeed, suppose an adversary $\mathcal{A}$ intercepts the leader's broadcast message and replaces it with

$$\left\langle \mathcal{U}_{\mathcal{A}}, \mathcal{E}_{e_1}(N_0), ..., \mathcal{E}_{e_n}(N_0), \mathcal{S}_{d'_{\mathcal{A}}}\left(\mathcal{U}_{\mathcal{A}} \| \ \mathcal{E}_{e_1}(N_0) \ \| \ ... \ \| \ \mathcal{E}_{e_n}(N_0)\right)\right\rangle,$$

where $\mathrm{U}_{\mathcal{A}} = \mathcal{R} \cup \{\mathcal{A}\}$. Each responder $U_i \in \mathcal{R}$ verifies that the signature on the set of encrypted values belongs to $\mathcal{A}$, recovers the encrypted contribution N_0 and then broadcasts their key contribution N_i. The adversary forwards these key contributions to the group leader. Each member computes the session key as usual. While the leader U_0 shares this key with the group of responders $\mathcal{R}$, the responders mistakenly believe that they share it with the adversary $\mathcal{A}$.

2.3.7 The Asymmetric Group Diffie-Hellman Protocol

Overview

In [BAA+07], Bhaskar *et al.* proposed the Asymmetric Group Diffie-Hellman (AGDH) protocol, which is actually a variation of Boyd's CKA protocol that uses the DH key exchange. Similar variations of this type have been proposed in [BCEP03, KLL04, NLKW05, KKC+06, Tse07]. As with the CKA protocol, we have a group leader U_0 and a group of responders $\mathcal{R}$. To keep our discussion consistent with the previous DH-based protocols, we present the AGDH protocol in terms of key agreement only. For the details of the protocol involving initialization and authentication, we refer the reader to [BAA+07].

Round 1 Each responder $U_i \in \mathcal{R}$ chooses $k_i \in_R \mathbb{Z}_q^\times$, computes the blinded key g^{k_i} and sends it to the group leader.

Round 2 The group leader collects all the blinded keys and chooses $k_0 \in_R \mathbb{Z}_q^\times$. For each responder $U_i \in \mathcal{R}$, the group leader computes $\left(g^{k_i}\right)^{k_0}$ and broadcasts them along with the original blinded key, as

$$\left\langle \left\{U_1, g^{k_1}, g^{k_0 k_1}\right\}, \left\{U_2, g^{k_2}, g^{k_0 k_2}\right\}, ..., \left\{U_n, g^{k_n}, g^{k_0 k_n}\right\} \right\rangle.$$

Key Computation Each responder U_i computes $\left(g^{k_0 k_i}\right)^{k_i^{-1}}$ to recover the leader's contribution g^{k_0}. Each member computes the group *TEK* as

$$K = g^{k_0} \cdot \prod_{i=1}^{n} g^{k_0 k_i} = g^{k_0(1+k_1+...+k_n)}.$$

Analysis

The AGDH protocol requires minimal communication and computational costs. In addition, the AGDH protocol provides auxiliary key agreement protocols to handle membership changes and scenarios for election of a group leader. The authors claim that the protocol provides perfect forward secrecy. However, auxiliary key agreement requires the responders to reuse their ephemeral secret from one session to the next. If members are required to store their secret value k_i for an extended period of time,

it can no longer be considered an ephemeral value. Since disclosure of the value k_i will compromise previous session keys, their protocol does not provide perfect forward secrecy as we have defined it in Section 2.1.2.

2.4 Summary

Throughout this chapter, we discussed the components and goals involved in key management for dynamic multicast groups and the various solutions to group key establishment. In particular, we investigated the concept of contributory group key agreement, including a discussion on the properties and formal communication and security models, and provided a survey of several different classes of group key agreement schemes based on the classic Diffie-Hellman key exchange protocol.

In the next chapter, we define the concept of a bilinear pairing and show how its unique properties allow the pairing to be used as a building block for cryptographic applications, including the recent development of identity-based cryptography. In Chapter 5, we'll see how cryptographers applied the bilinear pairing to the DH-based group key agreement models presented in this chapter to produce fast, efficient and secure pairing-based group key agreement protocols.

Chapter 3

Introduction to Pairings and Identity-based Cryptography

The objective of this chapter is to introduce the reader to bilinear pairings and their influence on cryptographic research. Until recently, bilinear pairings had played a rather small role in cryptography. The first practical use of pairings in cryptography was in [Kal87], where Kaliski used the Weil pairing (see Section A.3.2) to construct a pseudo-random bit generator based on elliptic curves. For the most part, pairings played a destructive role, as in [MOV93, FMR99], where they are used to attack elliptic curve cryptography by reducing the complexity of the discrete logarithm problem for some "weak" elliptic curves. However, in 2000 pairings earned a more positive reputation in cryptography, used independently by Joux [Jou00] to construct a three-participant version of the Diffie-Hellman key exchange and by Sakai *et al.* [SOK00] to construct a non-interactive key distribution scheme (NIKDS), although the latter protocol was not very well known at the time.

While Joux's protocol set the wheels of pairing-based cryptography in motion, the most important and influential contribution to pairing-based cryptography is arguably the identity-based encryption (IBE) scheme of Boneh and Franklin from [BF01, BF03]. The Boneh and Franklin IBE scheme caused a flurry of interest in pairing- and identity-based cryptography. Shortly after their paper was published, the first ID-based key agreement protocol was proposed by Smart in [Sma01], a number of ID-based signature schemes were proposed in [Pat02, Hes02, Hes03, CC03] and

the first ID-based signcryption scheme was proposed by Malone-Lee in [ML02].

In Section 3.1, we define the notion of a bilinear pairing as a cryptographic building block, which can be used to construct efficient pairing-based cryptosystems, and discuss the corresponding assumptions upon which the security of our cryptosystems will depend. As the main goal of our work is to discuss the applications of pairings and their use in group key agreement, we will treat pairings as "black boxes", focussing on their cryptographic properties rather than the mathematics behind them. This is a common practice in pairing- and identity-based cryptography, as it allows us to "concentrate on the general cryptographic principles behind the schemes and systems we study, without being distracted by the implementation details", as mentioned by Paterson in [BSe05, §X.1.2]. However, in Appendix A, we "open up the box" and provide a thorough discussion of the mathematical concepts used to define and compute pairings.

The remainder of the chapter is organized in such a way that we underscore the evolution of pairing-based cryptography, from its origins to its most powerful and interesting application, identity-based cryptography. We emphasize three applications of pairings in particular. In Section 3.2, we discuss Joux's tripartite Diffie-Hellman protocol, one of the pioneering works in pairing-based cryptography. We discuss the concept of identity-based cryptography at large in Section 3.3 and define the generalized algorithms that comprise an IBE scheme and the formal model used to prove the security of the scheme. In Section 3.4, we apply the concept of pairings to the generalized IBE algorithms to show how Boneh and Franklin constructed the first practical identity-based encryption scheme. We combine the ideas of Joux and Boneh and Franklin to construct Smart's identity-based key agreement protocol in Section 3.5 and give a summary highlighting the important concepts of the chapter in Section 3.6.

3.1 Preliminary Definitions

3.1.1 Pairings

Let $\mathbb{G}$ be an additive group with identity element $\mathcal{O}$. We write $\mathbb{G}^\times$ for $\mathbb{G} \setminus \{\mathcal{O}\}$. Recall that $\mathbb{G}$ is said to be a *group of exponent* n if $nP = \mathcal{O}$ for all elements $P \in \mathbb{G}$.

For example, the groups $\mathbb{Z}_n$ and $\mathbb{Z}_n \times \mathbb{Z}_n$ are of exponent n.

Definition 3.1 *Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be cyclic groups of large prime order q, where we denote $\mathbb{G}_1$ using additive notation and $\mathbb{G}_2$ using multiplicative notation. A map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a* (cryptographic) bilinear pairing *if it satisfies the following three properties:*

1. **Bilinearity**: For all $P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q^\times$, we have $e(aP, bQ) = e(P, Q)^{ab}$.
2. **Non-degeneracy**: For all $P \in \mathbb{G}_1^\times$, there exists some $Q \in \mathbb{G}_1^\times$ such that $e(P, Q) \neq 1$ and for all $Q \in \mathbb{G}_1^\times$, there exists some $P \in \mathbb{G}_1^\times$ such that $e(P, Q) \neq 1$.
3. **Computable**: There exists an efficient algorithm (usually taken to mean a polynomial time algorithm) for the computation of $e(P, Q)$ for all $P, Q \in \mathbb{G}_1$.

Essentially, a bilinear pairing sends a pair of elements from the group $\mathbb{G}_1$ to a single element in the group $\mathbb{G}_2$, while providing some useful properties. The most common cryptographic pairings are the *Weil pairing* (see Sections A.3.2 and A.4.3) and the *Tate pairing* (see Sections A.3.3 and A.4.4), which take $\mathbb{G}_1$ to be a subgroup of points on an elliptic curve and $\mathbb{G}_2$ to be a subgroup of the multiplicative group of a finite field. The Weil pairing and in some cases, the Tate pairing, also satisfy the *alternating* property, that $e(P, P) = 1$ for all $P \in \mathbb{G}_1$, which is not suitable for cryptographic applications. In fact, as we will see in future sections, most pairing-based protocols require that $e(P, P) \neq 1$ for all $P \in \mathbb{G}_1^\times$, a property known as *strong non-degeneracy*. In Section A.4.2, we discuss how the Weil pairing and Tate pairing can be modified to become strongly non-degenerate. For the remainder of our work, we assume that we are working with a strongly non-degenerate pairing.

Remark 3.1 *The pairing given in Definition 3.1 is known as a* symmetric pairing, *due to the fact that it maps a pair of points from the same group $\mathbb{G}_1$. Cryptographers have also considered the more general definition of a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, known as an* asymmetric pairing, *in the construction of cryptographic applications. As our goal is to discuss the classical applications, we will use the more familiar symmetric definition.*

From this point on, we assume $\mathbb{G}_1$ is a subgroup of points on an elliptic curve and $\mathbb{G}_2$ is a subgroup of the multiplicative group of a finite field. However, the results shown would be valid for any pairing that a cryptographer could find, provided that it satisfies Definition 3.1.

3.1.2 Diffie-Hellman Assumptions

In order to use the pairing for cryptographic purposes, we must discuss the various security assumptions that are used to construct pairing-based cryptosystems. Since a pairing is applied to points on an elliptic curve, we first consider two of the security problems from Section 2.3.1, the Computational Diffie-Hellman (CDH) problem and the Decisional Diffie-Hellman (DDH) problem, expressed in terms of elliptic curves.

Computational Diffie-Hellman (CDH) Assumption Given $P, aP, bP \in \mathbb{G}_1^\times$ for $a, b \in_R \mathbb{Z}_q^\times$, computing abP is hard.

Decisional Diffie-Hellman (DDH) Assumption For $P \in_R \mathbb{G}_1^\times$ and $a, b, c \in_R \mathbb{Z}_q^\times$, distinguishing between (P, aP, bP, abP) and (P, aP, bP, cP) is hard.

Clearly, the DDH problem in $\mathbb{G}_1$ is no harder than the CDH problem in $\mathbb{G}_1$, since the solution $abP = CDH(P, aP, bP)$ allows one to easily distinguish between (P, aP, bP, abP) and (P, aP, bP, cP). We also have a variation of the above problems for groups in which the CDH problem is difficult, but the DDH problem is easy, known as *Gap Diffie-Hellman* (GDH) groups.

Gap Diffie-Hellman (GDH) Assumption Given $P, aP, bP \in \mathbb{G}_1^\times$ for $a, b \in_R \mathbb{Z}_q^\times$, computing abP (i.e. solving the CDH problem $CDH(P, aP, bP)$) is hard, even with the help of a DDH oracle.

The GDH problem in $\mathbb{G}_1$ is no harder than the CDH problem in $\mathbb{G}_1$, since the former problem asks one to solve the latter. Interestingly, we can use bilinear pairings to construct GDH groups, as shown by Joux and Nguyen in [JN03]. Given a strongly non-degenerate bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, as defined in Definition 3.1, the DDH problem in $\mathbb{G}_1$ becomes easy. Given $P, aP, bP, cP \in \mathbb{G}_1^\times$, the DDH problem is

to determine whether $cP = abP$ (i.e. $c = ab \bmod q$). Using the pairing, we have

$$c = ab \bmod q \quad \text{if and only if} \quad e(P, cP) = e(aP, bP).$$

Thus a group $\mathbb{G}_1$ for which there exists a strongly non-degenerate bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ and for which the CDH problem is hard, is a GDH group.

Since pairings ensure that the DDH problem is easy in $\mathbb{G}_1$, it is not a suitable problem for the basis of pairing-based cryptosystems. The security of most pairing-based cryptosystems depend on a variant of the CDH problem, known as the Bilinear Diffie-Hellman (BDH) problem, proposed by Boneh and Franklin in [BF01] or the Decisional Bilinear Diffie-Hellman (DBDH) problem, which will be used to prove the security of our protocol in Chapter 7.

BDH Parameter Generator A BDH parameter generator $\mathcal{G}(1^\ell)$ is a probabilistic polynomial time (PPT) algorithm that, upon input of a security parameter ℓ, outputs a tuple $\langle \mathbb{G}_1, \mathbb{G}_2, e, q \rangle$ as in Definition 3.1.

Bilinear Diffie-Hellman (BDH) Assumption Given P, aP, bP, $cP \in \mathbb{G}_1^\times$ for P a generator of $\mathbb{G}_1$ and a, b, $c \in_R \mathbb{Z}_q^\times$, computing $e(P, P)^{abc}$ is hard.

Decisional Bilinear Diffie-Hellman (DBDH) Assumption Given $P \in \mathbb{G}_1$ and $a, b, c, d \in_R \mathbb{Z}_q^\times$, distinguishing between the tuples $(aP, bP, cP, e(P, P)^{abc})$ and $(aP, bP, cP, e(P, P)^d)$ is hard.

Similar to the argument above, the DBDH problem is no harder than the BDH problem. We also have that the BDH problem is no harder than the CDH problem in $\mathbb{G}_1$ or the CDH problem in $\mathbb{G}_2$. Rather informally, suppose that $CDH_{\mathbb{G}_1}$ and $CDH_{\mathbb{G}_2}$ are oracles to solve the CDH problem in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Given $abP = CDH_{\mathbb{G}_1}(P, aP, bP)$, one can easily compute $e(abP, cP) = e(P, P)^{abc}$ to solve the BDH problem. Similarly, if we let $g = e(P, P)$, then we one can easily compute $g^{ab} = e(aP, bP)$ and $g^c = e(P, cP)$ and compute $e(P, P)^{abc} = g^{abc} = CDH_{\mathbb{G}_2}(g, g^{ab}, g^c)$ to solve the BDH problem. For a detailed summary of other pairing-based security assumptions, we refer the reader to [CC05].

3.2 Joux's Tripartite Key Exchange Protocol

In Section 2.2, we discussed several solutions to allow a group of more than two participants to establish a shared session key. Recall that the majority of the protocols were based on the classic Diffie-Hellman key exchange and required at least two rounds of communication. In [Jou00], Joux proposed a simple tripartite extension of the Diffie-Hellman key exchange that allows three participants to establish a shared key in only a single round of communication.

3.2.1 Joux's Protocol

We assume that the three participants A, B and C agree in advance on the public system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, e, q, P \rangle$, where P is a generator of $\mathbb{G}_1$. Participant A chooses an ephemeral private key $a \in_R \mathbb{Z}_q^\times$ and computes and broadcasts the public key aP. Similarly, participants B and C choose ephemeral private keys $b, c \in_R \mathbb{Z}_q^\times$ and compute and broadcast the public keys bP and cP, respectively. From the bilinearity of the pairing e, we have

$$e(bP, cP)^a = e(aP, cP)^b = e(aP, bP)^c,$$

so that each participant may use their ephemeral private key to compute the shared key

$$K = e(P, P)^{abc}.$$

3.2.2 Security of Joux's Protocol

The security of Joux's protocol is directly related to the BDH problem, since a passive adversary in possession of the publicly broadcasted values aP, bP and cP cannot retrieve the shared key $e(P, P)^{abc}$ without solving the BDH problem. We note the requirement of the strong non-degeneracy property of the pairing e in Joux's protocol. Otherwise, the session key may trivially equal the identity element in $\mathbb{G}_2$. At the time, there were no known strongly non-degenerate pairings at Joux's disposal. As a solution, he adapted his protocol for use with the alternating Weil pairing by requiring each participant to broadcast a pair of points of the form (aP, aQ) for independent

P and Q. Unfortunately, this solution doubled the bandwidth requirement of the original protocol. Shortly after, Verheul [Ver01] proposed the concept of distortion maps (to be discussed in Appendix A.4), allowing cryptographers to construct modified pairings that are always strongly non-degenerate. In response to this discovery, Joux published an updated version of his paper using the single point approach in [Jou04].

Similar to the original DH key exchange protocol, Joux's protocol is vulnerable to a man-in-the-middle attack and is thus insecure against active adversaries. In [ARP02], Al-Riyami and Paterson proposed several authenticated variations of Joux's protocol which bring the long-term public/private keys of the participants into the computation of the shared session key. Unfortunately, the protocols were later shown to be vulnerable to some other attacks in [Shi03]. In [ZLK02], Zhang *et al.* proposed a more efficient authenticated variation of Joux's protocol by employing the concept of the next section, which is considered to be the killer application of bilinear pairings, *identity-based cryptography.* We discuss Zhang *et al.*'s authenticated version of Joux's protocol in more detail in Section 4.1.5.

3.3 Identity-based Cryptography

In 1984, Shamir [Sha85] proposed the idea of a public-key cryptosystem in which the user's public key could be any unique, arbitrary string, such as a name with a social insurance number or an email address, which he called *identity-based (or ID-based) public key cryptography.* In an ID-based cryptosystem, if Alice were to send an encrypted message to Bob, she would use his email address "bob@gmail.com", for example, as the public key. Bob would not have to derive his private key from his public key to decrypt the message. He would contact a *trusted third party* (TTP), known as the *private key generator* (PKG), and upon authentication the PKG would send the private key to Bob.

With traditional public-key cryptosystems, Alice would need to be certain that she has an authentic copy of Bob's current public key. If an attacker had published a falsified copy of Bob's public key, Alice may mistakenly encrypt messages for Bob using the attacker's key. To correct this problem, a TTP called the *certification*

authority (CA) issues a certificate for Bob which binds his public key to his identity. Alice can verify the certificate to be sure that she is using a valid copy of Bob's public key.

With an identity-based cryptosystem, there is no need for a CA. Since Alice uses Bob's identity to encrypt the message, she can be sure that the public key is authentic. Shamir had originally envisioned an encryption scheme that would enhance email systems by simplifying certificate management. Although Shamir had laid the foundation for an identity-based system, the mathematics required to fully satisfy such an encryption scheme had eluded him and other mathematicians for 17 years.

A practical identity-based encryption (IBE) scheme was not fully realized until 2001, when Boneh and Franklin [BF01] developed a creative solution to the problem posed by Shamir. Following Joux's constructive application of pairings in [Jou00], Boneh and Franklin constructed an IBE scheme, which we will henceforth call the BF-IBE scheme, using the Weil pairing. We note that, although the independently proposed NIKDS of Sakai *et al.* [SOK00] achieved identity-based encryption using pairings, this work was relatively unknown to the cryptographic community at the time. In addition, their scheme did not satisfy all the requirements of a practical IBE scheme.

In this section, we describe Boneh and Franklin's IBE scheme from a purely algorithmic perspective. In Section 3.3.1, we define the identity-based public key infrastructure (ID-PKI) and the encryption and decryption algorithms that comprise the IBE scheme. In Section 3.3.2, we define the formal security model for ID-based encryption that Boneh and Franklin adapted from the standard model for public key encryption. In addition, we define the most common security goal and the possible adversarial attacks. This section ultimately serves as motivation for the Section 3.4 where we show how Boneh and Franklin applied the concept of pairings to achieve identity-based encryption. Throughout these two sections, we follow the exhibition of Boneh and Franklin of [BF01] and develop the BF-IBE scheme in stages, with more detail being added to the algorithms after each step. By slowly building these algorithms, we will better understand the use of pairings in their construction. This technique will prepare the reader for Chapter 5 where we'll see how pairings can be

applied to the group key agreement protocols from the previous chapter.

3.3.1 IBE: A First Look

Boneh and Franklin defined an identity-based encryption scheme in terms of the algorithms that are used in its construction. This has become the convention for research in the field and we will follow suit and adopt the same notation.

Definition 3.2 *An* identity-based encryption scheme $\mathcal{E}$ *is characterized by four randomized algorithms:* **Setup, Extract, Encrypt** *and* **Decrypt.** *They are described as follows:*

Setup(ℓ)**:** Given a security parameter ℓ, the PKG generates the public system parameters, params and a master secret key msk, known only to the PKG. The system parameters include a description of the finite message space $\mathcal{M}$ and the finite ciphertext space $\mathcal{C}$. The PKG runs this algorithm to generate the initial system and it need only be executed once, when the cryptosystem is first created.

Extract(params, msk, ID)**:** Upon input of params, the master secret key msk and an arbitrary string $\mathsf{ID} \in \{0,1\}^*$, the PKG returns the private key d_{ID} to the user ID. The string ID will act as the user's public key and d_{ID} as its corresponding private key. The PKG runs this algorithm to generate the private key d_{ID} corresponding to a user ID (hence the name private key generator). The user ID uses the same private key d_{ID} to decrypt all messages encrypted using his corresponding public key and thus the PKG must only perform a single execution of the **Extract** algorithm for each private key extraction.

Encrypt(params, ID, m)**:** Upon input of params, the public key ID and a plaintext message $m \in \mathcal{M}$, the algorithm returns the corresponding encrypted ciphertext message $c \in \mathcal{C}$.

Decrypt(params, d_{ID}, c)**:** Upon input of params, the private key d_{ID} and a ciphertext message $c \in \mathcal{C}$, the algorithm returns the corresponding decrypted plaintext message m.

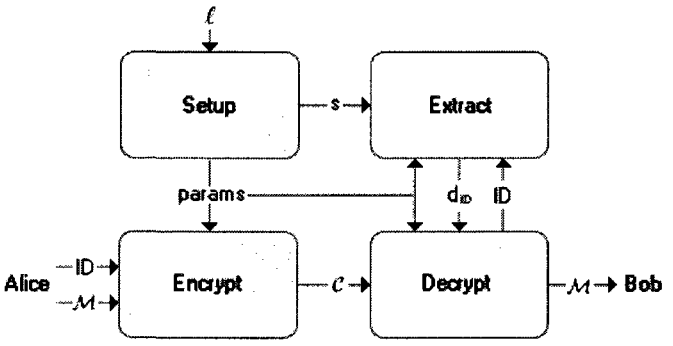

Figure 3.1: An Algorithmic View of BF-IBE Scheme

The above algorithms must satisfy the standard consistency constraint that, given an arbitrary string ID, the public parameters params generated by **Setup** and a private key d_{ID} produced by **Extract,** if c = Encrypt(params, ID, m), then m = Decrypt(params, d_{ID}, c). Figure 3.1 illustrates the simplified algorithmic model of the BF-IBE scheme.

3.3.2 Security Model for ID-based Encryption

A common security goal of encryption schemes is *indistinguishability under encryption*, proposed by Goldwasser and Micali in [GM84]. Indistinguishability (IND) refers to the property that an adversary cannot learn any information about a plaintext message given only its corresponding ciphertext. As shown in [BDPR98], the standard notion of security for public key encryption schemes is IND-CCA2 or *adaptive chosen ciphertext security.* To prove the security of their ID-based encryption scheme, Boneh and Franklin defined a new form of adaptive chosen ciphertext security for IBE schemes (IND-ID-CCA2) in [BF03]. In order to properly define these security notions, we first present the formal security model for IND-ID-CCA2 security of Boneh and Franklin, adapted from the standard model from [BDPR98, DDN00].

Adversarial Model

Similar to the security model for group key agreement (GKA) from Section 2.2.2, we prove the security of an encryption scheme by showing that an adversary has no

advantage in breaking a simulation of the scheme. However, in contrast to the model of Section 2.2.2, we define the encryption scheme simulation in terms of a security game between only two players: an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. This is due to the fact that our encryption scheme involves only two parties, a sender and a recipient. We allow the adversary to have certain capabilities, which are modelled by allowing her to issue the following queries to the challenger:

Extract(ID_i) The challenger responds by executing the **Extract** algorithm of the IBE scheme to obtain the private key d_i corresponding to the public key ID_i and returns d_i to the adversary.

Decrypt(c, ID_i) The challenger responds by first executing the **Extract** algorithm to obtain the private key d_i corresponding to the public key ID_i and then executing the **Decrypt** algorithm of the IBE scheme to decrypt the ciphertext c under the private key d_i of ID_i. The challenger returns the resulting decrypted plaintext message to the adversary.

We assume that the adversary $\mathcal{A}$ can perform any number of encryptions without issuing queries, by simply using the **Encrypt** algorithm of the IBE scheme. Using these adversarial capabilities, we can define the security game used to prove IND-ID-CCA2 security.

IND-ID-CCA2 Security Game We define the security of an identity-based encryption scheme $\mathcal{E}$ in terms of the following game between the adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

Setup: The challenger $\mathcal{C}$ runs the **Setup** algorithm to generate the system parameters params and the master secret key msk. The challenger sends params to the adversary and keeps msk hidden.

Phase 1: The adversary $\mathcal{A}$ issues Extract and Decrypt queries to the challenger, as described above. The adversary may issue these queries adaptively, so that the m^{th} query q_m may depend on queries $q_1, ..., q_{m-1}$. When $\mathcal{A}$ decides that Phase 1 is over, she produces two plaintext messages $\{m_0, m_1\}$ of equal length and an identity ID^* upon which to be tested. We require that $\mathcal{A}$ has not made an Extract query on ID^*.

Challenge: The challenger $\mathcal{C}$ chooses a random bit $b \in \{0,1\}$ and computes the challenge ciphertext $c^* = \mathsf{Encrypt}(\mathsf{params}, \mathsf{ID}^*, m_b)$. $\mathcal{C}$ returns the ciphertext c^* to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ continues to adaptively query the challenger, but may not issue the Extract query for ID^* or the Decrypt query for $\langle c^*, \mathsf{ID}^* \rangle$. The challenger responds in the same manner as in Phase 1. When $\mathcal{A}$ feels she has gathered sufficient information to make a guess at the value of the bit b, she ends Phase 2.

Response: The adversary outputs a guess $b' \in \{0,1\}$ as to the value of the bit b. The adversary wins the game if $b' = b$.

Notions of Security

Using the IND-ID-CCA2 security game, we can model adversarial attacks of various strengths by restricting the adversary's access to the oracle queries. We define the following attacks, discussed in [BDPR98, BF03], in order of increasing severity.

Chosen Plaintext Attack (CPA) Under a chosen plaintext attack (CPA), the adversary may choose an arbitrary number of plaintexts to be encrypted and obtain the corresponding ciphertexts. In the ID-based environment, we allow the adversary to only have access to the Extract query in the IND-ID-CCA2 security game. Naturally, the adversary may also encrypt messages using the **Encrypt** algorithm of the IBE scheme.

Chosen Ciphertext Attack (CCA1) Under a (non-adaptive) chosen ciphertext attack (CCA1), the adversary has the abilities of the CPA, but may also choose an arbitrary number of ciphertexts to be decrypted under some unknown private key and obtain the corresponding decrypted plaintext messages. In the ID-based environment, we allow the adversary to have access to both the Extract and Decrypt queries in Phase 1 of the IND-ID-CCA2 security game, but we only allow access to the Extract query in Phase 2. The term non-adaptive means that the adversary cannot adapt Decrypt queries based on the test ciphertext from the Challenge stage of the game. We note that the adversary may still issue Decrypt queries adaptively in Phase 1 of the game. This attack is often

referred to as a "lunchtime" or "midnight" attack, as the adversary has access to the Decrypt oracle for a short amount of time, presumably while the Challenger is out to lunch.

Adaptive Chosen Ciphertext Attack (CCA2) An adaptive chosen ciphertext attack (CCA2) is a stronger form of CCA1, under which the adversary may issue both the Extract and Decrypt queries in Phase 1 and Phase 2 of the IND-ID-CCA2 security game.

In [BDPR98], Bresson *et al.* show that *indistinguishability under adaptive chosen ciphertext attacks* (IND-CCA2) implies *indistinguishability under chosen plaintext attacks* (IND-CPA) and *indistinguishability under chosen ciphertext attacks* (IND-CCA1). Consequently, we focus on the stronger notion of adaptive chosen ciphertext security for IBE schemes (IND-ID-CCA2) and refer the reader to **[BF03]** for a discussion of IND-ID-CPA and IND-ID-CCA1.

We conclude this section by formally defining the notion of adaptive chosen ciphertext security for IBE schemes using the IND-ID-CCA2 security game.

Definition 3.3 *Let* Distinguish *be the event that the adversary can successfully distinguish between ciphertexts of the IND-ID-CCA2 security game. The advantage of an IND-ID-CCA2 adversary $\mathcal{A}$ in attacking $\mathcal{E}$ is defined as*

$$\mathrm{Adv}_{\mathcal{A},\mathcal{E}}^{\mathsf{Distinguish}}(\ell) = |2 \cdot \Pr[b' = b] - 1|$$

for the security parameter ℓ. We say that the identity-based encryption scheme $\mathcal{E}$ is semantically secure (or indistinguishable) under adaptive chosen attacks (IND-ID-CCA2) *if the adversary $\mathcal{A}$ has a negligible advantage.*

3.4 Boneh and Franklin's IBE Scheme

In this section, we present the BF-IBE scheme of Boneh and Franklin from [BF01, BF03]. We build upon the generalized IBE scheme of Section 3.3.1, emphasizing the use of bilinear pairings, to construct the `BasicIdent` version of the BF-IBE scheme in Section 3.4.1. While the `BasicIdent` scheme is secure against chosen plaintext

attacks (IND-ID-CPA), it is vulnerable to adaptive chosen ciphertext attacks (CCA2), as we will see in Section 3.4.2. In Section 3.4.3, we show how Boneh and Franklin adapted the `BasicIdent` scheme by applying a technique due to Fujisaki and Okamoto to produce the final IND-ID-CCA2 secure version of the BF-IBE scheme, `FullIdent`. We stress that the reader should pay particular attention to Boneh and Franklin's ID-PKI, comprised of the **Setup** and **Extract** algorithms of the `BasicIdent` and `FullIdent` schemes, as they are used in virtually all identity-based cryptosystems, including Smart's ID-based key agreement protocol in Section 3.5 and every ID-based protocol discussed in subsequent chapters.

3.4.1 The BasicIdent Scheme

The `BasicIdent` scheme is characterized by the following randomized algorithms:

Setup: Given a security parameter $\ell \in \mathbb{Z}^+$, the PKG performs the following:

1. Generate the BDH parameters $\langle \mathbb{G}_1, \mathbb{G}_2, e, q \rangle$ from the BDH generator $\mathcal{G}(1^\ell)$.
2. Find a random generator $P \in \mathbb{G}_1$.
3. Choose $s \in_R \mathbb{Z}_q^\times$ and set $P_{pub} = sP$. The value s will serve as the master secret key of the PKG, while P_{pub} serves as the global public key.
4. Choose hash functions $H_0 : \{0,1\}^* \rightarrow \mathbb{G}_1^\times$ and $H_1 : \mathbb{G}_2 \rightarrow \{0,1\}^*$.
5. The PKG returns the message space $\mathcal{M} = \{0,1\}^*$, the ciphertext space $\mathcal{C} = \mathbb{G}_1^\times \times \{0,1\}^*$ and the public system parameters

$$\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_{pub}, H_0, H_1 \rangle$$

and keeps the master secret key s hidden.

Extract: Given an identity string $\mathsf{ID} \in \{0,1\}^*$, the PKG performs the following:

1. Compute

i) the public key $Q_{\mathsf{ID}} = H_0(\mathsf{ID}) \in \mathbb{G}_1^\times$ using the hash function H_0 [1] ,

[1] In [BF03], Boneh and Franklin show that it suffices to have a hash function $H : \{0,1\}^* \rightarrow A$ and an admissible encoding function $L : A \rightarrow \mathbb{G}^*$. They describe an admissible encoding function MapToPoint that may be used in practical implementations.

ii) the private key $d_{\mathsf{ID}} = sQ_{\mathsf{ID}}$ using the master secret key s.

2. Returns the public/private key pair $(Q_{\mathsf{ID}}, d_{\mathsf{ID}})$ corresponding to the identity ID.

Encrypt: Given a plaintext message $m \in \mathcal{M}$ and a recipient identity ID, the algorithm performs the following:

1. Choose an ephemeral value $r \in_R \mathbb{Z}_q^\times$.
2. Compute
   i) $Q_{\mathsf{ID}} = H_0(\mathsf{ID}) \in \mathbb{G}_1^\times$ using the hash function H_0,
   ii) $g_{\mathsf{ID}} = e\left(Q_{\mathsf{ID}}, P_{pub}\right)$ using the pairing e and the global public key P_{pub},
   iii) $h_{\mathsf{ID}} = H_1(g_{\mathsf{ID}}^r)$ using the hash function H_1 and the ephemeral value r.
3. Return the ciphertext $c = \langle rP, m \oplus h_{\mathsf{ID}} \rangle$.

Decrypt: Given a ciphertext message $c = \langle U, V \rangle \in \mathcal{C}$ and a private key d_{ID}, the algorithm performs the following:

1. Compute
   i) $g_{\mathsf{ID}}^r = e(d_{\mathsf{ID}}, U)$ using the pairing e,
   ii) $h_{\mathsf{ID}} = H_1(g_{\mathsf{ID}}^r)$ using the hash function H_1.
2. Return the plaintext message computed as $m = V \oplus h_{\mathsf{ID}}$.

We prove the consistency of the `BasicIdent` scheme from the bilinearity of e as follows:

$$e(d_{\mathsf{ID}}, U) = e(sQ_{\mathsf{ID}}, rP) = e(Q_{\mathsf{ID}}, P)^{sr} = e(Q_{\mathsf{ID}}, sP)^r = e\left(Q_{\mathsf{ID}}, P_{pub}\right)^r = g_{\mathsf{ID}}^r.$$

Remark 3.2 *In the above* `BasicIdent` *scheme, we note that the sender explicitly computes the public key of the recipient using the hash function H_0 and the recipient's identity. When we discuss identity-based cryptosystems in future sections, we simply assume that the sender is already in possession of the long-term public key Q_{ID} of the recipient. In addition, we note that the PKG must find a generator P for the group $\mathbb{G}_1$, which can be a difficult task in itself. However, since we take $\mathbb{G}_1$ to be a cyclic*

group of prime order, each element of $\mathbb{G}_1$ *is a generator, allowing the PKG to simply choose an element* $P \in_R \mathbb{G}_1$ *for the generator.*

3.4.2 Chosen Ciphertext Attack on BasicIdent

In [BF03, Theorem 4.1], Boneh and Franklin show that the `BasicIdent` scheme is IND-ID-CPA or chosen plaintext secure. They note that the `BasicIdent` scheme is vulnerable to adaptive chosen ciphertext attacks and thus is not IND-ID-CCA2 secure, although they do not describe the successful attack. The vulnerability results from the fact that the value $h_{\mathsf{ID}} = H_1(e\,(Q_{\mathsf{ID}}, P)^{rs})$, that is ultimately used to encrypt and decrypt the plaintext message m, does not depend on the value of m. We show that the adversary in the IND-ID-CCA2 security game of Section 3.3.2 has a non-negligible advantage in distinguishing between the ciphertexts.

Suppose that the IND-ID-CCA2 adversary $\mathcal{A}$ submits an identity ID and plaintexts $\{m_0, m_1\}$ upon which to be tested at the end of Phase 1. The challenger $\mathcal{C}$ returns the ciphertext

$$c = \mathsf{Encrypt}(\mathsf{params}, \mathsf{ID}, m_b) = \langle rP, m_b \oplus h_{\mathsf{ID}} \rangle = \langle U, V \rangle$$

where $b \in_R \{0,1\}$, $r \in \mathbb{Z}_q^{\times}$ and $h_{\mathsf{ID}} = H_1(e(Q_{\mathsf{ID}}, P)^{rs})$ in the Challenge stage.

In Phase 2, $\mathcal{A}$ may issue adaptive $\mathsf{Extract}$ and $\mathsf{Decrypt}$ queries, as long as she does not issue the $\mathsf{Extract}$ query for ID or the $\mathsf{Decrypt}$ query for $\langle c, \mathsf{ID} \rangle$. The adversary chooses some $N \in \{0,1\}^*$ and issues the query $\mathsf{Decrypt}(\mathsf{params}, c', \mathsf{ID})$ where $c' = \langle U, N \oplus V \rangle$. Using the **Decrypt** algorithm from `BasicIdent`, the challenger computes

$$h_{\mathsf{ID}} = H_1(e(d_{\mathsf{ID}}, U))$$

and returns the decrypted plaintext

$$\begin{aligned} m' &= (N \oplus V) \oplus h_{\mathsf{ID}} \\ &= N \oplus (m_b \oplus h_{\mathsf{ID}}) \oplus h_{\mathsf{ID}} \\ &= N \oplus m_b. \end{aligned}$$

In the Response stage of the game, $\mathcal{A}$ recovers $m_b = N \oplus m'$ and can thus make a successful guess at the value of the bit b. The adversary will have a non-negligible advantage in the IND-ID-CCA2 security game and thus the `BasicIdent` scheme is not adaptive chosen ciphertext secure.

3.4.3 The FullIdent Scheme

In the previous section, we showed that, while the `BasicIdent` scheme is IND-ID-CPA secure, it is vulnerable to adaptive chosen ciphertext attacks. Recall that the vulnerability was ultimately due to the fact that the encryption value h_{ID} was independent from the plaintext message m. To provide IND-ID-CCA2 security, the general idea is to compute the ephemeral value r of the `BasicIdent` scheme as a function of the plaintext message m and a random string σ, rather than randomly generating the value.

To transform the `BasicIdent` scheme to an IND-ID-CCA2 secure scheme, Boneh and Franklin use a hybridization technique of Fujisaki and Okamoto from [FO99]. The Fujisaki-Okamoto technique can be used to turn an IND-CPA secure protocol into an IND-CCA2 secure scheme. This is formally proven for identity-based encryption schemes as well in [YKH+06]. Applying this technique to `BasicIdent`, we obtain the final IND-ID-CCA2 secure version of the BF-IBE scheme, the `FullIdent` scheme.

The `FullIdent` scheme is characterized by the following randomized algorithms:

Setup: The PKG performs operations as described in `BasicIdent`. We also choose additional hash functions

$$H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q^\times \qquad\qquad H_3 : \{0,1\}^* \rightarrow \{0,1\}^*.$$

The PKG publishes the message space $\mathcal{M}$, the ciphertext space $\mathcal{C}$ and the public system parameters

$$\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_{pub}, H_0, H_1, H_2, H_3 \rangle .$$

Extract: The PKG performs the algorithm as outlined in `BasicIdent`.

Encrypt: Given a plaintext message $m \in \mathcal{M}$ and a recipient identity ID, the algorithm performs the following:

1. Choose an ephemeral string $\sigma \in_R \{0,1\}^*$.
2. Compute
   i) $Q_{\mathsf{ID}} = H_0(\mathsf{ID}) \in \mathbb{G}_1^\times$ using the hash function H_0,
   ii) $g_{\mathsf{ID}} = e\left(Q_{\mathsf{ID}}, P_{pub}\right)$ using the pairing e and the global public key P_{pub},
   iii) $r = H_2(\sigma \parallel m) \in \mathbb{Z}_q^\times$ using the hash function H_2,
   iii) $h_{\mathsf{ID}} = H_1(g_{\mathsf{ID}}^r)$ using the hash function H_1.
3. Return the ciphertext $c = \langle rP, \sigma \oplus h_{\mathsf{ID}}, m \oplus H_3(\sigma) \rangle$.

Decrypt: Given a ciphertext message $c = \langle U, V, W \rangle \in \mathcal{C}$ and a private key d_{ID}, the algorithm rejects the ciphertext c if $U \notin \mathbb{G}_1^\times$. Otherwise, the algorithm performs the following:

1. Compute
   i) $g_{\mathsf{ID}}^r = e(d_{\mathsf{ID}}, U)$ using the pairing e,
   ii) $h_{\mathsf{ID}} = H_1(g_{\mathsf{ID}}^r)$ using the hash function H_1,
   iii) $\sigma = V \oplus h_{\mathsf{ID}}$.
2. Set the plaintext message as $m = W \oplus H_3(\sigma)$.
3. Set $r = H_2(\sigma \parallel m)$ and check that $U = rP$. If not, reject the ciphertext.

In addition to providing security against chosen ciphertext attacks, the `FullIdent` scheme allows us to check that the ciphertext has been transmitted correctly. In [BF03, Theorem 4.4], Boneh and Franklin prove that `FullIdent` is an adaptive chosen ciphertext secure IBE scheme, provided that the BDH problem is hard in $\mathbb{G}_1$ and $\mathbb{G}_2$.

3.5 Smart's ID-Based Authenticated Key Agreement Protocol

In Section 2.3, we briefly discussed the MQV protocol from [LMQ+98, LMQ+03], which allows two parties to perform authenticated key agreement without increasing

the communication costs or bandwidth of the DH key exchange. In [Sma01], Smart proposed the first identity-based key agreement protocol which allows two parties to perform authenticated key agreement, while each only broadcasting a single message. As mentioned in [Sma01], the message flows in Smart's protocol are identical to the elliptic curve Diffie-Hellman (ECDH) key exchange or the elliptic curve MQV (ECMQV) protocol and thus from an outside observer, Smart's protocol is indistinguishable from these schemes. However, the unique key computation stage combines the ideas of pairing-based key exchange from Joux and the identity-based encryption from Boneh and Franklin.

3.5.1 Smart's Protocol

Suppose Alice and Bob want to establish a shared session key. Using the **Setup** and **Extract** algorithms of the BF-IBE scheme from Section 3.4, the PKG publishes the public system parameters $\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_{pub} \rangle$ and distributes the identity-based public/private key pairs (Q_A, d_A) and (Q_B, d_B) to Alice and Bob, respectively. Smart's ID-based authenticated key agreement runs in the following round of communication:

Round 1 Alice chooses an ephemeral private key $k_A \in_R \mathbb{Z}_q^\times$ and computes the ephemeral public key $T_A = k_A P$. Similarly, Bob chooses $k_B \in_R \mathbb{Z}_q^\times$ and computes $T_B = k_B P$. Alice and Bob broadcast their respective ephemeral public keys, T_A and T_B, to one another.

Key Computation Through the bilinearity of the pairing e, Alice and Bob can each compute the shared session key

$$\begin{aligned} K &= e(k_A Q_B, P_{pub}) \cdot e(d_A, T_B) \\ &= e(k_A Q_B, sP) \cdot e(sQ_A, k_B P) \\ &= e(sQ_B, k_A P) \cdot e(k_B Q_A, sP) \\ &= e(d_B, T_A) \cdot e(k_B Q_A, P_{pub}). \end{aligned}$$

The beauty of Smart's protocol lies in the fact that

$$e(k_A Q_B, P_{pub}) = e(d_B, T_A).$$

The bilinearity of the pairing allows Alice to combine her ephemeral private key k_A and Bob's long-term public key Q_A and Bob to combine his long-term private key d_B and Alice's ephemeral public key T_A in such a way that they arrive at the same value. By performing this pairing computation for each member, the final key computation requires Alice to use both her ephemeral and long-term private keys, along with Bob's ephemeral and long-term public keys.

3.5.2 Security of Smart's Protocol

Similar to the MQV protocol, Smart's protocol only provides partial forward secrecy, as an adversary in possession of the long-term private keys of both Alice and Bob can easily compute the session key as $K = e(d_A, T_B) \cdot e(d_B, T_A)$. In [CK02], Chen and Kudla proposed an efficient modified version of Smart's protocol, called the SCK protocol, which requires Alice and Bob to compute the respective values $W_A = k_A Q_A$ and $W_B = k_B Q_B$ using their ephemeral private keys k_A and k_B, and broadcast these values in place of T_A and T_B in Smart's original protocol. Consequently, the session key can be computed with a single pairing operation as

$$K = e(d_A, W_B + k_A Q_B) = e(W_A + k_B Q_A, d_B) = e(Q_A, Q_B)^{s(k_A + k_B)}.$$

While the SCK protocol still only provides partial forward secrecy, the authors suggest that both protocols can be further adapted to provide perfect forward secrecy by bringing the ECDH key exchange into the key computation stage using some hash function H. Indeed, for Smart's original protocol we see that, by adding the ECDH key

$$k_A T_B = k_B T_A = k_A k_B P$$

to the group session key, as

$$TEK = H(k_A k_B P, K),$$

where H is some hash function $H : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \{0,1\}^*$, we obtain perfect forward secrecy, since an adversary must be in possession of the long-term and ephemeral private keys of a member in order to compute the session key. Additionally, the protocol provides TA forward secrecy, since the master secret key of the PKG may be disclosed without compromising the session key. Since the ephemeral contributions of Alice and Bob in the SCK protocol (i.e. $W_A = k_A Q_A$ and $W_B = k_B Q_B$) involve the long-term public keys and do not mimic the ECDH key contributions, Alice and Bob must also broadcast T_A and T_B in order to provide these security properties.

The security of Smart's protocol and the modified SCK protocol, along with a number of other ID-based key agreement protocols, were proven by Chen *et al.* in [CCS07]. This paper also provides a comprehensive survey of the ID-based key exchange protocols available, comparing the computational efficiency and security properties of each protocol, while also considering the provision of asymmetric pairings.

3.6 Summary

In this chapter, we introduced the bilinear pairing as a cryptographic building block. We gave a brief history of how the pairing became reknowned in the cryptographic community, citing early applications of pairings and providing examples of classical pairing-based protocols. In particular, we discussed Joux's protocol, which shows how the bilinear pairing can be used to produce a natural tripartite extension of the DH key exchange, and the so-called killer application of bilinear pairings, identity-based cryptography. We provided a thorough examination of the identity-based encryption scheme of Boneh and Franklin and presented an authenticated identity-based alternative to the key exchange schemes of Diffie-Hellman and Joux, due to Smart.

As pairings and identity-based cryptography play a fundamental role in our work, we reference these concepts in each subsequent chapter of the thesis. Specifically, we discuss identity-based signature and signcryption schemes in Chapter 4, identity-based group key agreement protocols in Chapter 5 and propose our own forward-secure identity-based authenticated group key agreement protocol in Chapter 7. In

Appendix A, we define two particular examples of bilinear pairings and discuss the mathematical principles that enable the nice cryptographic properties of these pairings.

Chapter 4

ID-based Signature and Signcryption Schemes

In their landmark paper [DH76], Diffie and Hellman first described the notion of a *digital signature scheme*, which they described as a digital phenomenon with the same properties as a handwritten signature. The idea suggested by Diffie and Hellman is that, by applying the concept of public key cryptography, we can provide the same level of authentication for an electronic message, as guaranteed by a handwritten signature on a letter. Digital signature schemes, or signature schemes for short, are now regarded as a cryptographic primitive, used in the construction of cryptographic protocols, along with encryption schemes. The first digital signature scheme (and one of the first public key encryption schemes) was proposed by Rivest, Shamir and Adleman in [RSA78]. The signature scheme, known as the RSA signature scheme, is based on the difficulty of factoring integers. In [Fia89], Fiat proposed the concept of batch cryptography in the context of the RSA signature scheme, where a signer could simultaneously generate a batch of independent signatures for different messages. This concept was later adapted to allow a user to verify a batch of signatures together, known as *batch verification.*

Shamir proposed the first identity-based signature (IBS) scheme in [Sha85], also based on the problem of integer factorization. However, this IBS scheme was not suitable for practical implementations. Shortly after Boneh and Franklin proposed the first practical and secure identity-based encryption (IBE) scheme (see Section

3.4), a number of efficient IBS schemes making use of the identity-based public key infrastructure (ID-PKI) of Boneh and Franklin were proposed, in particular the schemes of Paterson [Pat02], Hess [Hes02, Hes03] and Cha and Cheon [CC03]. Soon after, the concept of batch verification was applied to identity-based signature schemes in [ZK03, CKY04, YCK04, ZSNS04].

Prior to the work of Zheng in [Zhe97], confidentiality and authentication were solely provided by encryption and signature schemes, respectively. Zheng proposed a new cryptographic primitive known as *signcryption*, that provides both confidentiality and authentication at a cost that's less than applying encryption and signature schemes separately. The first identity-based signcryption scheme, again making use of the pairing-based ID-PKI of Boneh and Franklin, was proposed in [ML02]. More efficient signcryption schemes, providing a variety of different properties, were proposed in [Boy03, LQ03, NR03, CYHC04, LQ04b, BLMQ05, CML05].

In this chapter, we provide a thorough overview of the signature and signcryption schemes in an identity-based public key infrastructure. Section 4.1 is devoted to the topic of identity-based signatures. We formally define the concept and provide a few examples of schemes from the literature. In Section 4.2 we discuss the concept of batch verification, providing a formal definition and illustrating how schemes may be modified to provide batch verification. In Section 4.3, we discuss the concept of identity-based signcryption. Following the format of Section 4.1, we formally define the concept and provide examples of signcryption schemes from the literature. Finally, we give a summary of the results and look toward future chapters in Section 4.4.

4.1 Identity-based Signature Schemes

We provide a formal definition for an identity-based signature scheme and discuss the necessary properties in Section 4.1.1 and discuss the formal model used to analyze the security of identity-based signature schemes in Section 4.1.2. To illustrate the concept of identity-based signatures, we present the IBS scheme of Cha and Cheon in 4.1.3 and the scheme of Hess in 4.1.4. Finally, we present the IBS scheme of Zhang *et al.* in Section 4.1.5 and show they applied the scheme to provide authentication in

the tripartite key exchange protocol of Joux from Section 3.2.

4.1.1 Preliminary Definitions

As mentioned in [Pat02], it is both efficient and convenient to have an ID-based signature scheme which is based on the same underlying computational primitives and makes use of the same ID-PKI as an IBE scheme. Consequently, we define an identity-based signature scheme in the same manner as the IBE scheme of Section 3.3.1, where a private key generator (PKG) is used to establish the ID-PKI, consisting of the algorithms **Setup** and **Extract**.

Definition 4.1 *An* identity-based signature scheme $\mathcal{S}$ *is characterized by three randomized algorithms* **Setup**, **Extract** *and* **Sign**, *and a deterministic algorithm* **Verify**. *They are described as follows:*

Setup(ℓ): Given a security parameter ℓ, the PKG generates the pair $\langle \mathsf{msk}, \mathsf{params} \rangle$ at random, where msk is the master secret key, known only to the PKG and params refer to the public system parameters. These parameters include a global public key P_{pub} and a description of the finite message space $\mathcal{M}$ and the finite signature space Σ.

Extract($\mathsf{params}, \mathsf{msk}, \mathsf{ID}$): Upon input of params, the master secret key msk and an identity string $\mathsf{ID} \in \{0,1\}^*$, the PKG returns the private signing key d_{ID} corresponding to the user ID.

Sign($\mathsf{params}, d_{\mathsf{ID}}, m$) Upon input of params, the signer's private signing key d_{ID} and a plaintext message $m \in \mathcal{M}$, the algorithm returns the signature σ.

Verify($\mathsf{params}, \mathsf{ID}, m, \sigma$) Upon input of params, the public key ID and a signature σ, the algorithm returns $\top$ if σ is a valid signature by the sender ID on the message m and $\perp$, otherwise.

The above algorithms must satisfy the standard consistency constraint that, given an arbitrary string ID, the public parameters params generated by **Setup** and a private signing key d_{ID} produced by **Extract**, if $\sigma = \mathsf{Sign}(\mathsf{params}, d_{\mathsf{ID}}, m)$, then $\top = \mathsf{Verify}(\mathsf{params}, \mathsf{ID}, m, \sigma)$.

Properties of Signature Schemes

The primary goals of a signature scheme is to provide the properties of authentication and non-repudiation.

- **Authentication**: *Authentication* is the process whereby a recipient B of a signed message from the sender A is assured that the message was indeed sent by A. In other words, only an entity in possession of the signing key for an identity ID may create a valid signature that speaks in the name of ID.
- **Non-repudiation**: A signature scheme provides *non-repudiation* if the signer A of a message cannot later deny having sent the message, while also claiming that their signing key remains secret. In other words, the recipient B of a message can prove to a third party C that the message was indeed signed by the sender A.

In an identity-based signature scheme, as defined in Definition 4.1, the PKG has knowledge of the private signing key of all users and can thus generate a valid signature for any user. While the PKG is believed to be a trusted third party, ID-based signature schemes are sometimes criticized for not providing proper non-repudiation. A common solution to this problem is to partition the master secret key s amongst multiple PKGs using the concept of threshold cryptography (proposed by Shamir in [Sha79]), as mentioned in [CC03]. For our purposes, we assume that the PKG is completely trusted, but refer the interested reader to [CC03, BZ04] for more information.

4.1.2 Security Model for ID-based Signature Schemes

The standard notion of security for signature schemes is *existential unforgeability against adaptive chosen message attacks* (EUF-CMA), proposed by Goldwasser *et al.* in [GMR88]. The goal of the adversary is to produce a valid message-signature pair, or an *existential forgery*, for the signature scheme. In order to prove the security of their IBS scheme, Cha and Cheon defined a new form of security, known as *existential unforgeability against adaptive chosen message and ID attacks.* Following the approach of Section 3.3.2, we first present the formal model for analyzing the security of IBS schemes and then discuss the various attacks of the adversary.

Adversarial Model

Similar to the models of Sections 2.2.2 and 3.3.2, we prove the security of a signature scheme by showing that, in the course of breaking a simulation of the signature scheme, the adversary must solve some hard mathematical problem. We define the simulation in terms of security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, where the adversary's capabilities are modelled by allowing her to issue the following queries to the challenger:

Extract(ID_i) The challenger responds by executing the **Extract** algorithm of the IBS scheme to obtain the private key d_i corresponding to the public key ID_i and returns d_i to the adversary.

Sign(m, ID_i) The challenger responds by first executing the **Extract** algorithm to obtain the private key d_i corresponding to the public key ID_i, and then executing the **Sign** algorithm of the IBS scheme to produce a signature σ for the user ID_i, on the message m. The challenger returns the resulting signature to the adversary.

We assume that the adversary $\mathcal{A}$ has access to the **Verify** algorithm of the IBS scheme and can verify the authenticity of any message-signature pair without the use of an oracle. Using these adversarial capabilities, we define the security game for ID-based signature schemes as follows:

EUF-CMA Security Game The security of an identity-based signature scheme $\mathcal{S}$ is defined in terms of the following game between the adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

Setup: The challenger $\mathcal{C}$ runs the **Setup** algorithm to generate the system parameters params and the master secret key msk. The challenger sends params to the adversary and keeps msk hidden.

Phase 1: The adversary $\mathcal{A}$ issues Extract and Sign queries to the challenger, as described above, and may issue these queries adaptively, so that the m^{th} query q_m may depend on queries $q_1, ..., q_{m-1}$.

Forgery: $\mathcal{A}$ returns a triple $(\mathsf{ID}^*, m, \sigma)$, where $\mathsf{ID}^* \in \{0,1\}^*$ is an identity and $m \in \mathcal{M}$ is a message and $\sigma \in \Sigma$ is a signature.

Result: The adversary wins the game if σ is a valid signature by the user ID^* on the message m and if $\mathcal{A}$ made no $\mathsf{Extract}(\mathsf{ID}^*)$ query and no query of the form $\mathsf{Sign}(m, \mathsf{ID}^*) = \sigma'$.

Types of Attacks

As in the security game of Section 3.3.2, we can model attacks by adversaries of varying strengths by restricting the adversary's access to oracle queries in the EUF-CMA security game. We define the following attacks, as discussed in [GMR88], in order of increasing severity.

No Message Attack (NMA) Under a no message attack (NMA), the adversary is only given access to the public verification key (i.e. the identity) of the signers and has no access to the Extract or Sign oracles.

Known Message Attack (KMA) Under a known message attack (KMA), the adversary is given access to a set of message-signature pairs $\{(m_1, \sigma_1), ..., (m_n, \sigma_n)\}$, where the messages and corresponding signatures are not chosen by her. As with NMA, the adversary has no access to the Extract or Sign oracles.

Chosen Message Attack (CMA) Under a (non-adaptive) chosen message attack (CMA), the adversary is given access to the Extract and Sign queries and thus may sign any message of her choice. As with the IBE attacks of Section 3.3.2, the strongest variation of this attack allows the adversary to query the Sign oracle adaptively, known as an *adaptive chosen message attack.*

Stronger definitions in the security analysis of signcryption schemes are given in [GMR88], including the notion of a "total break", where the adversary obtains the private signing key of the signer, and a "universal forgery", where the adversary can construct an efficient signing algorithm that can produce signatures from the IBS scheme with a high probability of success. However, our main concern is the notion of existential unforgeability. We conclude this section with a formal definition.

Definition 4.2 *Let* Forge *be the event that the adversary successfully produces a valid message-signature forgery in the EUF-CMA security game. The advantage of an EUF-CMA adversary $\mathcal{A}$ in attacking $\mathcal{S}$ is defined as*

$$\mathrm{Adv}_{\mathcal{A},\mathcal{S}}^{\mathsf{Forge}}(\ell) = \Pr[\mathsf{Forge}]$$

for the security parameter ℓ. We say that the identity-based signature scheme $\mathcal{S}$ is existentially unforgeable against adaptive chosen-message attacks (EUF-CMA) *if the adversary $\mathcal{A}$ has a negligible advantage.*

4.1.3 The Cha-Cheon ID-based Signature Scheme

In [CC03], Cha and Cheon proposed one of the first provably-secure ID-based signature schemes, which we call the ChCh-IBS scheme. The original ChCh-IBS scheme was presented in the context of Gap Diffie-Hellman (GDH) groups (see Section 3.1). The authors show that, using GDH groups obtained from bilinear pairings, we obtain a special case of their IBS scheme that uses the familiar ID-PKI of Boneh and Franklin from Section 3.4. However, as noted by Paterson in [BSe05, §X.4.1], *all* realizations of these GDH groups currently use pairings on elliptic curves. Furthermore, the pairing-based case of the ChCh-IBS scheme allows for batch verification of signatures, as we will see in Section 4.2, while the GDH-based scheme does not. Consequently, we present the ChCh-IBS scheme in the context of pairings, as given in [BSe05, §X.4.1].

The ChCh-IBS scheme is characterized by the following algorithms:

Setup: Given a security parameter $\ell \in \mathbb{Z}^+$, the algorithm outputs the public system parameters

$$\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_{pub}, H_0, H_1 \rangle ,$$

where $\langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_{pub} \rangle$ are as defined in Section 3.4 and H_0 and H_1 are the hash functions $H_0 : \{0,1\}^* \rightarrow \mathbb{G}_1^\times$ and $H_1 : \mathbb{G}_1 \times \{0,1\}^* \rightarrow \mathbb{Z}_q^\times$. The PKG publishes the message space $\mathcal{M} = \{0,1\}^*$, the signature space $\Sigma = \mathbb{G}_1^\times \times \mathbb{G}_1^\times$ and params and keeps the master secret key $s \in \mathbb{Z}_q^\times$ hidden.

Extract: Given an identity $\mathsf{ID} \in \{0,1\}^*$, the PKG returns the public key $Q_{\mathsf{ID}} = H_0(\mathsf{ID})$ and the private key $d_{\mathsf{ID}} = sQ_{\mathsf{ID}}$.

Sign: To sign a message $m \in \mathcal{M}$, the sender with identity ID and public/private key pair $(Q_{\mathsf{ID}}, d_{\mathsf{ID}})$:

1. Chooses $r \in_R \mathbb{Z}_q^\times$.
2. Computes
   i) $X = rQ_{\mathsf{ID}}$,
   ii) $h = H_1(X, m)$,
   iii) $Z = (r + h)d_{\mathsf{ID}}$.
3. Returns the signature $\sigma = (X, Z)$.

Verify: To verify that the signature $\sigma = (X, Z)$ on the message m belongs to ID, the recipient:

1. Computes $h = H_1(X, m)$.
2. Checks that
$$e(Z, P) = e(X + hQ_{\mathsf{ID}}, P_{pub}).$$

The consistency of the ChCh-IBS scheme is easily proven from the bilinearity of e, as follows:

$$e(X + hQ_{\mathsf{ID}}, P_{pub}) = e(rQ_{\mathsf{ID}} + hQ_{\mathsf{ID}}, sP) = e((r + h)d_{\mathsf{ID}}, P) = e(Z, P).$$

In [CC03], Cha and Cheon prove that the ChCh-IBS scheme is secure against existential forgery on adaptively chosen message and ID attacks in the random oracle model, assuming the hardness of the CDH problem from Section 3.1. We refer the reader to their work for the security reduction.

4.1.4 Hess' ID-based Signature Scheme

Hess proposed a number of ID-based signature schemes in [Hes02, Hes03]. In particular, he proposed an efficient provably secure scheme in [Hes02, Scheme 3] and [Hes03, Scheme 1], which we call the H-IBS scheme scheme, using the ID-PKI of Boneh and Franklin from Section 3.4. For reasons to be discussed in Chapter 7, we present a

slightly modified version of Hess' signature scheme that supports batch verification, as given in [FGHP08].

The H-IBS scheme is characterized by the following algorithms:

Setup: Given a security parameter $\ell \in \mathbb{Z}^+$, the algorithm outputs the public system parameters

$$\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_{pub}, H_0, H_1 \rangle ,$$

where $\langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_{pub} \rangle$ are as defined in Section 3.4 and H_0 and H_1 are the hash functions $H_0 : \{0,1\}^* \rightarrow \mathbb{G}_1^\times$ and $H_1 : \mathbb{G}_2 \times \{0,1\}^* \rightarrow \mathbb{Z}_q^\times$. The PKG publishes the message space $\mathcal{M} = \{0,1\}^*$, the signature space $\Sigma = \mathbb{G}_2^\times \times \mathbb{G}_1^\times$ and params and keeps the master secret key $s \in \mathbb{Z}_q^\times$ hidden.

Extract: As in the ChCh-IBS scheme of Section 4.1.3.

Sign: To sign a message $m \in \mathcal{M}$, the sender with identity ID and public/private key pair $(Q_{\mathsf{ID}}, d_{\mathsf{ID}})$:

1. Chooses $r \in_R \mathbb{Z}_q^\times$ and $P_1 \in_R \mathbb{G}_1^\times$.
2. Computes
   i) $R = e(P_1, P)^r$,
   ii) $h = H_1(R, m)$,
   iii) $Z = rP_1 + hd_{\mathsf{ID}}$.
3. Returns the signature $\sigma = (R, Z)$.

Verify: To verify that the signature $\sigma = (R, Z)$ on the message m belongs to ID, the recipient:

1. Computes $h = H_1(R, m)$.
2. Checks that
$$e(Z, P) = R \cdot e(hQ_{\mathsf{ID}}, P_{pub}).$$

The consistency of the H-IBS is easily proven from the bilinearity of e, as follows:

$$R \cdot e(hQ_{\mathsf{ID}}, P_{pub}) = e(rP_1, P) \cdot e(hd_{\mathsf{ID}}, P) = e(rP_1 + hd_{\mathsf{ID}}, P) = e(Z, P).$$

Similar to the ChCh-IBS scheme from Section 4.1.3, the H-IBS scheme is proven secure against existential forgery on adaptively chosen message and ID attacks in the random oracle model, assuming the hardness of the CDH problem, as shown in [Hes03, Thm 1].

4.1.5 Zhang *et al.*'s Authenticated Tripartite Key Exchange Protocol

In [ZLK02], Zhang *et al.* proposed a variation of the ChCh-IBS scheme, which we call the ZLK-IBS scheme. The authors apply the ZLK-IBS scheme to Joux's tripartite key exchange protocol from Section 3.2, to produce an authenticated tripartite key exchange protocol, which we call the ZLK protocol. In addition, they perform some optimizations to reduce the communication costs of the protocol. We first present the ZLK-IBS scheme and then show how it was used in the ZLK protocol.

Zhang *et al.*'s ID-based Signature Scheme

The ZLK-IBS scheme is characterized by the following algorithms:

Setup: As in the ChCh-IBS scheme of Section 4.1.3, where $H_0 : \{0,1\}^* \rightarrow \mathbb{G}_1^\times$ and $H_1 : \mathbb{G}_1 \times \{0,1\}^* \rightarrow \mathbb{Z}_q^\times$.

Extract: As in the ChCh-IBS scheme of Section 4.1.3.

Sign: To sign a message $m \in \{0,1\}^*$, the sender with identity ID and public/private key pair $(Q_{\mathsf{ID}}, d_{\mathsf{ID}})$:

1. Chooses $a \in_R \mathbb{Z}_q^\times$.
2. Computes
   i) $P_A = aP$,
   ii) $h = H_1(P_A, m)$,
   iii) $\mu = H_0(m)$,
   iv) $T_A = hd_{\mathsf{ID}} + a\mu$.
3. Returns the signature $\sigma = (P_A, T_A)$.

Verify: To verify that the signature $\sigma = (P_A, T_A)$ on the message m belongs to ID, the recipient:

1. Computes $h = H_1(P_A, m)$ and $\mu = H_0(m)$.
2. Checks that
$$e(T_A, P) = e(hQ_{\mathsf{ID}}, P_{pub}) \cdot e(\mu, P_A).$$

The consistency of the ZLK-IBS is easily proven from the bilinearity of e, as follows:

$$e(hQ_{\mathsf{ID}}, P_{pub}) \cdot e(\mu, P_A) = e(hd_{\mathsf{ID}}, P) \cdot e(a\mu, P) = e(hd_{\mathsf{ID}} + a\mu, P) = e(T_A, P).$$

Zhang *et al.*'s Authenticated Tripartite Key Exchange Protocol

A common approach to providing authentication in key exchange protocols is to set the public key contribution as the message to be signed. In order to achieve this, we redefine the hash function H_1 as $H_1 : \mathbb{G}_1 \rightarrow \mathbb{Z}_q^\times$.

Recall from Joux's protocol in Section 3.2, that participant A chooses $a \in_R \mathbb{Z}_q^\times$ and computes the public key contribution $P_A = aP$. To reduce communication costs in the ZLK protocol, Zhang *et al.* use the value P_A as both part of the signature σ and as the public key contribution. This idea will be used to increase the efficiency of our protocols in Chapters 6 and 7.

Suppose that three participants A, B and C, with respective ID-based public/private key pairs (Q_A, d_A),(Q_B, d_B) and (Q_C, d_C), wish to agree upon a session key. The ZLK protocol runs in a single round, as follows:

Round 1 Participant A chooses an ephemeral private value $a \in_R \mathbb{Z}_q^\times$ and broadcasts $P_A = aP$ and $T_A = H_1(aP)d_A + aP_A$. Similarly, participants B and C choose ephemeral private values $b, c \in_R \mathbb{Z}_q^\times$ and broadcast (P_B, T_B) and (P_C, T_C), respectively.

Key Computation Upon receiving (P_B, T_B) from B and (P_C, T_C) from C, participant A verifies that

$$e(T_B + T_C, P) = e(H(P_B)Q_B + H(P_C)Q_C, P_{pub}) \cdot e(P_B, P_B) \cdot e(P_C, P_C).$$

If the verification holds, participant A computes the session key as

$$K = e(P_B, P_C)^a = e(P, P)^{abc}.$$

Similarly, participants B and C verify their respective signature sets and compute the session key K.

4.2 Batch Verification

Suppose that a user has to verify a collection of signatures on potentially different messages generated by potentially different signers. Requiring the user to verify each signature individually can be computationally expensive. For example, suppose that the ChCh-IBS scheme is used in an authenticated group key agreement protocol for a group of n members and every member is required to individually verify the key contributions of each of the other $n - 1$ members in the group. Each group member will have a heavy computational load of $2(n - 1)$ pairing computations in that communication round, creating a performance bottleneck. However, we can substantially reduce the computational costs by verifying all the signatures at once, known as *batch verification.*

The first efficient batch verifiable version of the digital signature algorithm (DSA) of [FIP93] was proposed by Naccache *et al.* in [NMVR94], but a security flaw found by Lim and Lee in [LL94] allows an attacker to forge multiple signatures that will pass the batch verification stage. Several pairing- and identity-based signature schemes supporting some form of batch verification have been proposed in [ZK03, BGLS03, CKY04, YCK04, ZSNS04, CHP07, FGHP08]. In [CLX06], Cao *et al.* showed that the attack of Lee and Lim translates to the schemes of [ZK03, CKY04, YCK04, ZSNS04] as well. This security flaw allows us to define two different forms of batch verification algorithms: *signature screening* and *strong batch verification.*

In this section, we discuss the concept of batch verification as it relates to identity-based signature schemes. In Section 4.2.1, we discuss signature screening and illustrate the concept using the Cha-Cheon signature (ChCh-IBS) scheme of Section 4.1.3. In Section 4.2.2, we formally define the concept of strong batch verification for ID-based signature schemes and illustrate how a signature screening algorithm can be

transformed into a strong batch verification algorithm.

We note that the traditional notion of batch verification is defined in the context of a collection of different signatures on different messages generated by a single signer. Throughout this section, we view a batch of signatures as a collection of multiple signatures on multiple message generated by multiple signers, where each message is signed by a distinct user, as discussed in [YCK04].

4.2.1 Signature Screening

The concept of signature screening was defined by Bellare, Garay and Rabin in [BGR98] in response to the attack on the batch verifiable signature scheme of Naccache *et al.* from [NMVR94]. Informally speaking, a screening algorithm will detect a signature forgery with high probability, but may not detect whether a valid signature from a previous session has been modified. We illustrate the concept of signature screening by presenting a screening algorithm for the ChCh-IBS scheme of Section 4.1.3.

Example

Suppose we want to verify a batch of signatures $\{(\mathsf{ID}_1, m_1, \sigma_1), ..., (\mathsf{ID}_n, m_n, \sigma_n)\}$, where each $\sigma_i = (X_i, Z_i)$ is the ChCh-IBS signature of a user ID_i, with public/private key pair (Q_i, d_i), on a message m_i. We verify that

$$e(\sum_{i=1}^{n} Z_i, P) = e(\sum_{i=1}^{n} (X_i + h_i Q_i), P_{pub}),$$

where $h_i = H_1(X_i, m_i)$ and $H_1 : \mathbb{G}_1 \times \{0,1\}^* \rightarrow \mathbb{Z}_q^{\times}$. Consistency follows from the consistency of the **Verify** algorithm of the ChCh-IBS scheme and the bilinearity of the pairing.

However, suppose the batch of signatures is modified so that $\sigma_1' = (X_1, Z_1 + \alpha)$ and $\sigma_2' = (X_2, Z_2 - \alpha)$, where $\alpha \in \mathbb{G}_1^{\times}$. Clearly these invalid signatures will still pass the screening test, since

$$(Z_1 + \alpha) + (Z_2 - \alpha) = Z_1 + Z_2.$$

Thus the screening process does not ensure the validity of each individual signature.

While screening is a weak form of batch verification, some applications may not be concerned with the vulnerabilities. For example, an aggregate signature scheme, proposed by Boneh *et al.* in [BGLS03], allows a batch of signatures to be compressed into a single short signature that can be verified using an aggregate-verification algorithm. As discussed in [CHP07], screening is the strongest form of batch verification that can be provided in most aggregate signature schemes.

4.2.2 Strong Batch Verification

To complement the notion of signature screening, Bellare *et al.* defined the concept of strong batch verification in [BGR98]. Similar to signature screening, strong batch verification will detect a forgery within a batch of signatures with high probability. However, a strong batch verification algorithm will also detect with high probability, whether each individual signature in the batch is valid. More formally, we present the following definition for strong batch verification of identity-based signatures, adapted from [BGR98, CHP07].

Definition 4.3 *Let ℓ be the security parameter. Suppose we have an identity-based signature scheme $\mathcal{S} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Sign}, \mathsf{Verify})$, $n \in poly(\ell)$, $\mathsf{Setup}(1^\ell) = \langle \mathsf{msk}, \mathsf{params} \rangle$ and $(\mathsf{ID}_1, d_1), ..., (\mathsf{ID}_n, d_n)$ are generated independently by* Extract. *A probabilistic algorithm* Batch *is called a* strong batch verification *algorithm if the following conditions hold:*

i. If $\mathsf{Verify}(\mathsf{ID}_i, m_i, \sigma_i) = 1$ for all $1 \leq i \leq n$, then

$$\mathsf{Batch}\left((\mathsf{ID}_1, m_1, \sigma_1), ..., (\mathsf{ID}_n, m_n, \sigma_n)\right) = 1.$$

ii. If $\mathsf{Verify}(\mathsf{ID}_i, m_i, \sigma_i) = 0$ for any $1 \leq i \leq n$, then

$$\mathsf{Batch}\left((\mathsf{ID}_1, m_1, \sigma_1), ..., (\mathsf{ID}_n, m_n, \sigma_n)\right) = 0$$

except with negligible probability.

In [BGR98], Bellare *et al.* proposed three separate techniques, known as the *small exponents test*, *random subset test* and the *bucket test*, that can be applied to transform a signature screening algorithm into a strong batch verification algorithm. In [CLX06], Cao *et al.* applied a pairing-based version of the small exponents test to the schemes of [CKY04, YCK04, ZK03, ZSNS04] to provide strong batch verification. Similarly, Ferrara *et al.* used the pairing-based version of the small exponents test in [FGHP08] to produce strong batch verification algorithms for several popular pairing- and identity-based signature schemes, including the ChCh-IBS scheme of Section 4.1.3 (which was also proposed in [LM07]) and the H-IBS scheme of Section 4.1.4. We demonstrate the strong verification algorithm of the ChCh-IBS scheme by applying the pairing-based version of the small exponents test to the signature screening example of the previous section.

Example

Suppose we want to verify a batch of signatures $\{(\mathsf{ID}_1, m_1, \sigma_1), ..., (\mathsf{ID}_n, m_n, \sigma_n)\}$, where each $\sigma_i = (X_i, Z_i)$ is believed to be the ChCh-IBS signature of user ID_i on a message m_i. We choose a vector $(\delta_1, ..., \delta_n) \in_R \left(\mathbb{Z}_q^{\times}\right)^n$ and verify that

$$e(\sum_{i=1}^{n} \delta_i Z_i, P) = e(\sum_{i=1}^{n} (\delta_i X_i + \delta_i h_i Q_i), P_{pub}).$$

where $h_i = H_1(X_i, m_i)$.

We see that the batch $\{\sigma_1, ..., \sigma_n\}$ from the previous example will not pass this test, since

$$e(\delta_1 Z_1 + \delta_1 \alpha + \delta_2 Z_2 - \delta_2 \alpha, P) \neq e(\delta_1 Z_1 + \delta_2 Z_2, P),$$

unless $\delta_1 \alpha = \delta_2 \alpha$, which happens with negligible probability.

For practical applications of batch verification algorithms, we require a method for identifying invalid signatures when the batch verification algorithm fails. In [LM07, Mat09], Mat and Law outlined several algorithms for finding invalid signatures in the strong batch verification algorithm of the ChCh-IBS scheme. Through practical implementations of the algorithms in [FGHP08], Ferrara *et al.* discovered that if no more than 15% of the signatures were invalid, strong batch verification is still more efficient than verifying all the signatures individually. For a thorough discussion on

the different algorithms, we refer the reader to [LM07, Mat09].

4.3 Identity-based Signcryption Schemes

In this section, we discuss the concept of identity-based signcryption schemes. We discuss the different types of signcryption schemes in Section 4.3.1, give a formal definition of identity-based signcryption in Section 4.3.2 and discuss the various properties of signcryption schemes in Section 4.3.3. In Section 4.3.4, we combine the security models for IBE schemes (see Section 3.3.2) and IBS schemes (see Section 4.1.2), to define the formal model for identity-based signcryption schemes. To illustrate the concept of ID-based signcryption, we present the scheme of Chen and Malone-Lee in Section 4.3.5 and a multi-receiver variant of their scheme, due to Yu *et al.*, in Section 4.3.6.

4.3.1 Types of Signcryption

In [Zhe97], Zheng defined signcryption in terms of a single *signcrypt* algorithm that simultaneously provides confidentiality and authentication in an efficient manner. This single algorithm approach is often referred to as *monolithic signcryption.* However, An *et al.* [ADR02] suggest that efficiency may not be our only concern when designing such a scheme and proposed that the term signcryption refer to *any* secure scheme that provides both confidentiality and authentication in the public key setting. The authors go on to formally define four basic approaches to signcryption: *sign-then-encrypt* ($\mathcal{S}t\mathcal{E}$), *encrypt-then-sign* ($\mathcal{E}t\mathcal{S}$), *commit-then-encrypt-and-sign* ($Ct\mathcal{E}\&\mathcal{S}$) and the so-called *monolithic signcrypt* of Zheng. We note that the $\mathcal{S}t\mathcal{E}$ and $\mathcal{E}t\mathcal{S}$ don't necessarily refer to the sequential composition of independent signature and encryption algorithms, but may also refer to *cooperative* $\mathcal{S}t\mathcal{E}$ and $\mathcal{E}t\mathcal{S}$ schemes, where one algorithm has access to some internal ephemeral data of the previous algorithm. As shown in [Boy03, CML05], these cooperative schemes can be very efficient and often produce results comparable to a monolithic signcryption scheme.

In this section, we define the monolithic signcrypt, sign-then-encrypt and encrypt-then-sign schemes in terms of generalized encryption and signature algorithms, as in [ADR02, Smi05]. As the commit-then-encrypt-and-sign approach is less common in

the literature, we omit this type of scheme from our discussion and refer the reader to [ADR02] for more details.

Signcryption Scheme

A *(monolithic) signcryption* scheme $\mathcal{SC} = (\mathsf{KeyGen}, \mathsf{Sign/Encrypt}, \mathsf{Decrypt/Verify})$ is defined in terms of the following algorithms:

KeyGen(1^ℓ) Given a security parameter ℓ, the algorithm outputs $(\mathsf{SDK}, \mathsf{VEK})$, where SDK is the private sign/decrypt key and VEK is the public verify/encrypt key.

Sign/Encrypt$(\mathsf{SDK}_A, \mathsf{VEK}_B, m)$ Given the private key SDK_A of the sender A, the public key VEK_B of the recipient B and a plaintext message m, return the signcryption ciphertext c.

Decrypt/Verify$(\mathsf{VEK}_A, \mathsf{SDK}_B, c)$ Given the public key VEK_A of the sender A, the private key SDK_B of the recipient B and the signcrypted message c, output m if c is a valid signcryption from the sender A, and $\perp$, otherwise. We often refer to this process as *designcryption.*

Sign-then-Encrypt Scheme

Let $\mathcal{S}$ be a signature scheme (Sign-KeyGen, Sign, Verify) and $\mathcal{E}$ be an encryption scheme (Enc-KeyGen, Encrypt, Decrypt). Using these two schemes, we define the *sign-then-encrypt* scheme $\mathcal{St\!E}$ in terms of the following algorithms:

KeyGen(1^ℓ) Given a security parameter ℓ, obtain the key pair $(\mathsf{SK}, \mathsf{VK})$ from Sign-KeyGen, where SK is the private signing key and VK is the public verification key, and the key pair $(\mathsf{EK}, \mathsf{DK})$ from Enc-KeyGen, where EK is the public encryption key and DK is the private decryption key. Return the key pair $(\mathsf{SDK}, \mathsf{VEK})$, where $\mathsf{SDK} = (\mathsf{SK}, \mathsf{DK})$ is the private sign/decrypt key and $\mathsf{VEK} = (\mathsf{VK}, \mathsf{EK})$ is the public verify/encrypt key.

Sign/Encrypt$(\mathsf{SDK}_A, \mathsf{VEK}_B, m)$ Given the private key pair $\mathsf{SDK}_A = (\mathsf{SK}_A, \mathsf{DK}_A)$ of the sender A, the public key pair $\mathsf{VEK}_B = (\mathsf{VK}_B, \mathsf{EK}_B)$ of the recipient B and a plaintext message m, obtain the signature σ by signing the message m using

the Sign algorithm and the private signing key SK_A of the sender. Obtain the ciphertext c by encrypting the message "$m \parallel \sigma$" for the recipient B using the Encrypt algorithm and the public encryption key EK_B. Return the ciphertext c.

Decrypt/Verify$(\mathsf{VEK}_A, \mathsf{SDK}_B, c)$ Given the public key pair $\mathsf{VEK}_A = (\mathsf{VK}_A, \mathsf{EK}_A)$ of the sender A, the private key pair $\mathsf{SDK}_B = (\mathsf{SK}_B, \mathsf{DK}_B)$ of the recipient B and the ciphertext c, recover the message "$m \parallel \sigma$" by decrypting the ciphertext c using the Decrypt algorithm and the private decryption key DK_B of the recipient. Verify the signature σ on the message m using the Verify algorithm and the public verification key VK_A of the sender. If the verification is successful, return the message m, and $\perp$, otherwise.

Encrypt-then-Sign Scheme

Similar to the previous section, let $\mathcal{S}$ be a signature scheme (Sign-KeyGen, Sign, Verify) and $\mathcal{E}$ be an encryption scheme (Enc-KeyGen, Encrypt, Decrypt). Using these two schemes, we define an *encrypt-then-sign* scheme $\mathcal{E}t\mathcal{S}$ in terms of the following algorithms:

KeyGen(1^ℓ) As in the sign-then-encrypt scheme.

Sign/Encrypt$(\mathsf{SDK}_A, \mathsf{VEK}_B, m)$ Given the private key pair $\mathsf{SDK}_A = (\mathsf{SK}_A, \mathsf{DK}_A)$ of the sender A, the public key pair $\mathsf{VEK}_B = (\mathsf{VK}_B, \mathsf{EK}_B)$ of the recipient B and a plaintext message m, obtain the encryption ciphertext e by encrypting the message m for the recipient B using the Encrypt algorithm and the public encryption key EK_B. Obtain the signature σ by signing the encryption ciphertext e using the Sign algorithm and the private signing key SK_A of the sender. Return the ciphertext $c = (e, \sigma)$.

Decrypt/Verify$(\mathsf{VEK}_A, \mathsf{SDK}_B, c)$ Given the public key pair $\mathsf{VEK}_A = (\mathsf{VK}_A, \mathsf{EK}_A)$ of the sender A, the private key pair $\mathsf{SDK}_B = (\mathsf{SK}_B, \mathsf{DK}_B)$ of the recipient B and the ciphertext c, verify the signature σ on the encryption ciphertext e using the Verify algorithm and the public verification key VK_A of the sender. If the verification is successful, recover the message m by decrypting the encryption

ciphertext e using the Decrypt algorithm and the private decryption key DK_B of the recipient B, and return the message m. Otherwise, return the value $\perp$.

4.3.2 Identity-based Signcryption

We formally define the concept of identity-based signcryption using the monolithic signcrypt approach. However, we note that this definition may be modified to allow for the sign-then-encrypt or encrypt-then-sign approaches.

Definition 4.4 *An* identity-based signcryption scheme *consists of three randomized algorithms* **Setup**, **Extract** *and* **Sign/Encrypt**, *and one deterministic algorithm* **Decrypt/Verify**. *They are described as follows:*

Setup(ℓ)**:** Given a security parameter ℓ, the PKG generates the pair $\langle$msk,params$\rangle$, where msk is the master secret key and params refer to the public system parameters. These parameters include a global public key P_{pub} and a description of the finite message space $\mathcal{M}$, finite signature space Σ and finite ciphertext space $\mathcal{C}$.

Extract(params, msk, ID)**:** Upon input of params, the master secret key msk and an identity string $\mathsf{ID} \in \{0,1\}^*$, the PKG returns the private key d_{ID} corresponding to the user ID.

Sign/Encrypt(params,$d_{\mathsf{A}}, \mathsf{ID}_b, m$) Upon input of params, the private key d_A of the sender A, the identity ID_B of the recipient B and a plaintext message $m \in \mathcal{M}$, the algorithm returns the signcrypted ciphertext c.

Decrypt/Verify(params,d_B, ID_A, c) Upon input of params, the private key d_B of the recipient B, the identity ID_A of the sender A and the signcrypted ciphertext c, the algorithm returns the message m if c decrypts into a message bearing A's signature, or $\perp$ otherwise.

The above algorithms must satisfy the consistency requirement that, given a ciphertext c = Sign/Encrypt(params, d_A, ID_B, m), we have m = Decrypt/Verify(params, d_B, ID_A, c).

4.3.3 Properties of Signcryption Schemes

We give a number of definitions that are relevant in discussions of signcryption schemes.

Message Confidentiality A signcryption scheme provides *message confidentiality* if only the intended recipient can recover the signcrypted message. This property is essential to the security of a signcryption scheme, as discussed in Section 4.3.4.

Non-repudiation A signcryption scheme provides *non-repudiation* if the sender A of a signcrypted message cannot later deny having sent the message. As discussed in [ML05], signcryption non-repudiation is not a straightforward consequence of unforgeability, as is the case for signature schemes. In particular, we may consider signcryption non-repudiation with respect to the signcryption ciphertext or with respect to the signature embedded in the ciphertext. We usually associate ciphertext non-repudiation with the monolithic approach and signature non-repudiation with the two-layer approach, as discussed in [ADR02]. The nature of a monolithic signcryption scheme ensures that a ciphertext can only be verified by the intended recipient. Thus a third party cannot settle repudiation disputes without access to some private information. However, there are methods to provide non-repudiation without compromising private information, as illustrated by the following property.

$\mathcal{S}$-Verifiability In [SLS03], Shim *et al.* proposed a specific form of non-repudiation, which they called signature verifiability. Given a signature scheme $\mathcal{S}$, a signcryption scheme is said to be *$\mathcal{S}$-verifiable* if the **Decrypt/Verify** algorithm additionally outputs a *detachable signature* σ for the signcrypted message that the responder can verify using $\mathcal{S}$. As long as the signature scheme $\mathcal{S}$ provides non-repudiation, the detachable signature may be forwarded to a third party for verification.

Surreptitious Forwarding A signcryption scheme is susceptible to *surreptitious forwarding* if the recipient B of a signcrypted message from the sender A can re-encrypt the message for a third party C in such a way that C believes it

was from the original sender A. Signcryption schemes using a naive sign-then-encrypt approach are often vulnerable to this type of attack.

Forward Security A signcryption scheme is said to be *forward secure* if the disclosure of the signer's private key does not allow an adversary to recover the signcrypted message. Conversely, a signcryption scheme is said to provide *past message recovery* if the signer can recover the signcrypted message using *only* her private key.

Multi-Receiver Signcryption A signcryption scheme provides *efficient support for multiple recipients* and is known as a *multi-receiver signcryption scheme*, if the sender A can signcrypt a message for a group of recipients without executing the signcrypt algorithm for each individual recipient. In general, most sign-then-encrypt schemes can be securely extended to support multiple recipients, provided that they satisfy some slight modifications to the security proofs, as shown in [Boy03].

4.3.4 Security of Signcryption Schemes

Since signcryption combines the concepts of encryption and signature schemes, the standard notion of security for signcryption schemes is message confidentiality (i.e. indistinguishability under adaptive chosen ciphertext attacks (IND-CCA2) defined in Section 3.3.2) and signature non-repudiation (i.e. existential unforgeability under adaptive chosen message attacks (EUF-CMA) defined in Section 4.1.2). We define the formal model for analyzing ID-based signcryption schemes as given by Chen and Malone-Lee in [CML05]. We discuss the properties in terms of both *insider security* and *outsider security* from [ADR02]. We view insider security as security against either the sender or receiver, depending on the security notion, and outsider security as security against a third party.

Adversarial Model

Working from the IND-ID-CCA2 security game of Section 3.3.2 and the EUF-CMA security game of Section 4.1.2, we prove the security of a signcryption scheme by showing that an adversary has no advantage in breaking a simulation of the scheme.

We define the simulation in terms of a security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, where the adversary's capabilities are modelled by allowing her to issue the following queries to the challenger:

Extract(ID_i) Returns the private key d_i corresponding to ID_i obtained from the **Extract** algorithm.

Sign/Encrypt($m, \mathsf{ID}_A, \mathsf{ID}_B$) Returns the message m signed by the sender ID_A and encrypted under the public key of the receiver ID_B.

Decrypt/Verify($c, \mathsf{ID}_B, \mathsf{ID}_A$) Returns the decrypted message m for sender ID_A if, under the recipient's private key d_B, the ciphertext c decrypts to a valid message signed by ID_A. Otherwise, the algorithm returns $\perp$.

For each query, we assume that ID_A is different from ID_B. We also assume that the signcryption scheme to be proven secure takes a monolithic approach. When proving the security of a cooperative $\mathcal{S}t\mathcal{E}$ (resp. $\mathcal{E}t\mathcal{S}$) scheme, we must allow the **Encrypt** (resp. **Sign**) algorithm to have access to the ephemeral data of the **Sign** (resp. **Extract**) algorithm, and thus we will have the same monolithic oracle queries.

Message Confidentiality

We discuss message confidentiality in terms of an adversary's ability to distinguish between signcrypted ciphertexts in a forward secure signcryption scheme.

IND-IBSC-CCA2 Security Game The security of an identity-based signcryption scheme $\mathcal{SC}$ is defined in terms of the following security game between a distinguisher $\mathcal{D}$ and a challenger $\mathcal{C}$:

Phase 1: The distinguisher $\mathcal{D}$ issues queries from above to the challenger. Afterwards, $\mathcal{D}$ outputs two identities $\{\mathsf{ID}_A^*, \mathsf{ID}_B^*\}$ and two messages $\{m_0, m_1\}$ upon which to be tested. We require that $\mathcal{D}$ has not made an Extract query on ID_B^*.

Challenge: The challenger $\mathcal{C}$ chooses a random bit $\theta \in \{0, 1\}$ and computes the ciphertext c^* from Sign/Encrypt for the triple $(m_\theta, \mathsf{ID}_A^*, \mathsf{ID}_B^*)$. $\mathcal{C}$ returns the ciphertext c^* to $\mathcal{D}$.

Phase 2: $\mathcal{D}$ continues to query the oracles, but may not issue the $\mathsf{Extract}(\mathsf{ID}_B^*)$ or $\mathsf{Decrypt/Verify}(c^*, \mathsf{ID}_B^*, \mathsf{ID}_A^*)$ queries.

Response: $\mathcal{D}$ returns a guess θ' for the value of the bit θ. The adversary wins the game if $\theta' = \theta$.

From this security game, we formally define the notion of ciphertext indistinguishability for ID-based signcryption schemes.

Definition 4.5 *Let* Distinguish *be the event that the distinguisher $\mathcal{D}$ can successfully distinguish between ciphertexts. The advantage of $\mathcal{D}$ in attacking $\mathcal{SC}$ is defined as*

$$\mathrm{Adv}_{\mathcal{D},\mathcal{SC}}^{\mathsf{Distinguish}}(\ell) = |2 \cdot \Pr[\theta' = \theta] - 1|$$

for the security parameter ℓ. We say that an identity-based signcryption scheme $\mathcal{SC}$ is semantically secure (or indistinguishable) under adaptive chosen ciphertext attacks *(IND-IBSC-CCA2) if $\mathcal{D}$ has a negligible advantage.*

We note that this definition involves insider security, as the private key of the sender may be disclosed without compromising confidentiality.

Signature Unforgeability

We discuss unforgeability in terms of the adversary's ability to forge a signature within the ciphertext. We note that the following model is a natural adaptation of the accepted security notion for digital signature schemes, the EUF-CMA security model from Section 4.1.2, as discussed in [CML05].

EUF-IBSC-CMA Security Game The security of an identity-based signcryption scheme $\mathcal{SC}$ is defined in terms of the following security game between a forger $\mathcal{F}$ and a challenger $\mathcal{C}$:

Phase 1: The forger $\mathcal{F}$ makes queries from above to the challenger.

Forgery: $\mathcal{F}$ returns a triple $(c^*, \mathsf{ID}_A^*, \mathsf{ID}_B^*)$. Suppose that the output of the $\mathsf{Decrypt/Verify}$ oracle for the ciphertext c^* under the private key of ID_B^* is the message m.

Result: The forger wins the game if m is a valid message and if $\mathcal{F}$ made no $\mathsf{Extract}(\mathsf{ID}_A^*)$ query and no query of the form $\mathsf{Sign/Encrypt}(m, \mathsf{ID}_A^*, \mathsf{ID}_B) = c$ such that $\mathsf{Decrypt/Verify}(c, \mathsf{ID}_B, \mathsf{ID}_A^*) = m$.

From this security game, we formally define the notion of existential unforgeability for ID-based signcryption schemes.

Definition 4.6 *Let* Forge *be the event that the forger $\mathcal{F}$ successfully forges a signature within the ciphertext for $\mathcal{SC}$. The advantage of $\mathcal{F}$ in attacking $\mathcal{SC}$ is defined as*

$$\mathrm{Adv}_{\mathcal{F},\mathcal{SC}}^{\mathsf{Forge}}(\ell) = \Pr[\mathsf{Forge}]$$

for the security parameter ℓ. We say that the signcryption scheme $\mathcal{SC}$ is existentially unforgeable against chosen message attacks *(EUF-IBSC-CMA) if $\mathcal{F}$ has a negligible advantage.*

We note that, by allowing the forger to have access to the private key of the recipient ID_B, we obtain insider security.

4.3.5 Chen and Malone-Lee Signcryption Scheme

Improving Boyen's scheme from [Boy03], Chen and Malone-Lee [CML05] proposed one of the most efficient identity-based signcryption schemes to date, which we call the CML-IBSC scheme. Their scheme follows the cooperative sign-then-encrypt approach, using the ChCh-IBS scheme from Section 4.1.3 and an encryption process similar to the `BasicIdent` scheme of Boyen and Franklin from Section 3.4.1.

The CML-IBSC scheme is characterized by the following algorithms:

Setup: Given a security parameter $\ell \in \mathbb{Z}^+$, the algorithm outputs the public system parameters

$$\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_{pub}, H_0, H_1, H_2 \rangle ,$$

where $\langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_{pub} \rangle$ are as defined in Section 3.4 and H_0, H_1 and H_2 are hash functions defined by $H_0 : \{0,1\}^* \to \mathbb{G}_1^\times$, $H_1 : \mathbb{G}_2 \to \mathbb{Z}_q^\times$ and $H_2 : \mathbb{G}_1 \times \{0,1\}^* \to \mathbb{Z}_q^\times$. The PKG publishes the message space $\mathcal{M} = \{0,1\}^*$, the

ciphertext space $\Sigma = \mathbb{G}_1^\times \times \{0,1\}^*$ and **params** and keeps the master secret key $s \in \mathbb{Z}_q^\times$ hidden.

Extract: As in the ChCh-IBS scheme of Section 4.1.3.

Sign: To sign a message $m \in \{0,1\}^*$, the signer A with identity ID_A and public/private key pair (Q_A, d_A):

1. Chooses $r \in_R \mathbb{Z}_q^\times$.
2. Computes
   i) $X = rQ_A$,
   ii) $h = H_2(X, m)$,
   iii) $Z = (r + h)d_A$.
3. Returns the signature (X, Z) and forwards (m, r, X, Z) to the **Encrypt** algorithm.

Encrypt: To encrypt the message m for a recipient B, with identity ID_B and public/private key pair (Q_B, d_B), using (m, r, X, Z), the signer:

1. Computes
   i) $w = e(rd_A, Q_B)$,
   ii) $y = H_1(w) \oplus (Z \parallel \mathsf{ID}_A \parallel m)$. [1]
2. Returns the ciphertext $c = (X, y)$.

Decrypt: To decrypt the ciphertext (X, y), the recipient B:

1. Computes
   i) $w = e(X, d_B)$
   ii) $(Z \parallel \mathsf{ID}_A \parallel m) = y \oplus H_1(w)$

[1] Although not discussed in [CML05], the scheme would require some invertible function $f : \mathbb{G}_1 \to \{0,1\}^*$ to map the element $Z \in \mathbb{G}_1$ to a bitstring.

2. Forwards the message m, signature $\sigma = (X, Z)$ and signer's identity ID_A to the **Verify** algorithm.

Verify: To verify that the signature (X, Z) on the message m belongs to user A, the recipient B:

1. Computes $h = H_2(X, m)$.

2. Returns $\top$ if $e(Z, P) = e(X + hQ_A, P_{pub})$. Else, returns $\perp$.

The consistency of the designcryption stage follows from the bilinearity property of the pairing. Given the ciphertext $c = (X, y)$ and signature $\sigma = (X, Z)$, we have

$$w = e(rd_A, Q_B) = e(rsQ_A, Q_B) = e(rQ_A, sQ_B) = e(X, d_B),$$

and

$$e(X + hQ_A, P_{pub}) = e((r + h)\, Q_A, sP) = e(Z, P).$$

4.3.6 The YYHZ Multi-Receiver Signcryption Scheme

Although the CML-IBSC scheme of Section 4.3.5 can support multiple recipients, Yu *et al.* made some modifications in [YYHZ07] to produce an efficient multi-receiver signcryption scheme, which we call the YYHZ-IBSC scheme. However, in [SVG+08a], Sharmila Deva Selvi *et al.* showed that the proofs for confidentiality and authentication were flawed. They proposed some slight modifications and formally proved that their modified multi-receiver signcryption scheme is secure. We present the YYHZ-IBSC scheme as given by Sharmila Deva Selvi *et al.* in [SVG+08a].

The YYHZ-IBSC scheme is characterized by the following algorithms:

Setup: Given a security parameter $\ell \in \mathbb{Z}^+$, the algorithm outputs the public system parameters

$$\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_{pub}, R, \theta, H_0, H_1, H_2 \rangle .$$

where $\langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_{pub}, H_0, H_1 \rangle$ are as defined in the CML-IBSC scheme of Section 4.3.5 and the hash function H_2 is given by $H_2 : \mathbb{G}_1^2 \times \{0,1\}^* \to \mathbb{Z}_q^\times$. In

addition, the PKG chooses a public point $R \in_R \mathbb{G}_1^\times$ and computes

$$\theta = e(R, P_{pub}).$$

The PKG publishes the message space $\mathcal{M} = \{0,1\}^*$, the ciphertext space $\Sigma = \mathbb{G}_1^\times \times \{0,1\}^*$ and params and keeps the master secret key $s \in \mathbb{Z}_q^\times$ hidden.

Extract: As in the ChCh-IBS scheme of Section 4.1.3.

Sign/Encrypt: To signcrypt a message $m \in \{0,1\}^*$ for the recipients $(\mathsf{ID}_1, ..., \mathsf{ID}_n)$ with respective ID-based public/private key pairs $(Q_1, d_1), ..., (Q_n, d_n)$, the sender A with identity ID_A and public/private key pair (Q_A, d_A):

1. Chooses $r, t \in_R \mathbb{Z}_q^\times$.
2. Computes
   i) $X = rQ_A$,
   ii) $U = tP$,
   iii) $h = H_2(X, U, \text{“}\mathsf{ID}_A \parallel m\text{”})$,
   iv) $Z = (r + h)d_A$,
   v) $\omega = e(Z, P)$,
   vi) $y = (X \parallel Z \parallel m) \oplus H_1(\omega)$, [2]
   vii) $W = \omega \cdot \theta^t$.
3. Computes $c_i = t(Q_i + R)$ for $1 \leq i \leq n$.
4. Returns the ciphertext $\langle y, U, W, c_1, c_2, ..., c_n \rangle$, along with some list indicating that the ciphertext c_i corresponds to the recipient ID_i, for $1 \leq i \leq n$.

Decrypt/Verify: To decrypt the ciphertext set $\langle y, U, W, c_i \rangle$, the recipient with identity ID_i:

[2] As in the CML-IBSC scheme, we require some invertible function $f : \mathbb{G}_1 \to \{0,1\}^*$ to map X and Z to strings.

1. Computes

   i) $\omega = W \cdot e(d_i, U) \cdot e(c_i, P_{pub})^{-1}$,

   ii) $X \parallel Z \parallel m = y \oplus H_1(\omega)$,

   iii) $h = H_2(X, U, \text{“}\mathsf{ID}_A \parallel m\text{”})$.

2. If $\omega = e(X + hQ_A, P_{pub})$, returns m, and $\perp$, otherwise.

The consistency of the designcryption stage follows from the bilinearity property of the pairing. Given a ciphertext $\langle y, U, W, c_i \rangle$, we have

$$\begin{aligned} W \cdot e(d_i, U) \cdot e(c_i, P_{pub})^{-1} &= \omega \cdot \theta^t \cdot e(sQ_i, tP) \cdot e(t(Q_i + R), P_{pub})^{-1} \\ &= \omega \cdot e(R, P_{pub})^t \cdot e(sQ_i, tP) \cdot e(tQ_i + tR, P_{pub})^{-1} \\ &= \omega \cdot e(R, P_{pub})^t \cdot e(tQ_i, sP) \cdot e(tQ_i + tR, P_{pub})^{-1} \\ &= \omega \cdot e(tR, P_{pub}) \cdot e(tQ_i, P_{pub}) \cdot e(tQ_i, P_{pub})^{-1} \cdot e(tR, P_{pub})^{-1} \\ &= \omega \cdot e(tR, P_{pub}) \cdot e(tR, P_{pub})^{-1} \\ &= \omega, \end{aligned}$$

and

$$e(X + hQ_A, P_{pub}) = e((r + h)\, Q_A, sP) = e(Z, P) = \omega.$$

4.4 Summary

In this chapter, we discussed signature and signcryption schemes in the context of an identity-based public key infrastructure. In particular, we provided formal definitions of identity-based signature and signcryption schemes and looked at the various properties of the schemes and the models used to analyze their security. To illustrate the concepts, we presented the identity-based signature schemes of Cha-Cheon and Hess, the identity-based signcrypion schemes of Chen-Malone-Lee and Yu *et al.*, and showed how Zhang *et al.* used identity-based signatures to produce an efficient authenticated version of Joux's tripartite key exchange protocol. In addition, we discussed the concept of batch verification and illustrated the differences between signature screening and strong batch verification using the Cha-Cheon signature scheme.

This chapter lays much of the groundwork for topics to be covered in future chapters. In Chapter 5, we present several identity-based group key agreement protocols that achieve authentication using adaptations of the signature schemes presented in this chapter, and in Chapter 7, we use the signature schemes of Cha-Cheon and Hess to produce a new, forward secure identity-based authenticated group key agreement protocol. The concepts presented in this chapter are essential to our understanding of Chapter 6, where we present a framework for producing computationally asymmetric group key agreement protocols from identity-based signcryption schemes. We further show that the most efficient protocols can be produced by choosing signcryption schemes that satisfy some certain properties, presented in this chapter, such as support for multiple recipients and strong batch verification. To illustrate our framework, we generate a computationally asymmetric group key agreement protocol using the multi-receiver signcryption scheme of Yu *et al.*, presented in this chapter.

Chapter 5

Group Key Agreement from Pairings

In this chapter, we analyze how group key agreement (GKA) protocols can be improved through the application of bilinear pairings. After an extensive search, yielding dozens of pairing-based GKA protocols, we arrived at two important observations. Firstly, although group key agreement from pairings is a very active area of cryptographic research, a literature survey summarizing the dozens of protocols has not yet been published. This chapter serves as the first formal survey on pairing-based group key agreement from the literature. More importantly, while analyzing the various pairing-based GKA protocols, we found that every protocol that provided the necessary property of group key secrecy (see Section 2.1.2), was a pairing-based version of one of the following three classes of GKA protocols: the Tree-based Group Diffie-Hellman (TGDH) protocol from Section 2.3.4, the Burmester-Desmedt (BD) protocol from Section 2.3.5 or the computationally asymmetric Conference Key Agreement (CKA) protocol from Section 2.3.6. In our survey, we first present a generalized pairing-based version for each class of GKA protocol and then discuss the details of each specific protocol. We compare the pairing-based variations with the original GKA protocol in terms of the security and contributory properties from Section 2.2.1 and computational and communication efficiency. As a result of this analysis, we determine the most efficient pairing-based GKA protocol for each class, which will serve as a comparison for our pairing-based GKA protocol in Chapter 7. Thus, this

chapter also serves as a novel classification of the current pairing-based solutions to group key agreement.

For each class, we present the original and most influential pairing-based adaptations, while also considering elliptic curve variants that use pairings in some interesting way. Hence, while our classification is not exhaustive, it does illustrate how classical GKA protocols may be improved through the application of pairings. To maintain consistency with the ID-based schemes of Chapters 3 and 4, we only present ID-based GKA protocols that utilize the ID-based public key infrastructure (ID-PKI) of Boneh and Franklin from Section 3.4.

Unless otherwise stated, we assume that a multicast group of n members $\mathcal{U} = \{U_1, ..., U_n\}$ wish to agree upon a shared session key. Throughout this section, we use the public parameters $\langle \mathbb{G}_1, \mathbb{G}_2, e, q, P \rangle$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of order q, $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a strongly non-degenerate bilinear pairing, as defined in Section 3.1 and P is a generator of $\mathbb{G}_1$. For identity-based GKA protocols, we additionally have $P_{pub} = sP$, where $s \in \mathbb{Z}_q^{\times}$ is the private key generator's (PKG) master secret key, and we assume that each member U_i has long-term public/private key pair (Q_i, d_i). For discussions of computational costs, we let $\mathcal{P}$ represent a pairing operation and $\mathcal{M}$ represent a scalar multiplication in $\mathbb{G}_1$.

5.1 Tree-based Group Key Agreement

In Section 2.3.4 we discussed the Tree-based Group Diffie-Hellman (TGDH) protocol, which allows a group of members associated with the leaves of a tree to agree upon a session key, while inducing computational and communication complexity proportional to the height of the tree. Although the TGDH protocol uses the DH key exchange, we note that any two party key exchange protocol may be used in its place. In this section, we present tree-based GKA protocols from the literature that improve upon the TGDH protocol by replacing the DH key exchange protocol with different pairing- and identity-based key exchange protocols.

5.1.1 Extension of Smart's Protocol : The RN-ID-AGKA Protocol

In [RN02], Reddy and Nalla proposed the first pairing-based protocol to allow a group of $n > 3$ members to agree upon a shared session key. Their identity-based authenticated group key agreement (ID-AGKA) protocol, which we call the RN-ID-AGKA protocol, is a binary tree-based protocol which replaces the DH key exchange of the TGDH protocol with the authenticated ID-based key exchange protocol of Smart from Section 3.5.

The RN-ID-AGKA protocol follows the structure of the TGDH protocol, associating each member of the multicast group with a leaf node of a binary tree. Similar to the TGDH protocol, the RN-ID-AGKA protocol associates every non-root node i with a secret/blinded node key pair $(k_i, bk_i = k_i P)$ and the root node with the group TEK. In addition, the protocol associates each non-root node i with the public/private key pair $(Q_i, d_i = sQ_i)$. Using Smart's key exchange protocol, the secret node key is given by

$$k_i = e(k_{L(i)}Q_{R(i)}, P_{pub}) \cdot e(d_{L(i)}, bk_{R(i)}) = e(d_{R(i)}, bk_{L(i)}) \cdot e(k_{R(i)}Q_{L(i)}, P_{pub}), \quad (5.1.1)$$

where $L(i)$ and $R(i)$ represent the left and right child nodes of the i^{th} node.

Similar to the TGDH protocol, we require a hash function $H_1 : \mathbb{G}_2 \rightarrow \mathbb{Z}_q^\times$ to map the secret key for non-leaf nodes $k_i \in \mathbb{G}_2$ to the group $\mathbb{Z}_q^\times$. For leaf nodes, the pair (Q_i, d_i) corresponds to the identity-based public/private key pair issued to member U_i by the PKG. For a non-leaf node i, the left and right child node members submit the identity $Q_{L(i)}+Q_{R(i)}$ to the PKG, who returns the public key $Q_i = H_2\left(Q_{L(i)} + Q_{R(i)}\right)$, using the hash function $H_2 : \mathbb{G}_1^\times \rightarrow \mathbb{G}_1^\times$, and the private key $d_i = sQ_i$.

We illustrate the RN-ID-AGKA protocol with the following example.

Example

Suppose we have a group of 4 members, $\mathcal{U} = \{U_1, ..., U_4\}$ that agree upon the parameters $\langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, H_1, H_2 \rangle$. As in Smart's protocol, each group member U_i is given the ID-based public/private key pair (Q_i, d_i) and chooses a secret value $k_i \in_R \mathbb{Z}_q^\times$. The members compute the secret key and blinded key trees from Figure 5.1 as follows:

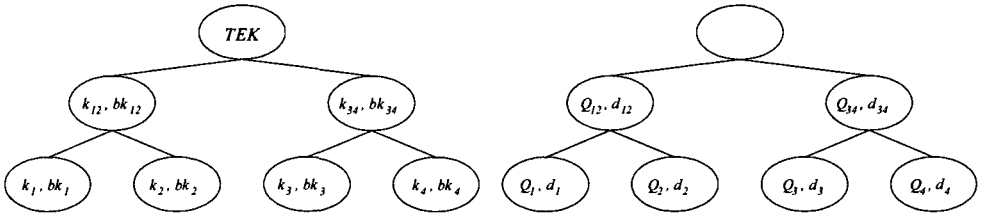

Figure 5.1: Key Trees of the RN-ID-AGKA Protocol

1. Each member U_i computes and broadcasts their blinded key $bk_i = k_i P$.

2. Upon receiving $bk_2 = k_2 P$ from U_2, member U_1 retrieves the public key Q_2 and computes $k_{12} = e(k_1 Q_2, P_{pub}) \cdot e(d_1, bk_2)$ using Equation (5.1.1). Similarly, member U_2 computes the same value $k_{12} = e(k_2 Q_1, P_{pub}) \cdot e(d_2, bk_1)$. Member U_1 computes and broadcasts the blinded key $bk_{12} = H_1(k_{12})P$. Similarly, members U_3 and U_4 compute k_{34} and U_3 broadcasts $bk_{34} = H_1(k_{34})P$.

3. Members $\{U_1, U_2\}$ receive $bk_{34} = H_1(k_{34})P$ from U_3, obtain their ID-based public/private key pair $(Q_{12} = H_2(Q_1 + Q_2), d_{12} = sQ_{12})$ from the PKG and compute the public key $Q_{34} = H_2(Q_3 + Q_4)$. They each use Equation (5.1.1) to compute the group TEK as

$$K = e(H_1(k_{12})Q_{34}, P_{pub}) \cdot e(d_{12}, bk_{34}) = e(d_{34}, bk_{12}) \cdot e(H_1(k_{34})Q_{12}, P_{pub})$$

Members U_3 and U_4 obtain the public/private key pair (Q_{34}, d_{34}) from the PKG and compute the group TEK similarly, using the public keys Q_{12} and bk_{12} and the private keys d_{34} and k_{34}.

Analysis

An advantage of the RN-ID-AGKA protocol over the TGDH protocol is that it provides implicit authentication through the use of Smart's authenticated key exchange protocol. However, a clear disadvantage of the protocol is the extensive role of the PKG. In order to distribute the public/private node key pairs, the PKG requires detailed knowledge of the multicast group and the tree structure, as well as the algorithms and

policies of the key agreement protocol. While most ID-based cryptosystems require the PKG to distribute private keys to members only upon inception into the group communication environment, the RN-ID-AGKA protocol requires the PKG to play an active role in the key agreement process, distributing node key pairs for each round of the protocol.

Recall from Section 3.5 that Smart's protocol only provides partial forward secrecy. Unfortunately, we found that this property translates to the RN-ID-AGKA protocol as well. Indeed, an adversary in possession of the long-term private keys of sibling node members can compute the secret key associated with the parent node as in Smart's protocol. If the adversary can recover the secret keys associated with two adjacent parent nodes, she can compute the secret node key of the next level without knowledge of the long-term private keys of the parent nodes on the current level. For instance, given the long-term private keys $\{d_1, d_2, d_3, d_4\}$, the adversary can recover the keys k_{12} and k_{34} as in Smart's protocol. The adversary can then compute

$$k_{1234} = e(H_1(k_{12})Q_{34}, P_{pub}) \cdot e(H_1(k_{34})Q_{12}, P_{pub})$$

without knowledge of d_{12} or d_{34}. Working in a recursive manner up the key tree, the adversary can compute the group TEK as long as she has knowledge of the long-term private key of *every* member. The security of Smart's protocol implies that if the adversary is missing the long-term private key of a single member, she won't be able to compute the secret key of the parent node and thus can't compute the group session key. Thus, while the RN-ID-AGKA protocol does not provide perfect forward secrecy, it still provides partial forward secrecy since an adversary with knowledge of any proper subset of long-term private keys cannot recover the group session key.

5.1.2 Extension of Joux's Protocol : General Remarks

In Section 3.2, we presented Joux's protocol, which extends the two-party ECDH key exchange to three parties using bilinear pairings. While Joux's protocol is the most natural replacement for the DH key exchange in the TGDH protocol, the first tree-based extension of Joux's protocol came later chronologically, first proposed by Barua *et al.* in [BDS03] in 2003. Since the protocol allows three parties to establish a shared

key, we may associate members with the leaves of a balanced ternary tree rather than the binary tree of TGDH. This effectively reduces the height of the key tree from $\lceil \log_2 n \rceil$ to $\lceil \log_3 n \rceil$, which in turn reduces the computational and round complexity of the protocol. In addition, it shortens the length of the key path, reducing the number of secret node keys that need to be stored.

Several protocols replacing the DH key exchange in TGDH with Joux's tripartite key exchange have been proposed in [BDS03, LKKR04, SHL05]. Although proposed independently, each protocol follows the same basic structure, where every non-root node i of the tree is associated with a secret node key k_i and a blinded key $bk_i = k_i P$ and the root node is associated with the group TEK. Using Joux's protocol, each secret node key is given by

$$k_i = e\left(bk_{L(i)}, bk_{C(i)}\right)^{k_{R(i)}} = e(P,P)^{k_{L(i)}k_{C(i)}k_{R(i)}}, \tag{5.1.2}$$

where $L(i)$, $C(i)$ and $R(i)$ represent the left, center and right child nodes of the i^{th} node. We reiterate the importance that our pairing be strongly non-degenerate (i.e. $e(P,P) \neq 1$ for all $P \in \mathbb{G}_1$), otherwise we obtain a trivial node key. Similar to the RN-ID-AGKA protocol, we need the hash function $H_1 : \mathbb{G}_2 \rightarrow \mathbb{Z}_q^\times$ to map the secret key for non-leaf nodes $k_i \in \mathbb{G}_2$ to the group $\mathbb{Z}_q^\times$.

Before discussing the subtle differences between each particular protocol, we provide an example to illustrate the general key agreement process.

Example

Suppose we have a group of 9 members, $\mathcal{U} = \{U_1, ..., U_9\}$ that agree on the public system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, H_1 \rangle$. Each group member U_i chooses a value $k_i \in_R \mathbb{Z}_q^\times$. The members compute the secret key and blinded key trees from Figure 5.2 as follows:

1. Each member U_i computes and broadcasts their blinded key $bk_i = k_i P$.

2. Member U_1 receives $bk_2 = k_2 P$ from U_2 and $bk_3 = k_3 P$ from U_3 and computes

$$k_{123} = e\left(k_2 P, k_3 P\right)^{k_1} = e(P,P)^{k_1 k_2 k_3}$$

using Equation (5.1.2). Similarly, member U_2 computes $k_{123} = e\left(k_1P, k_3P\right)^{k_2}$ and U_3 computes $k_{123} = e\left(k_1P, k_2P\right)^{k_3}$. Of $\{U_1, U_2, U_3\}$, the member associated with the left-most child node (i.e. U_1) computes and broadcasts the blinded key $bk_{123} = H_1(k_{123})P$. Similarly, $\{U_4, U_5, U_6\}$ compute k_{456} and U_4 broadcasts $bk_{456} = H_1(k_{456})P$ and $\{U_7, U_8, U_9\}$ compute k_{789} and U_7 broadcasts $bk_{789} = H_1(k_{789})P$.

3. Members $\{U_1, U_2, U_3\}$ receive bk_{456} and bk_{789} and use Equation (5.1.2) to compute the group TEK as

$$\begin{aligned} K &= e(bk_{456}, bk_{789})^{H_1(k_{123})} \\ &= e\left(H_1(k_{456})P, H_1(k_{789})P\right)^{H_1(k_{123})} \\ &= e\left(P, P\right)^{H_1(k_{123})\cdot H_1(k_{456})\cdot H_1(k_{789})} \\ &= e(P, P)^{H_1(e(P,P)^{k_1k_2k_3})\cdot H_1(e(P,P)^{k_4k_5k_6})\cdot H_1(e(P,P)^{k_7k_8k_9})}. \end{aligned}$$

Members $U_4, ...U_9$ compute the group TEK similarly.

In the previous example, we specifically chose the group size to be a power of 3 so that the members could be arranged in a perfect ternary tree. The various tree-based extensions of Joux's protocols mainly differ in the construction of the key tree for imperfect cases, as in Figure 5.3, and the manner in which a pair of child nodes compute the secret key of their parent node, which is achieved using some two-party key agreement protocol. Thus we compare the different protocols in terms of the following three points: key tree construction, the particular two-party key agreement protocol used by a pair of child nodes and the manner in which the protocol provides authentication. For simplicity, we illustrate the two-party key agreement protocols by showing how members $\{U_7, U_8\}$ of Figure 5.3 compute the node key k_{78}.

5.1.3 Extension of Joux's Protocol : The BDS-ID-AGKA Protocol

In [BDS03], Barua *et al.* proposed a pairing-based TGDH protocol, which we call the BDS-ID-AGKA protocol.

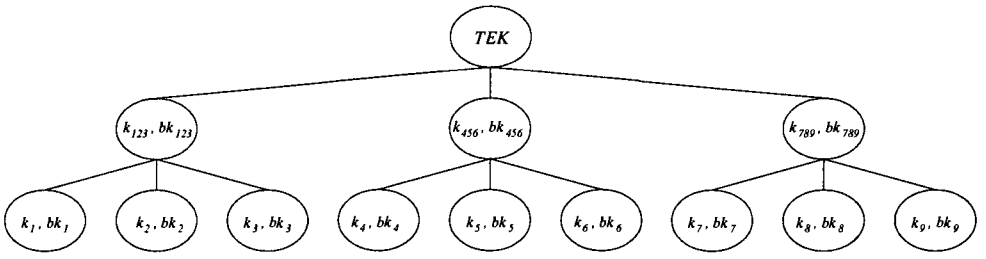

Figure 5.2: Perfect Secret/Blinded Node Key Tree in Tree-based Extensions of Joux's Protocol

Tree Construction

Suppose we have a multicast group $\mathcal{U}$ of size n with $m = \lfloor \frac{n}{3} \rfloor$ and $r = n \bmod 3$. The BDS-ID-AGKA protocol constructs the initial key tree as follows:

1. The group $\mathcal{U}$ is partitioned into three sets of respective sizes

   i) (m, m, m) if $r = 0$,

   ii) $(m, m, m+1)$ if $r = 1$,

   iii) $(m, m+1, m+1)$ if $r = 2$,

   with each subgroup rooted at one of the three level 1 nodes in the tree.

2. Each individual set is recursively partitioned in the same manner, until we obtain a tree of height $h = \lceil \log_3 n \rceil$.

The BDS-ID-AGKA tree construction algorithm distributes the nodes evenly in each subtree of the key tree. For example, a multicast group of size 8 would produce the mirror image of the key tree in Figure 5.3, whereas a group of size 14 would produce a key tree as in Figure 5.4.

Two-Party Node Key Agreement

To compute the node key for a pair of members $\{U_7, U_8\}$, Barua *et al.* choose one member, say U_7, to be a distinguished representative. In addition to computing the blinded key $bk_7 = k_7 P$, member U_7 chooses an additional ephemeral value $k'_7 \in \mathbb{Z}_q^\times$ and computes $bk'_7 = k'_7 P$. The representative U_7 broadcasts both bk_7 and bk'_7 to the

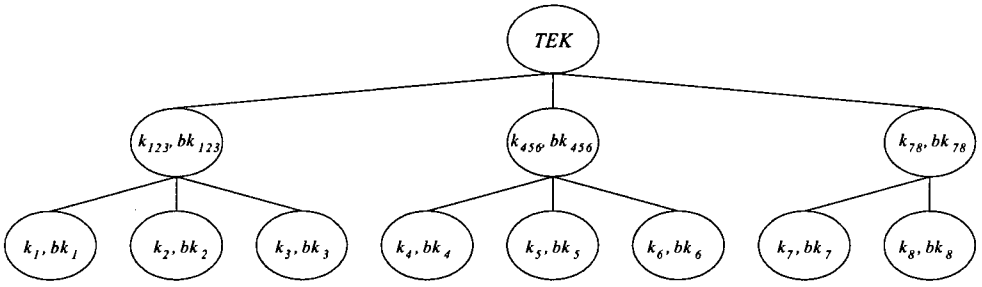

Figure 5.3: Imperfect Secret/Blinded Node Key Tree in Tree-based Extensions of Joux's Protocol

group, while U_8 broadcasts $bk_8 = k_8P$ as usual. Members $\{U_7, U_8\}$ then compute the node key k_{78} using Joux's protocol, as

$$k_{78} = e(k_7P, k'_7P)^{k_8} = e(k_8P, k'_7P)^{k_7} = e(P, P)^{k_7k'_7k_8},$$

and proceed with the pairing-based TGDH protocol as usual.

Authentication

Barua *et al.* propose both an unauthenticated and an authenticated version of the BDS-ID-AGKA protocol. As Joux's protocol does not provide authentication, the tree-based extension presented above is the unauthenticated version of the BDS-ID-AGKA protocol. In the authenticated version, the authors use the authenticated tripartite protocol of Zhang *et al.* from Section 4.1.5 in place of Joux's protocol.

5.1.4 Extension of Joux's Protocol : The LKKR-GKA Protocol

Lee *et al.* independently proposed an unauthenticated pairing-based TGDH protocol in [LKKR04], which we call the LKKR-GKA protocol.

Tree Construction

Lee *et al.* choose a simpler construction for their key trees, building a balanced ternary tree of height $h = \lfloor \log_3 n \rfloor$ and simply adding the remaining $n - h$ leaf nodes to the last row of the key tree, from left to right. For example, the LKKR-GKA key

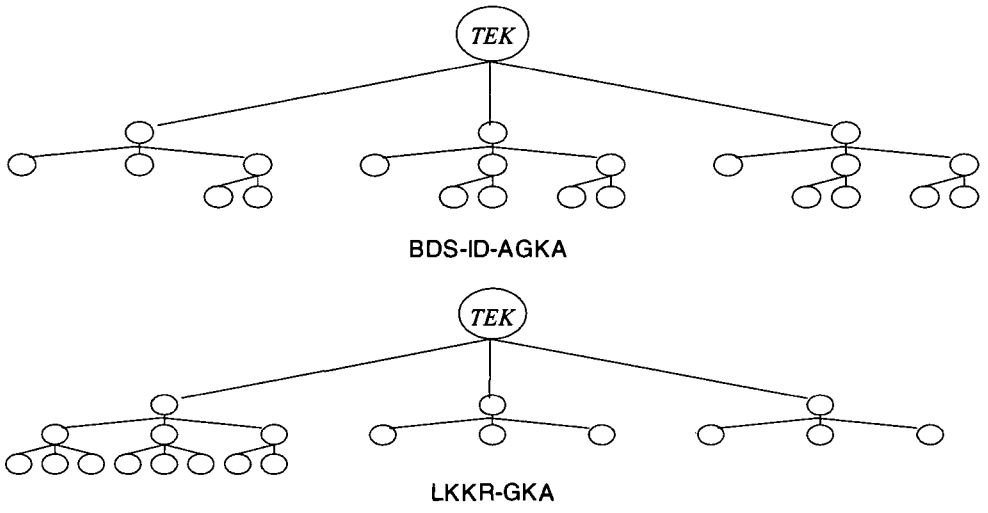

Figure 5.4: BDS-ID-AGKA and LKKR-GKA Key Trees for Group of Size $n = 14$.

tree for a group of size 8 is given in Figure 5.3, while the tree for a group of size 14 is compared with the BDS-ID-AGKA protocol in Figure 5.4.

Two-Party Node Key Agreement

The LKKR-GKA protocol improves upon the BDS-ID-AGKA protocol by requiring that the group members $\{U_7, U_8\}$ simply perform elliptic curve Diffie-Hellman (ECDH) key exchange to compute k_{78}. Members U_7 and U_8 broadcast their respective blinded keys, $bk_7 = k_7P$ and $bk_8 = k_8P$, and compute the node key k_{78} as

$$k_{78} = k_7(k_8P) = k_8(k_7P) = k_7k_8P.$$

This process necessitates the inclusion of an additional hash function $H_2 : \mathbb{G}_1 \rightarrow \mathbb{Z}_q^\times$ for the computation of the blinded key $bk_{78} = H_2(k_{78})P$. As with the BDS-ID-AGKA protocol, each member then proceeds with the pairing-based TGDH protocol as usual.

Authentication

The LKKR-GKA protocol proposed in [LKKR04] does not provide authentication. However, we note that an authenticated version can easily be obtained by substituting Joux's protocol for the authenticated protocol of Zhang *et al.* for three party node

key computation, as in the BDS-ID-AGKA protocol, and using some authenticated ECDH key exchange protocol, such as ECMQV, for two party node key computation.

5.1.5 Extension of Joux's Protocol : The SHL-ID-AGKA Protocol

In [SHL05], Shiau *et al.* proposed another TGDH-based extension of Joux's protocol, which we call the SHL-ID-AGKA protocol, that uses binary trees rather than ternary trees. In order to use Joux's protocol for node key computation, the protocol associates members of the multicast group with any type of node, and not just the leaf nodes as in the other pairing-based variations. However, we note that the protocol may also be implemented using a member-per-leaf ternary tree with a particular initial key tree construction algorithm. We first present the SHL-ID-AGKA protocol, as given by the authors, in the Overview section below, and then go on to show that the protocol in fact fits the same general framework as the other ternary-tree based extensions of Joux's protocol, despite initial appearances.

Overview

Each node i in the tree corresponds to a member U_i, that generates a secret key $k_i \in \mathbb{Z}_q^\times$ and blinded key $bk_i = k_i P \in \mathbb{G}_1^\times$. In addition, each non-root parent node in the tree is associated with a secret node key $K_i \in \mathbb{Z}_q^\times$ and blinded node key $BK_i \in \mathbb{G}_1^\times$. As usual, the root node is associated with the group TEK. Examples of the key trees can be found in Figure 5.5. Each parent node member performs Joux's tripartite key exchange with its two child node members to compute the secret node key associated with that parent node. In other words, each secret node key K_i is given by

$$K_i = e\left(bk_{L(i)}, bk_{R(i)}\right)^{k_i} = e(P,P)^{k_{L(i)}k_{R(i)}k_i}, \tag{5.1.3}$$

where $L(i)$ and $R(i)$ represent the left and right child nodes of the parent node i. When computing the next highest secret node key in the tree, the parent node member broadcasts the blinded node key BK_i rather than their personal blinded key bk_i.

A clear advantage of the SHL-ID-AGKA protocol is that its structure simplifies

group management policies for choosing distinguished members in key computations by requiring that the parent node members always assume the role of distinguished representatives. We further illustrate the SHL-ID-AGKA protocol with the following example.

Example

Suppose we have a group of 10 members, $\mathcal{U} = \{U_1, ..., U_{10}\}$ arranged as in Figure 5.5, that agree on the public system parameters $\langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, H_1 \rangle$, where $H_1 : \mathbb{G}_2 \rightarrow \mathbb{Z}_q^\times$. Each group member U_i chooses a value $k_i \in_R \mathbb{Z}_q^\times$. The members compute the secret key and blinded key trees from Figure 5.5 as follows:

1. Members $\{U_4, U_8, U_9, U_{10}\}$ compute and broadcast their personal blinded key $bk_i = k_i P$, $i \in \{4, 8, 9, 10\}$. Since member U_5 has only a single child node member (U_{10}), he chooses a second secret key $k_5' \in \mathbb{Z}_q^\times$ and computes and broadcasts the blinded keys $bk_5 = k_5 P$ and $bk_5' = k_5' P$.

2. Members $\{U_4, U_8, U_9\}$ compute the node key $K_4 = e(P, P)^{k_4 k_8 k_9}$, using Equation (5.1.3). Similarly, members $\{U_5, U_{10}\}$ compute the node key $K_5 = e(P, P)^{k_5 k_5' k_{10}}$. After computing the secret node keys, the parent node members $\{U_4, U_5\}$ compute and broadcast the blinded node keys, $BK_4 = H_1(K_4)P$ and $BK_5 = H_1(K_5)P$, respectively. At this point, members $\{U_2, U_3, U_6, U_7\}$ compute and broadcast their personal blinded key, $bk_i = k_i P$, $i \in \{2, 3, 6, 7\}$.

3. Using Equation (5.1.3), members $\{U_2, U_4, U_5, U_8, U_9, U_{10}\}$ compute the secret node key $K_2 = e(P, P)^{k_2 \cdot H_1(K_4) \cdot H_1(K_5)}$, while members $\{U_3, U_6, U_7\}$ compute the node key $K_3 = e(P, P)^{k_3 k_6 k_7}$. After computing the secret node keys, the parent node members $\{U_2, U_3\}$ compute and broadcast the blinded node keys $BK_2 = H_1(K_2)P$ and $BK_3 = H_1(K_3)P$, respectively. At this point, the root member U_1 computes and broadcasts his personal blinded key $bk_1 = k_1 P$.

4. Finally, all members compute the final group session key as

$$\begin{aligned} K &= e(P, P)^{k_1 \cdot H_1(K_2) \cdot H_1(K_3)} \\ &= e(P, P)^{k_1 \cdot H_1(e(P,P)^{k_2 H_1(K_4) H_1(K_5)}) \cdot H_1(e(P,P)^{k_3 k_6 k_7})} \end{aligned}$$

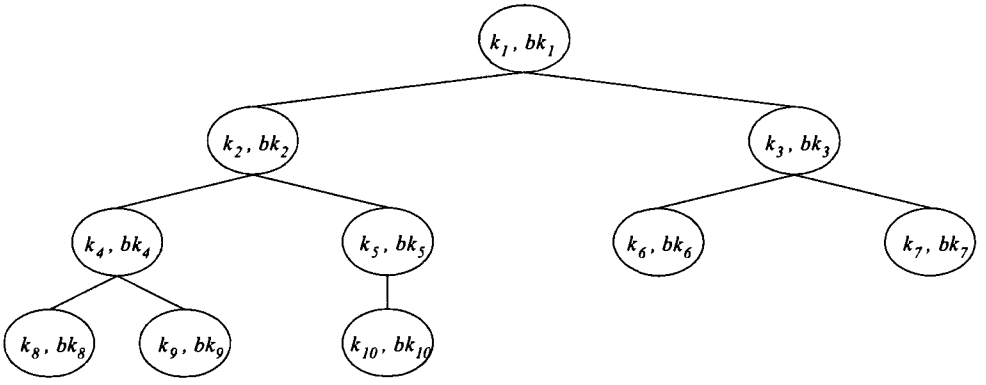

Figure 5.5: SHL-ID-AGKA Key Trees

$$= e(P,P)^{k_1 \cdot H_1(e(P,P)^{k_2 H_1(e(P,P)^{k_4 k_8 k_9}) H_1(e(P,P)^{k_5 k_5' k_{10}})) \cdot H_1(e(P,P)^{k_3 k_6 k_7})}.$$

Tree Construction

Although the SHL-ID-AGKA protocol was proposed using binary trees with members associated with nodes, we have found that the protocol can also be implemented in the exact same manner as the BDS-ID-AGKA and LKKR-GKA protocols, where members are associated with the leaves of a ternary tree that's constructed using some particular algorithm. To describe the tree construction algorithm, we outline how the binary key tree can be converted to a ternary key tree.

Given a binary tree, we construct the corresponding ternary tree as follows:

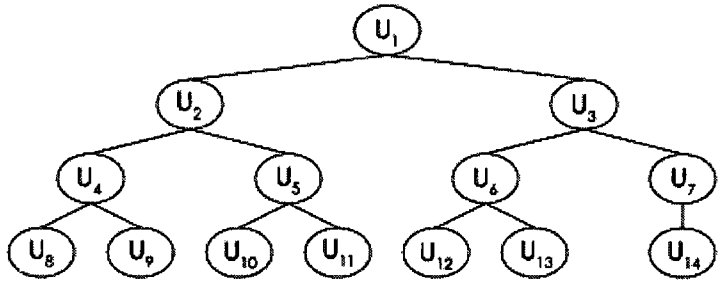

Figure 5.6: Binary Tree Representation of the SHL-ID-AGKA Protocol.

1. If the binary tree is empty, then so is the corresponding ternary tree.

2. If the binary tree consists of a single node, then so does the corresponding ternary tree.

3. If the binary tree consists of two nodes A and B, the corresponding ternary tree has a root node with leaf nodes A and B.

4. Otherwise, we recursively build the ternary tree from the root node down by replacing a parent node A with child subtrees B and C, with a root node that is the parent of the following child nodes, from right to left: the node A, the subtree B and the subtree C.

Whether we implement the SHL-ID-AGKA protocol with the binary tree construction or with the corresponding ternary tree construction described above, we obtain identical message flows between group members, producing the same session key. We illustrate the member-per-node binary tree construction in Figure 5.6 and our member-per-leaf ternary tree construction of the SHL-ID-AGKA protocol in Figure 5.7.

Two-Party Node Key Agreement

In the case of two-party node key agreement, Shiau *et al.* use the method of Barua *et al.* from [BDS03], where a distinguished member broadcasts two blinded key contributions and the two members use Joux's protocol to compute the node key. However, we note that the method of Lee *et al.* in [LKKR04] may also be used.

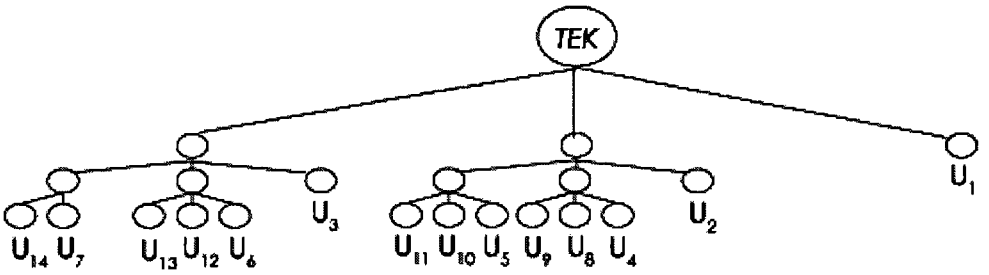

Figure 5.7: Ternary Tree Representation of the SHL-ID-AGKA Protocol

Authentication

Similar to [BDS03], the SHL-ID-AGKA protocol provides authentication by replacing Joux's protocol with the authenticated key exchange protocol of Zhang *et al.* from [ZLK02].

5.1.6 Analysis of Tree-based Protocols

We compare the original TGDH protocol with its pairing-based counterparts from [RN02, BDS03, LKKR04, SHL05]. To ease in our discussion, we drop the ID-AGKA and GKA suffixes from the protocol pseudonyms. We compare the various contributory and security properties of the protocols in Table 5.1.

Since we are working with both binary and ternary trees, we let $h_1 = \lceil \log_2 n \rceil$ be the height of the binary tree in the TGDH and RN protocols, $h_2 = \lfloor \log_2 n \rfloor$ be the height of the binary tree in the SHL protocol and $h_3 = \lceil \log_3 n \rceil$ be the height of the ternary tree in the BDS and LKKR protocols. We note that $h_2 = h_1 - 1$, unless $n = 2^\ell$ for some $\ell \in \mathbb{N}$ and in general, $h_3 \leq h_2 \leq h_1$. Since the number of rounds required in each tree-based protocol and the number of secret node keys stored in a member's key path are equivalent to the height of the tree, the ternary structure of the BDS and LKKR protocols offer reduced round complexity and key storage in comparison to the other binary tree-based protocols. Although the SHL protocol reduces the length of the key path of members near the root of the key tree, the storage cost of the average member is still higher than that of the BDS and LKKR protocols.

An advantage of tree-based protocols is that they allow for efficient dynamic membership. Each of the tree-based protocols propose algorithms for auxiliary key

agreement (AKA) for join, leave, merge or partition operations. Members are required to store the keys from their key path for use in the AKA protocols. In the AKA protocols, some members will have to update the node keys in their key path to ensure key independence. Determining these specific members depends on the specific protocol's policy for adding and removing members from the key tree. However, after a certain number of membership changes, we can be assured that every member will have had to update keys in their key path. Thus, although members are required to store the keys in their key path for future use, these keys can still be regarded as ephemeral values and thus the tree-based protocols maintain perfect forward secrecy when the auxiliary key agreement protocols are applied. We note that the RN protocol is the only exception, as we showed that it only provides partial forward secrecy in Section 5.1.1.

In a tree-based protocol, each member only directly interacts with members in their own subtree. As a result, they must rely on the members of other subgroups to verify the contributory key material of other members and correctly compute the intermediate node keys. Since a member cannot personally verify that the final session key was computed using the contribution of each other member, the tree-based GKA protocols cannot provide the properties of verifiable contributiveness or key integrity, as defined in Section 2.2.1.

	TGDH	BDS	LKKR	SHL	RN
Rounds (Max)	h_1	h_3	h_3	h_2	h_1
Authentication	–	✓	–	✓	✓
AKA Protocols	✓	✓	✓	✓	✓
Perfect Forward Secrecy	✓	✓	✓	✓	×
Verifiable Contributiveness	×	×	×	×	×
Key Integrity	×	×	×	×	×

Table 5.1. Properties of tree-based GKA protocols.

Computational Costs

Since the computational costs are dominated by the expensive pairing operation, and may vary from one member to another depending on their height in the key tree, we compare the computational costs of the pairing-based TGDH protocols in terms of the

total number of pairings $\mathcal{P}$ computed by the group, and thus omit the original TGDH protocol from our discussion. As the authors of the pairing-based TGDH protocols do not provide the computational costs associated with initial key agreement, we consider these costs in our comparison rather than the costs of AKA protocols, which can be found in the original papers. Consequently, the pairing costs given in Table 5.2 are of our own calculation and were determined inductively. To illustrate our comparison, we provide a graphical representation of the total number of pairings required for initial key agreement in groups of size $1 \leq n \leq 10000$ in Figure 5.8.

Recall that in the case that a given node has only two child nodes, the ternary tree-based extensions of Joux's protocol, including the SHL protocol, use some two party key agreement protocol to allow the two remaining members to establish a secret node key. The initial key tree construction of the LKKR and SHL protocols ensure that we must only handle this case for a single pair of members and only when n is even. To account for this in our computational costs, we define the following function.

$$\delta_n = \begin{cases} 1 & \text{if } n \text{ is even.} \\ 0 & \text{if } n \text{ is odd.} \end{cases}$$

However, the initial key tree construction of the BDS protocol requires that members are distributed evenly in the last level of the key tree, which increases the frequency of the case given above. As a result, we define the following function for the BDS protocol:

$$\gamma_n = \begin{cases} 1 & \text{if } 3^{h_3-1} < n \leq 2 \cdot 3^{h_3-1}. \\ 0 & \text{if } 2 \cdot 3^{h_3-1} < n \leq 3^{h_3}. \end{cases}$$

We compare the computational costs of the pairing-based TGDH protocols in Table 5.2 below.

Protocol	Pairing Computations $\mathcal{P}$ (Total)
BDS-ID-AGKA	$nh_3 + \gamma_n \left(n - 2 \cdot 3^{h_3-1}\right)$
LKKR-GKA	$nh_3 + \frac{n-3^{h_3}}{2} - \frac{3}{2}\delta_n$
SHL-ID-AGKA	$h_2(n+1) - 2^{h_2+1} + \frac{n+3}{2} + \frac{1}{2}\delta_n$
RN-ID-AGKA	$2n(h_1+1) - 2^{h_1+1}$

Table 5.2. Computational costs of tree-based GKA protocols

Communication Costs

We compare the communication costs of the various tree-based protocols in Table 5.3 in terms of the total number of unicast messages and the total number of broadcast messages required for initial key agreement. For identity-based protocols, we do not include the unicast messages associated with private key distribution, as this may be done offline. However, since the PKG has an active role in the RN protocol, we include the unicast messages sent for non-member private key distribution (i.e. the private keys d_{12} and d_{34} in Figure 5.1). To account for the case of two-party node key agreement in our costs, we use the functions δ_n and γ_n defined above. In Figure 5.8, we graph the number of broadcast messages required for the initial key agreement stage of each of the pairing-based TGDH protocols, for groups of size $1 \leq n \leq 10000$.

Protocol	Messages (Total)	
	Unicast	Broadcast
TGDH	–	$2(n-1)$
BDS-ID-AGKA	–	$\frac{3^{h_3}-3}{2} + n + \gamma_n \left(n - 2 \cdot 3^{h_3-1}\right)$
LKKR-GKA	–	$\frac{3n-3}{2} + \frac{1}{2}\delta_n$
SHL-ID-AGKA	–	$\frac{3n-3}{2} + \frac{1}{2}\delta_n$
RN-ID-AGKA	$nh_1 - 2^{h_1}$	$2(n-1)$

Table 5.3. Communication costs of tree-based GKA protocols

Conclusions

Based on the costs given in Tables 5.2 and 5.3 and illustrated in Figure 5.8, we can accurately compare the different approaches of the pairing-based TGDH protocols. We first address the issue of the different key exchange protocols used in place of the DH key exchange. As discussed in Section 5.1.1, the tree-based extension of Smart's protocol allows a group of members to perform authenticated group key agreement in an efficient manner, but requires the PKG to perform a more extensive role than in other ID-based public key infrastructures and play an active part in the key agreement process. Unless the role of the PKG can be minimized, tree-based extensions of Smart's protocol are not practical for group key agreement. The tree-based extensions of Joux's protocol do not suffer from this drawback and are a more favourable option. Amongst the various tree-based extensions of Joux's protocol, we

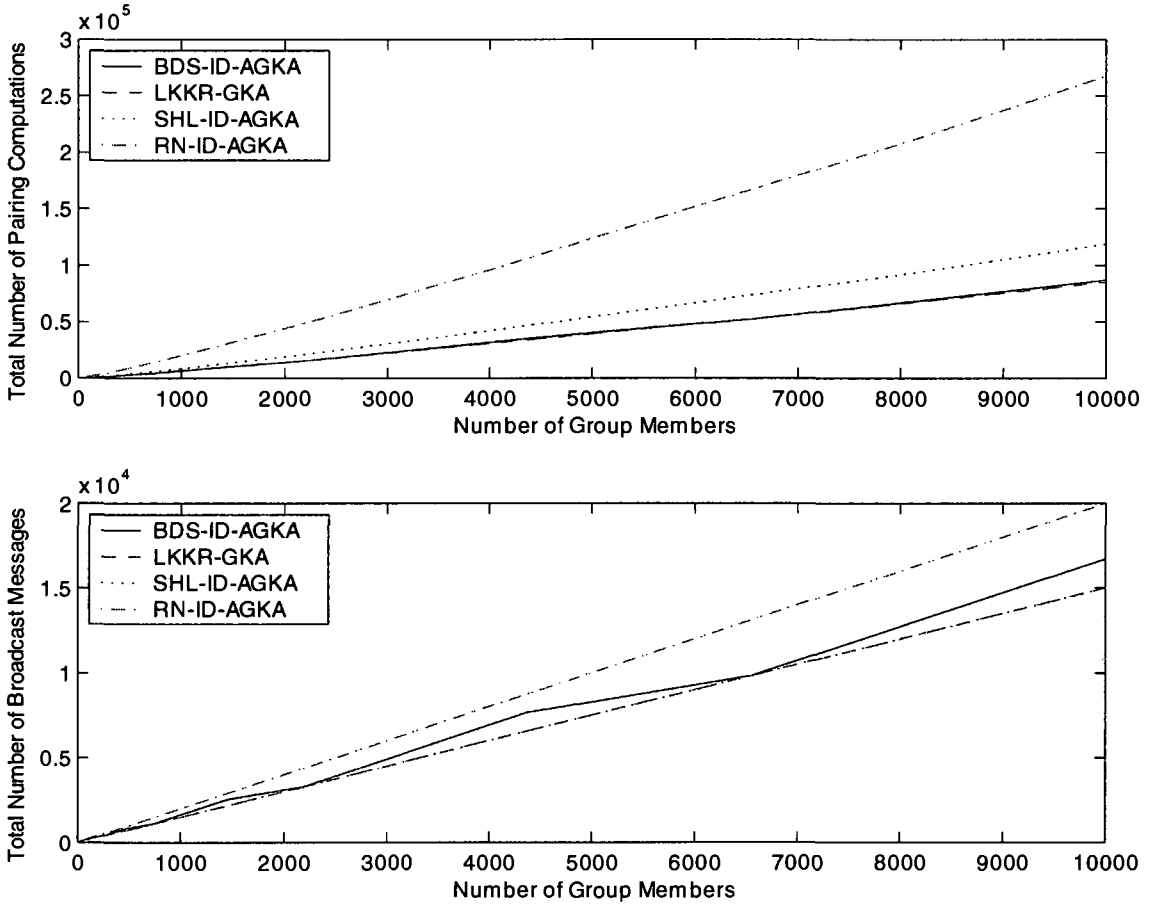

Figure 5.8: Comparison of the Computational and Communication Costs of the Tree-based Protocols

must compare the different approaches of tree construction, two party key agreement and authentication.

As shown in Figure 5.8, the complicated tree construction algorithms of the BDS and SHL protocols do not provide any computational or communication advantages over the simplistic tree construction algorithm of the LKKR protocol. While the SHL protocol offers reduced computational costs for members close to the root of the tree and reduced communication costs for the average member, the tree structure increases the average computational costs. Conversely, while the structure of the BDS protocol offers reduced computational costs for the average member, the structure increases the average number of broadcast messages. The LKKR protocol provides computational costs comparable to the BDS protocol and communication costs identical to the SHL protocol and thus provides the most favourable tree construction algorithm.

Using Joux's protocol for two-party node key agreement, the BDS and SHL protocols require a total computational cost of $2(\mathcal{P}+\mathcal{M}+\mathcal{E})+\mathcal{M}$ for the two members, where $\mathcal{M}$ refers to scalar multiplication in $\mathbb{G}_1$ and $\mathcal{E}$ refers to exponentiation in $\mathbb{G}_2$, with the extra multiplication resulting from the additional blinded key computation of the distinguished member. Using the ECDH key exchange protocol, the LKKR protocol reduces the cost to only $4\mathcal{M}$. While the protocols induce the same number of broadcasts, the BDS and SHL protocols require the distinguished member to broadcast a message that's twice the size, increasing the bandwidth. As noted in the computation section above, the tree structure of the BDS protocol actually increases the frequency of the two-party node key agreement cases. Thus the LKKR protocol handles two-party node key agreement more efficiently at the expense of requiring members to keep track of two different key exchange protocols.

Finally, we note that the computational costs given in Table 5.2 do not include the costs of authentication for the tree-based extensions of Joux's protocol. Each pairing computation given in Table 5.2 is the result of a node key calculation and thus corresponds to an execution of Joux's protocol. By replacing Joux's protocol with the authenticated version of Zhang *et al.*, we obtain computational costs proportional to those given in Table 5.2. We reiterate that, although the LKKR protocol does not specify the use of a specific signature scheme, authentication may be provided in the same manner as the BDS and SHL protocols, using Zhang *et al.*'s protocol along

with some authenticated version of the ECDH protocol. Thus the various tree-based extensions of Joux's protocol are quite similar in terms of authentication.

In conclusion, we propose that the LKKR protocol is the most efficient of the pairing-based TGDH protocols considered. The LKKR protocol's simplified tree construction algorithm and inexpensive ECDH key exchange protocol allow a group of members to agree upon a session key in the fewest number of rounds, while inducing the fewest pairing computations and broadcast messages. As a final remark, we note that the most computationally efficient tree-based protocol would be obtained by coupling the tree structure of the BDS protocol, which increases the frequency of two-party node key agreement, with the ECDH scheme of the LKKR protocol. However, this protocol would still incur the high communication costs of the BDS protocol.

5.2 Burmester-Desmedt-based Group Key Agreement

The Burmester-Desmedt (BD) protocol described in Section 2.3.5, is an efficient 2-round GKA protocol based on the DH key exchange protocol. Pairing-based BD protocols were proposed in [DWGW03a, DWGW03b, CHL04, HLH07]. Each protocol adopts the following initial structure of an EC-based BD protocol. For clarity, we address the issue of authentication for each protocol individually.

Round 1 Each member $U_i \in \mathcal{U}$ chooses an ephemeral private key $k_i \in_R \mathbb{Z}_q^\times$ and broadcasts the ephemeral public key $P_i = k_i P$.

Round 2 Each member U_i uses his ephemeral private key k_i and some small subset Σ_i of the public keys $\{P_1, ..., P_{i-1}, P_{i+1}, ..., P_n\}$ to compute a value X_i. Each member broadcasts the value X_i.

Key Computation Each member U_i computes the shared session key as

$$\begin{aligned} K &= Z_i^{nk_i} \cdot \prod_{j=0}^{n-1} X_{i+j \bmod n}^{n-j-1 \bmod n} \\ &= Z_i^{nk_i} \cdot X_i^{n-1} \cdot X_{i+1}^{n-2} \cdot \ldots \cdot X_{i-2}, \end{aligned}$$

where Z_i is computed as a function of Σ_i by each U_i.

While the pairing-based BD protocols follow the general framework given above, each individual protocol uses pairings in some unique way, either in the computation of the values X_i and Z_i or in the authentication process. Consequently, we present each protocol in terms of key material and session key computation, the manner in which they provide authentication and vulnerability to attacks. In Section 5.2.4, we compare the efficiency of each protocol with the original BD protocol.

5.2.1 The DWGW-ID-AGKA Protocol

The first pairing-based BD protocol was proposed by Du *et al.* in [DWGW03a], which we call the DWGW-ID-AGKA protocol. Their variation of the EC-based BD protocol uses pairings in both key computation and authentication.

Key Computation

The protocol uses pairings in the computation of the key material X_i in the second round and the value Z_i in the key computation stage. Member U_i uses his ephemeral private key k_i and the subset of public keys $\Sigma_i = \{P_{i-1}, P_{i+1}\}$ to compute

$$X_i = e(P_{pub}, k_i(P_{i+1} - P_{i-1})).$$

In the final stage, each member U_i computes $Z_i = e(P_{pub}, P_{i-1})$ so that

$$K = e(P, P)^{(k_1k_2+k_2k_3+\ldots+k_nk_1)s}.$$

For a proof of the consistency of the protocol, we refer the reader to [DWGW03a]. We note the requirement of strong non-degeneracy for our pairing.

Authentication

The DWGW-ID-AGKA protocol provides authentication using Hess' ID-based signature scheme discussed in Section 4.1.4. Each member U_i is required to sign their key contribution P_i and screen the signatures of all other members. As described in Section 4.2, signature screening is a weaker form of batch verification and does

not guarantee that each individual signature will pass the verification stage. Furthermore, we note that the protocol does not require members to sign the publicly broadcasted value X_i from the second round.

Attacks

In [ZC03], Zhang and Chen outlined an insider impersonation attack on the DWGW-ID-AGKA protocol, whereby members U_{i-1} and U_{i+1} can collaborate and impersonate member U_i, given previous authentication transcripts for U_i. Indeed, given an authenticated value P_i, members U_{i-1} and U_{i+1} can collude to compute

$$\begin{aligned} X_i &= e(P_i, (k_{i+1} - k_{i-1})\, P_{pub}) \\ &= e(k_i P, s(P_{i+1} - P_{i-1})) \\ &= e(P_{pub}, k_i(P_{i+1} - P_{i-1})). \end{aligned}$$

In response to this attack, Du *et al.* improved their protocol in [DWGW03b] using a synchronous counter c to update the long-term private key $d_i' = cd_i$ for each session. This ensures that the signature on P_i is unique to each session. Zhang and Chen also suggested that the protocol could be repaired by incorporating time stamps into the signature process. However, in [TM04], Tang and Mitchell showed that the improved version of the DWGW-ID-AGKA protocol and the repaired protocol suggested in [ZC03] are still vulnerable to insider impersonation attacks. They note that the impersonation attacks are due to the lack of direct authentication of the value X_i and that the attacks can be prevented by requiring members to authenticate these values. Using Hess' signature scheme with the screening approach, each member would have to compute an additional $3\mathcal{M}+1\mathcal{H}$ for signature generation and $2\mathcal{P}+(n-1)\,(\mathcal{M}+\mathcal{H})$ for signature verification, for a total of $2\mathcal{P} + (n+2)\mathcal{M} + n\mathcal{H}$ computations.

5.2.2 The CHL-ID-AGKA Protocol

In [CHL04], Choi *et al.* independently proposed a pairing-based BD protocol, which we call the CHL-ID-AGKA protocol. Similar to the DWGW-ID-AGKA protocol, the authors use pairings in key computation and authentication stages.

Key Computation

In contrast to the DWGW-ID-AGKA protocol, Choi *et al.* use the bilinear nature of the pairing to extend the size of the public key subset for each member from 2 keys to 3 keys. Using the private key k_i and the public key subset $\Sigma_i = \{P_{i-1}, P_{i+1}, P_{i+2}\}$, member U_i computes

$$X_i = e(P_{i+1}, k_i(P_{i+2} - P_{i-1})).$$

While this is a more natural construction, we note that it provides no advantage over the construction of the DWGW-ID-AGKA protocol. In the final stage, each member U_i uses the value $Z_i = e(P_{i-1}, P_{i+1})$ to compute the group TEK as

$$K = e(P, P)^{k_1k_2k_3 + k_2k_3k_4 + \ldots + k_nk_1k_2}.$$

Authentication

Similar to the DWGW-ID-AGKA protocol, the CHL-ID-AGKA protocol uses a variation of Hess' ID-based signature scheme from Section 4.1.4 with signature screening to provide authentication and only requires each member U_i to sign their first round key contribution P_i. To reduce computational costs, each U_i is only required to screen the signatures of the contributions in their public key subset $\Sigma_i = \{P_{i-1}, P_{i+1}, P_{i+2}\}$ and rely on other members to properly verify their respective key subsets.

Attacks

The CHL-ID-AGKA protocol is also vulnerable to the insider impersonation attack of the DWGW-ID-AGKA protocol, as shown by Zhang and Chen in [ZC03]. Given previously authenticated values P_i and P_{i+1} for respective users U_i and U_{i+1}, members U_{i-1} and U_{i+2} can collude to impersonate U_i by computing

$$\begin{aligned} X_i &= e(P_{i+1}, P_i)^{k_{i+2} - k_{i-1}} \\ &= e(P_{i+1}, k_i(k_{i+2} - k_{i-1})P) \\ &= e(P_{i+1}, k_i(P_{i+2} - P_{i-1})). \end{aligned}$$

Similar to the discussion in Section 5.2.1, Tang and Mitchell claim that the se-

curity vulnerability of the CHL-ID-AGKA protocol is due to the lack of authentication of the X_i values and the repair proposed by Zhang and Chen in [ZC03] is still vulnerable to attacks. In [Shi07], Shim showed that three users can collude to impersonate a single user without the use of authentication transcripts and proposed a modified version, secure against all previous insider impersonation attacks, in which all members sign and verify the P_i and X_i values. Modified versions of the CHL-ID-AGKA protocol, secure against the attacks of [ZC03, TM05, Shi07] were proposed in [YHVK08, PHYK08]. However, in [CHL08] Choi *et al.*, the authors of the original CHL-ID-AGKA protocol, showed that the modified protocol of Shim from [Shi07] is still vulnerable to another form of impersonation attack. They proposed a definitive version, in which all members sign their P_i contribution, along with the set of member identities, and sign their X_i value, along with the set of contributions $\{P_1, ..., P_n\}$ and the set of member identities, that is secure, "provided that at least one member of the group is honest".

5.2.3 The HLH-ID-AGKA Protocol

In response to the insider impersonation attacks on the previous protocols, He *et al.* proposed an insider attack resistant protocol in [HLH07], which we call the HLH-ID-AGKA protocol. In contrast to the previous two protocols, the HLH-ID-AGKA protocol does not use pairings in the key computation stages. However, the protocol does use pairings for a special verification stage and for identity-based authentication purposes.

Key Computation

Since pairings are not used in the key computation, the HLH-ID-AGKA protocol is essentially an elliptic curve variant of the BD protocol. Using the private key k_i and the public key subset $\Sigma_i = \{P_{i-1}, P_{i+1}\}$, each member U_i computes

$$X_i = k_i(P_{i+1} - P_{i-1}),$$

in the second round and the group TEK as

$$\begin{aligned} K &= nk_iP_{i-1} \cdot \sum_{j=0}^{n-1}(n-j-1 \bmod n)X_{i+j \bmod n} \\ &= nk_iP_{i-1} \cdot (n-1)\,X_i \cdot (n-2)\,X_{i+1} \cdot \ldots \cdot X_{i-2} \\ &= (k_1k_2 + \ldots + k_nk_1)\,P, \end{aligned}$$

so that $Z_i = P_{i-1}$.

Recall that the use of pairings allowed malicious members to impersonate other members in the DWGW-ID-AGKA and CHL-ID-AGKA protocols. The HLH-ID-AGKA protocol takes advantage of the bilinearity property to allow members to verify the X_i contributions of each member, by checking that

$$e(X_i, P) = e(P_{i+1} - P_{i-1}, P_i)$$

for every X_i, $1 \leq i \leq n$.

Authentication

To provide authentication, the HLH-ID-AGKA protocol uses a variant of the ID-based aggregate signature scheme of Cheon *et al.* of [YCK04], which is itself a variation of the ID-based signature scheme of Cha and Cheon from Section 4.1.3. The variant requires each member U_i to incorporate their public/private ephemeral key material *and* the entire public key set $\Sigma = \{P_1, \ldots, P_n\}$ in their signature, rather than signing their personal public key P_i alone. However, as they are not required to sign P_i in the first round, the members are essentially signing the unauthenticated key contributions of all the other members, which may make the protocols vulnerable to some attack. Similar to the DWGW-ID-AGKA and CHL-ID-AGKA protocols, the HLH-ID-AGKA protocol uses the weaker notion of signature screening and does not require members to authenticate the X_i values.

Attacks

By requiring each member to sign the entire public key set, the signature is dependent on the public contributions of every member in the session. Thus colluding members

cannot use previous authentication transcripts to impersonate a member in a new session unless the public key set of the current session is identical to that of some previous session. As a result, the HLH-ID-AGKA protocol is protected against the impersonation attack of [ZC03]. However, we note that the authors do not provide a formal proof of the security of their protocol.

5.2.4 Analysis of BD-based Protocols

We compare the pairing-based BD protocols from [DWGW03a, CHL04, HLH07] with the original BD protocol from [BD94] in terms of group key properties and communication costs in Table 5.4. As in the analysis of Section 5.1, we drop the ID-AGKA suffix from our protocol pseudonyms.

Since the pairing-based BD protocols closely follow the structure of the original BD protocol, each requires a total of $2n$ broadcast messages sent over 2 broadcast rounds. To be comprehensive, we include the values in Table 5.4, although a comparison is not very interesting.

We first note, that the EC-based BD protocol reduces the bitsize of broadcast messages over the original field-based BD protocol. Since the pairing-based BD protocols follow the same EC-based framework, they offer the same advantage over the BD protocol. We note that the use of pairings in key computation offers no direct advantage over the EC-based BD protocol and actually increases the computational costs. The benefits of the pairing-based protocols are brought about by the use of an ID-PKI in the authentication process, which offers several advantages over the traditional certificate-based authentication of the BD protocol, as discussed in Section 3.3. However, as discussed in the sections above, each protocol employs the weaker notion of signature screening, rather than strong batch verification, in the verification stage and thus does not provide proper authentication of every member contribution, denoted by the symbol $*$ in Table 5.4. As an example of their scalable authenticated key establishment compiler, Katz and Yung proposed an authenticated version of the BD protocol which employs some unspecified signature scheme in [KY03]. We denote the fact that the protocol does not recommend the use of a specific signature scheme by the symbol $\Diamond$ in Table 5.4.

As mentioned in Section 2.3.5, the BD protocol does not provide provide auxiliary

key agreement protocols and requires the initial key agreement protocol to be executed following a membership change. The pairing-based BD protocols suffer a similar fate and are thus inefficient for dynamic groups. Since each session key is uniquely determined by the ephemeral secrets of the members in the session, each protocol inherently provides perfect forward secrecy. However, it is not clear that the BD-based protocols would maintain perfect forward secrecy if auxiliary key agreement schemes were proposed, since this is a difficult problem, as we shall see in Chapter 7.

Loosely speaking, each BD-based protocol provides contributory group key agreement, since each member uses the key material X_i of every other member in the computation of the group TEK. The HLH protocol has the added advantage of allowing members to confirm that each other member indeed computed X_i correctly. However, since these values are not authenticated in any of the protocols, they cannot be authentically linked to their respective members and thus the protocols do not provide verifiable contributiveness. As a result, the protocol does not provide the stronger notion of key integrity.

Finally, as discussed throughout this section, the protocols from [DWGW03a, DWGW03b, CHL04] were shown to be vulnerable to insider impersonation attacks in [ZC03, TM04], which do not affect the protocol from [HLH07].

		BD	DWGW	CHL	HLH
Rounds		2	2	2	2
Authentication		✓$^{\diamond}$	✓*	✓*	✓*
AKA Protocols		×	×	×	×
Perfect Forward Secrecy		✓	✓	✓	✓
Verifiable Contributiveness		×	×	×	×
Key Integrity		×	×	×	×
Known Attacks		[TM05]	[ZC03, TM04]	[ZC03, TM04, Shi07]	–
Messages (Total)	Unicast	–	–	–	–
	Broadcast	$2n$	$2n$	$2n$	$2n$

Table 5.4. Communication costs and properties of the pairing-based BD protocols.

Computational Costs

We compare the computational costs of the various pairing-based BD protocols in Table 5.5 in terms of the number of pairing operations $\mathcal{P}$ and scalar multiplications $\mathcal{M}$ in the group $\mathbb{G}_1$. As the costs are the same for each member of the group, we give the operations for a single member. However, we also provide the total number of computations for the group, to allow comparison with other protocols. Since each protocol employs a different scheme for authentication, we consider the costs of key agreement and authentication separately. For each entry in the table, we write the costs as $a \ / \ b$, where a refers to the number of operations in the key agreement process and b refers to the number of operations required to provide authentication, including signature generation and verification.

Protocol	Computations (GKA / Authentication)	
	$\mathcal{P}$	$\mathcal{M}$
DWGW-ID-AGKA	$2 \ / \ 2$	$2 \ / \ n+1$
(Total)	$4n$	n^2+3n
CHL-ID-AGKA	$2 \ / \ 2$	$2 \ / \ 5$
(Total)	$4n$	$7n$
HLH-ID-AGKA	$2(n-1) \ / \ 2$	$n+1 \ / \ 3$
(Total)	$2n^2$	n^2+4n

Table 5.5. Computational costs of the pairing-based BD protocols.

Conclusions

From Tables 5.4 and 5.5, we can compare the different approaches of the various pairing-based BD protocols. In the DWGW and CHL protocols, the pairing provides a simple and elegant method for combining multiple contributions in the computation of the X_i values. However, this approach provides no advantage over the inexpensive elliptic curve BD computations of the HLH protocol and in fact only increases the computational costs.

Each protocol employs the weaker notion of screening in the verification of member contributions and thus does not provide proper authentication of key contributory material. As a result, we cannot obtain the nice contributory properties of verifiable contributiveness or key integrity. However, we note that this authentication issue

can be fixed by using the method of strong batch verification discussed in Section 4.2. If one must use a signature screening approach, we prefer the method of CHL protocol as it offers reduced computational costs in comparison to the DWGW and HLH protocols.

As mentioned in Sections 5.2.1 and 5.2.2, the lack of direct authentication of the X_i values makes the DWGW and CHL protocols vulnerable to insider impersonation attacks. While the HLH protocol is resistant to the attack, the security of the protocol is not formally proven. In addition, the HLH protocol requires each member U_i to verify the proper computation of the second round contribution X_j of every other member U_j, inducing an impractical number of pairing computations for each member. The DWGW and CHL protocols can be protected against the insider impersonation attacks by authenticating the X_i values along with the identity and contribution sets, which is more efficient than the expensive verification procedure of the HLH protocol.

Unfortunately, no clear winner emerges from our comparison of the pairing-based BD protocols as all of the protocols are flawed in one way or another. We propose that the CHL protocol is the most efficient of the pairing-based BD protocol, as they have been presented in this section. The protocol is the least expensive in terms of computational costs and provides the most efficient form of signature screening. Although vulnerable to the insider attack, the protocol is still secure against outsider attacks. However, we note that there is plenty of room for improvement in the pairing-based BD class of group key agreement protocols.

5.3 Computationally Asymmetric Conference Key Agreement

The Conference Key Agreement (CKA) protocol, discussed in Section 2.3.6, is an efficient computationally asymmetric GKA protocol in which a group of responders $\mathcal{R}$ broadcast their key contributions in the clear while a distinguished group leader U_0 sends her contribution in some confidential and authenticated manner. Often the leader will mask her contribution for each responder using some unique value that can only be computed by the leader and the particular responder. While the term CKA is often used interchangably with GKA, we slightly abuse terminology here and

will refer to computationally asymmetric GKA protocols as CKA-based protocols.

Recall from Section 2.3.6, that several variations of the CKA protocol based on the DH key exchange have been proposed. Similarly, several pairing-based variations of the CKA protocol and the DH-based CKA protocols were proposed in [KNKW05, ZSM06, CSCW07, HLL07].

Suppose we have a group of $n+1$ members $\mathcal{U} = \{U_0, U_1, ..., U_n\}$, consisting of a group leader U_0 and a group of responders $\mathcal{R} = \{U_1, ..., U_n\}$. Each CKA-based protocol adopts the following structure:

Round 1 Each responder $U_i \in \mathcal{R}$ chooses an ephemeral private key $k_i \in_R \mathbb{Z}_q^\times$ and computes the ephemeral public key $P_i = k_i P$ and the signature σ_i authenticating P_i. Each responder broadcasts the pair (P_i, σ_i).

Round 2 The group leader U_0 verifies the signatures of all group members. If the verification holds, U_0 chooses some secret key contribution X and masks the contribution for each individual responder U_i using some shared masking value α_i to produce the set $Y = \{y_1, ..., y_n\}$. By shared masking value, we mean that the value α_i can only be computed by U_0 and U_i. The leader broadcasts Y along with some set of ephemeral public keys and a signature on the contribution X.

Key Computation Each responder $U_i \in \mathcal{R}$ extracts and unmasks the value y_i to recover the leader's secret key contribution X. After verifying the authenticity of the leader's contribution, each member computes the shared session key as

$$K = H(X \parallel Y)$$

using some cryptographic hash function H.

In the pairing- and ID-based CKA protocols from [KNKW05, ZSM06, CSCW07, HLL07], the group leader often uses pairings in some unique way to mask her secret key contribution X or to provide authentication of key contributory material. Consequently, we present each protocol in terms of the masking process and the manner in which they provide authentication of member contributions.

5.3.1 The KNKW-ID-AGKA Protocol

To our knowledge, the first ID-based CKA protocol was proposed by Kim *et al.* in [KNKW05], which we call the KNKW-ID-AGKA protocol. We note that the other three authors published one of the original DH-based CKA protocols in [NLKW05]. The KNKW-ID-AGKA protocol uses pairings to achieve ID-based authentication and in the computation of the contribution mask shared between the leader and each responder. We describe these particular aspects of the protocol in more detail.

Masking Process

In the second round of the KNKW-ID-AGKA protocol, the leader chooses two ephemeral private values $k_0, r \in \mathbb{Z}_q^\times$ and computes the ephemeral public keys $P_0 = k_0 P$ and $P_r = rP$. The leader chooses a secret key contribution $X \in \mathbb{G}_2$ and masks her contribution for each responder U_i using the shared value

$$\alpha_i = e(P_0, P_i)^r = e(P, P)^{rk_0k_1} = e(P_0, P_r)^{k_i}.$$

The leader computes each masked value as $y_i = X \cdot \alpha_i$ to obtain the set $Y = \{y_1, ..., y_n\}$.

To unmask the value y_i and recover the leader's contribution, each responder U_i computes $y_i \cdot \alpha_i^{-1} = X$. Each member computes the group TEK as $K = H(X, Y)$, using the hash function $H : (\mathbb{G}_2)^{n+1} \rightarrow \{0,1\}^*$.

Authentication

The KNKW-ID-AGKA protocol provides authentication by means of the ID-based aggregate signature scheme of Cheon *et al.* from [CKY04]. The protocol requires the leader to screen the signed contributions of the responders. The authors suggest that the leader's secret contribution may be signed and subsequently verified by the responders, although they do not describe an explicit signature scheme to be used.

We note that the signature scheme of [CKY04] uses signature screening rather than strong batch verification, as discussed in Section 4.2, and as a result, the KNKW-ID-AGKA protocol cannot guarantee the authentication of all member contributions.

5.3.2 The CSCW-ID-AGKA Protocol

Shortly after publication of the KNKW-ID-AGKA protocol, one of the authors (Dongho Won) published a similar protocol along with Cho *et al.* in [CSCW07]. The CSCW-ID-AGKA protocol, as we've called it, uses pairings in a similar manner, but employs completely different masking and authentication schemes.

Masking Process

In the second round of the CSCW-ID-AGKA protocol, the group leader chooses a single ephemeral private value $k_0 \in \mathbb{Z}_q^\times$ and computes the ephemeral public key $P_0 = k_0 P$. In addition, the leader broadcasts a string $\delta \in_R \{0,1\}^*$ and chooses a secret key contribution $X \in_R \{0,1\}^*$. To mask her contribution X for each responder U_i, the leader computes the shared value

$$\alpha_i = e(k_0 P_{pub}, P_i) = e(P, P)^{sk_0 k_1} = e(P_0, P_{pub})^{k_0}$$

and computes the masked value as $y_i = X \oplus H_1(\alpha_i, \delta)$, using the hash function $H_1 : \mathbb{G}_2 \times \{0,1\}^* \rightarrow \{0,1\}^*$, to generate the set $Y = \{y_1, ..., y_n\}$.

To unmask the value y_i and recover the leader's contribution, each responder U_i computes $y_i \oplus H_1(\alpha_i, \delta) = X$. Each member computes the group TEK as $K = H_2(X \parallel Y)$, using the hash function $H_2 : \{0,1\}^* \rightarrow \{0,1\}^*$.

Authentication

To provide authentication, the CSCW-ID-AGKA protocol uses a variant of Hess' ID-based signature scheme, discussed in Section 4.1.4. Similar to the KNKW-ID-AGKA protocol, the group leader verifies the signatures of all responders and requires the responders to authenticate the leader's contribution using some unspecified signature scheme. In addition, the protocol employs signature screening rather than strong batch verification and thus cannot guarantee the authenticity of responder contributions.

5.3.3 The HLL-GKA Protocol

In [HLL07], Hu *et al.* proposed an elliptic curve variation of the DH-based CKA protocol proposed in [NLKW05], which we call the HLL-GKA. While the protocol only employs pairings to verify the successful transmission of the leader's contribution, it is still worth mentioning due to the similarities to previous protocols.

Masking Process

Similar to the KNKW-ID-AGKA protocol, the group leader chooses two ephemeral private values $k_0, r \in \mathbb{Z}_q^\times$ and computes the ephemeral public keys $P_0 = k_0 P$ and $P_r = rP$. In contrast to the previous protocols, the leader computes her secret contribution as a product of the public keys of the responders and her ephemeral private key r, as $X = \sum_{i=0}^{n} rP_i$. To mask the secret contribution X, the leader computes the shared value

$$\alpha_i = rP_i = rk_iP = k_iP_r.$$

The leader computes each masked value as $y_i = X - H_1(\alpha_i)$, using the hash function $H_1 : \mathbb{G}_1^\times \rightarrow \mathbb{G}_1^\times$, to generate the set $Y = \{y_1, ..., y_n\}$.

To unmask the value y_i and recover the leader's contribution, each responder computes $y_i + H_1(\alpha_i) = X$. In addition, the HLL-GKA protocol requires the responders to check that

$$e\left(\sum_{i=0}^{n} P_i, P_r\right) = e\left(X, P\right)$$

and verify that the contribution X was recovered successfully. Finally, each member computes the group TEK as $K = H_2(X, Y)$, using the hash function $H_2 : (\mathbb{G}_1)^{n+1} \rightarrow \{0,1\}^*$.

Authentication

In order to provide authentication, the authors suggest applying the compiler of Katz and Yung from [KY03] to transform the protocol into an authenticated GKA protocol.

5.3.4 The ZSM-ID-AGKA Protocol

In [ZSM06], Zhou *et al.* proposed a unique ID-based authenticated group key agreement protocol, which we call the ZSM-ID-AGKA protocol, that varies from the structure of the CKA protocol. In the previous pairing-based protocols, the group leader masks her contribution for each responder, while the responders broadcast their contributions in the clear. In the ZSM-ID-AGKA protocol, the leader masks a secret value X, which *all* members use to encrypt and broadcast their contributions.

Masking Process

The group leader chooses a secret value $X \in_R \{0,1\}^*$, a public value $\delta \in_R \mathbb{G}_2$ and a private ephemeral value $k_0 \in_R \mathbb{Z}_q^\times$. The group leader's key contribution is the value k_0P, although this is never explicitly computed. To encrypt her key contribution, the leader computes $T_0 = (k_0 \cdot H_1(X))\, P$ using the secret value X and the hash function $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^\times$. To mask the secret value X, the group leader computes the shared value

$$\alpha_i = e(d_0, Q_i) = e(Q_0, Q_i)^s = e(Q_0, d_i).$$

Using the hash function $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^*$, she computes the masked value $y_i = X \oplus H_2(\alpha_i \cdot \delta)$ for each responder U_i to produce the set $Y = \{y_1, ..., y_n\}$. The leader broadcasts T_0 and the set Y, along with δ and another value $P_0 = k_0 P_{pub}$ used for verification purposes.

To unmask the value y_i and recover the secret value X, each responder computes $y_i \oplus H_2(\alpha_i \cdot \delta) = X$. The responders then recover the leader's contribution by computing $k_0P = (H_1(X))^{-1} T_0$. Similarly, each responder $U_i \in \mathcal{R}$ chooses a private ephemeral value $k_i \in \mathbb{Z}_q^\times$, uses the secret value X to encrypt their contribution as $T_i = H_1(X) \cdot k_i P$ and broadcasts T_i along with $P_i = k_i P_{pub}$. After recovering all contributions k_iP, the ZSM-ID-AGKA protocol requires each member to check that

$$e\left(P, \textstyle\sum_{i=0}^{n} P_i\right) = e\left(P_{pub}, \textstyle\sum_{i=0}^{n} k_i P\right),$$

before computing the group TEK as

$$K = H_3(k_0P) \oplus H_3(k_1P) \oplus ... \oplus H_3(k_nP),$$

using the hash function $H_3 : \mathbb{G}_1 \rightarrow \{0,1\}^*$.

Authentication

The ZSM-ID-AGKA protocol doesn't explicitly use an ID-based signature scheme to provide authentication. The verification stage does not ensure that messages are sent from the correct users. The protocol relies on the fact that an adversary cannot obtain the key contributions without knowledge of the secret value X and cannot obtain the secret value X without access to one of the long-term private keys used to mask the y_i values. However, the lack of authentication of individual key contributions leaves the protocol vulnerable to an insider impersonation attack, as shown in the next section.

Insider Impersonation Attack

Recently in [PAK09], Park *et al.* showed that the ZSM-ID-AGKA protocol is vulnerable to an insider impersonation attack. Indeed, after recovering the masking value X, a malicious group member U_k can impersonate a member U_i by choosing some $k_i \in \mathbb{Z}_q^\times$ and broadcasting $T_i = H_1(X) \cdot k_i P$ and $P_i = k_i P_{pub}$. As mentioned by the authors, the verification stage does not require any private information of member U_i, which causes the vulnerability to the insider attack.

The authors propose a new version of the ZSM-ID-AGKA protocol that protects against the insider impersonation attack by bringing the members' long-term ID-based public/private key pairs into the batch verification stage. For more details, we refer the reader to [PAK09].

5.3.5 Analysis of CKA-based Protocols

We compare the pairing- and identity-based CKA protocols from [KNKW05, ZSM06, CSCW07, HLL07] with the original CKA protocol from [BN03]. For simplicity, we drop the ID-AGKA suffix from our protocol pseudonyms for this analysis.

Although the CKA protocol may be executed in only a single round, the protocol does not provide perfect forward secrecy. In order to achieve this property, the KNKW, CSCW and HLL protocols require the leader to use ephemeral data broadcasted by the responders in the masking process, inducing an additional broadcast

round. The ZSM protocol also requires an additional round to allow the responders to recover the secret encryption value of the group leader. However, the protocol does not provide perfect forward secrecy.

The KNKW and CSCW protocols provide efficient auxiliary key agreement protocols to update the session key after a membership change in the group. However, the protocols require members to use the same ephemeral values from the previous session, violating our definition of perfect forward secrecy from Section 2.2.1. Thus, while these protocols provide perfect forward secrecy if the initial key agreement scheme is executed following a membership change, it is not provided in the presence of the auxiliary key agreement schemes that have been proposed by the authors. This is denoted by the symbol † in Table 5.6.

The CKA and HLL protocols provide authentication, although they do not specify the use of some explicit signature schemes. Similarly, the KNKW and CSCW protocols suggest that the leader's contribution be signed and verified using some unspecified scheme. This is denoted by the symbol ◊ in Table 5.6. While the KNKW and CSCW protocols use familiar ID-based signature schemes from Chapter 4, they employ the weaker notion of signature screening and thus do not provide proper authentication of member contributions. Although the ZSM protocol suggests some form of batch verification of member contributions, the protocol is vulnerable to an insider impersonation attack, and thus does not provide adequate authentication. We denote the improper authentication of leader and responder contributions by the symbol $*$ in Table 5.6.

Recall that in the KNKW, CSCW and HLL protocols, the group session key is computed as a function of the set of masked values Y and the leader's secret contribution X. Since each responder recovered the leader's secret contribution from one of the masked values in the set Y, they can be sure that their ephemeral key contribution is somehow contained in Y, ensuring freshness of the key. Thus the protocols do not suffer from the mistake discussed in [Tse06], where the leader has key control, and may be called contributory group key agreement protocols. However, since the responders cannot verify that the set Y contains the contributions of other members or does not contain any extraneous contributions, the protocols do not provide verifiable contributiveness or key integrity. The ZSM protocol does not

provide these properties due to the vulnerability to an insider impersonation attack.

Each protocol requires only a single message to be sent by each member, achieving the optimal value for group key agreement protocols, as shown in [BW98]. We note that while the KNKW and CSCW protocols specify that the responders unicast their contributions to the group leader, there is no harm in broadcasting these values to the group. However, the CKA, ZSM and HLL protocols require that the responder contributions be broadcasted to the entire group to ensure proper execution.

		CKA	KNKW	CSCW	ZSM	HLL
Rounds		1	2	2	2	2
Authentication		$\checkmark^{\Diamond}$	$\checkmark^{*\Diamond}$	$\checkmark^{*\Diamond}$	$\checkmark^{*}$	$\checkmark^{\Diamond}$
AKA Protocols		×	✓	✓	×	×
Perfect Forward Secrecy		×	$\checkmark^{\dagger}$	$\checkmark^{\dagger}$	×	✓
Verifiable Contributiveness		×	×	×	×	×
Key Integrity		×	×	×	×	×
Known Attacks		§2.3.6	–	–	§5.3.4	–
Messages (Total)	Unicast	–	n	n	–	–
	Broadcast	$n+1$	1	1	$n+1$	$n+1$

Table 5.6. Communication costs and properties of the pairing-based CKA protocols.

Computational Costs

We compare the computational costs of the various protocols in Table 5.7. Since the original CKA protocol does not specify the use of explicit encryption or signature schemes, we omit it from our discussion. Furthermore, due to the computationally asymmetric nature of the CKA protocol, we consider the computational costs of the group leader U_0 and a responder U_i separately. Finally, since each protocol provides authentication in a different manner, we separate the costs associated with key agreement and authentication for each type of member, as in our analysis of the pairing-based BD protocols. To be comprehensive, we also include the total number of operations for each protocol.

Protocol	Computations (GKA / Authentication)			
	$\mathcal{P}$		$\mathcal{M}$	
	U_0	U_i	U_0	U_i
KNKW-ID-AGKA	n / $n+1$	1 / –	3 / n	– / 3
(Total)	$3n+1$		$4n+3$	
CSCW-ID-AGKA	n / $2n$	1 / 1	2 / –	– / 2
(Total)	$5n$		$2n+2$	
ZSM-ID-AGKA	n / 2	1 / 2	1 / 1	1 / 1
(Total)	$4n+2$		$2n+2$	
HLL-GKA	–	2	$2n+2$	3
(Total)	$2n$		$5n+2$	

Table 5.7. Computational costs of the pairing-based CKA protocols.

Conclusions

From the analysis above, we can compare the approaches of the various pairing-based CKA protocols. As shown in Table 5.6, the KNKW and CSCW protocols have a slight communication advantage over the ZSM and HLL protocols, as they provide the responders with the option of sending their contribution to the leader via either unicast or broadcast channels. In terms of computational costs, the KNKW protocol is the clear winner. Although it requires twice the number of scalar multiplications as CSCW and ZSM, the KNKW protocol offers huge computational gains in terms of the expensive pairing computations, requiring the fewest of all authenticated protocols considered, as shown in Table 5.7.

Each protocol fails to provide adequate authentication, as shown in Table 5.6. In particular, the ZSM protocol is vulnerable to an insider impersonation attack due to lack of authentication. Similar to the pairing-based BD protocols of Section 5.2, the authentication issues of the KNKW and CSCW protocols can be fixed by applying strong batch verification technique of Section 4.2 in place of the signature screening approach.

As the KNKW protocol is the most computationally and communication efficient, we conclude that the KNKW protocol is the most efficient of the pairing-based CKA protocols discussed in this section.

5.4 Summary

In this chapter, we presented the first comprehensive survey and classification of pairing-based group key agreement protocols from current literature. We showed that each protocol is a pairing- or identity-based variation of the Tree-based Group Diffie-Hellman protocol, the Burmester-Desmedt protocol or the computationally asymmetric Conference Key Agreement protocol by outlining the structural and computational differences of the various protocols for each particular class. Furthermore, we provided an in-depth analysis of the communication and computational costs of each protocol, from which we concluded that the tree-based protocol of Lee *et al.* from [LKKR04], the BD-based protocol of Choi *et al.* from [CHL04] and the computationally asymmetric protocol of Kim *et al.* from [KNKW05], provide the most efficient and secure solutions to pairing-based group key agreement for each class.

For each type of group key agreement protocol, we also showed how the bilinear pairing can be applied to improve upon the original scheme. Aside from the obvious advantages of identity-based cryptography, the pairing allows us to efficiently combine a triple of points from our base group. For the tree-based protocols, the pairing allows node keys to be computed using the secret keys of three members, permitting the use of ternary key trees which reduce both round and communication complexity. While the pairing offers no direct advantage in the computation of key contributory material in the BD-based protocols, it can be used more fittingly to verify the computation of key contributions in elliptic curve variations. Finally, the pairing has a powerful application in computationally asymmetric protocols, allowing the group leader to use a combination of both long-term and ephemeral keys in the computation of the masking value, which can yield additional benefits in terms of security.

Finally, we outlined vulnerabilities of several of the pairing-based GKA protocols that stem from the use of signature screening in the authentication process of the protocols. We propose that the protocols use strong batch verification in place of signature screening in order to provide proper authentication and allow for stronger contributory properties, such as verifiable contributiveness and key integrity.

Now that we have a firm understanding of some of the pairing- and identity-based group key agreement protocols from the literature, we propose our own solutions to group key agreement. In the next chapter, we propose a framework for produc-

ing computationally asymmetric authenticated group key agreement protocols from signcryption schemes and provide an example of our framework using schemes from Chapter 4. In Chapter 7, we adapt the generalized framework of Chapter 6 to produce a new identity-based authenticated group key agreement protocol with auxiliary key agreement, suitable for dynamic groups. We go on to show that the proposed protocol provides perfect forward secrecy in the presence of auxiliary key agreement protocols.

Chapter 6

Authenticated Group Key Agreement from Signcryption

In Section 2.1.3, we discussed the class of computationally asymmetric group key agreement (GKA) protocols, in which the group session key was computed as a function of the publicly broadcasted key contributions of the responders and the secret key contribution of the group leader. In this class of protocol, the group leader is required to send her contribution to each responder in some confidential and authenticated manner. In the computationally asymmetric Conference Key Agreement (CKA) protocol from Section 2.3.6, the group leader uses some generic signature and encryption schemes to secure the transmission of her key contribution. The CKA protocol may be thought of as a framework for producing computationally asymmetric GKA protocols from different encryption and signature schemes. For example, the asymmetric group Diffie-Hellman (AGDH) protocol of Section 2.3.7 is a modification of the CKA protocol using the DH key exchange, while the pairing-based computationally asymmetric GKA protocols of Section 5.3 are modifications using some pairing-based encryption schemes.

While the CKA-based protocols use separate encryption and signature schemes to securely transmit the leader's secret contribution, this task can also be achieved using the concept of signcryption from Chapter 4, which by definition provides both confidentiality and authentication in a manner that is more efficient than encryption and signature schemes. In this chapter, we propose a novel framework that

transforms any secure signcryption scheme into a provably-secure computationally asymmetric authenticated group key agreement (AGKA) protocol. Our COMPutationally Asymmetric Signcryption Scheme based Authenticated Group Key Agreement (COMPASS-AGKA) framework, or COMPASS framework for short, achieves the same optimal communication efficiency of the CKA framework, requiring only a single broadcast for each member in a single round of communication, but in a more computationally efficient manner. We reduce the security of the resulting COMPASS-AGKA protocol to the security of the underlying signcryption scheme, in the random oracle model. In addition, the COMPASS framework provides a stronger notion of security through resistance to unknown key share attacks and by providing the desirable contributory properties of Section 2.2.1.

We note that the notion of using signcryption schemes in group key establishment has been considered before. Shortly after introducing signcryption to the cryptographic community, Zheng and Imai proposed the idea of using signcryption schemes for authenticated session key establishment and presented several different protocols for signcryption-based key distribution and key agreement in [ZI98]. Although most of the protocols were for two-party key establishment, one particular protocol was proposed for the case of multicast group communication. Informally speaking, the protocol requires a group leader to choose a random session key and signcrypt it for each responder, taking a distributive approach to key establishment. Soon after, Shoup [Sho99] proposed a simlar two-party key exchange protocol using a sign-then-encrypt form of signcryption, that was later simplified using a monolithic approach by Dodis *et al.* in [DFJW04]. However, as far as we know, the COMPASS framework is the first algorithm to produce efficient and secure GKA protocols from generic signcryption schemes.

In this chapter, we follow the approach of Section 3.4 and build the COMPASS framework in stages. In Section 6.1, we provide a first look at the COMPASS framework in terms of generalized signature and signcryption schemes. We discuss the various advantages of the COMPASS approach in comparison to the CKA protocol and outline the areas where the basic COMPASS framework needs improvement. In Section 6.2, we discuss several methods for optimizing the basic COMPASS framework, that can be achived by choosing the input signcryption scheme to satisfy some

specific properties. In Section 6.3, we present the optimized COMPASS framework and in Section 6.4 we discuss the efficiency of a general COMPASS-AGKA protocol in comparison to other GKA frameworks from the surveys of Section 2.3 and Chapter 5. In Section 6.5, we provide short proofs for the contributory property claims and a formalized proof for the security of the COMPASS framework in the random oracle model. We provide an example to illustrate the COMPASS framework using the signcryption scheme of Yu *et al.* in Section 6.6 and summarize the results of the chapter in Section 6.7.

6.1 COMPASS-AGKA: A First Look

Suppose a group of $n+1$ members $\mathcal{U} = \{U_0, ..., U_n\}$ wish to agree upon a shared session key. Member U_0 has the distinguished position of group leader, while the rest of the members belong to the group of responders, $\mathcal{R}$. Let $\mathcal{S} = (\mathsf{Sign\text{-}KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme and $\mathcal{SC} = (\mathsf{KeyGen}, \mathsf{Sign/Encrypt}, \mathsf{Decrypt/Verify})$ be a signcryption scheme, as defined in Section 4.3.1. We assume each member $U_i \in \mathcal{U}$ obtains a key pair $(\mathsf{SK}_i, \mathsf{VK}_i)$ from $\mathsf{Sign\text{-}KeyGen}$, where SK_i is the private signing key and VK_i is the public verification key, and a key pair $(\mathsf{SDK}_i, \mathsf{VEK}_i)$ from KeyGen, where SDK_i is the private signing/decryption key and VEK_i is the public verification/encryption key.

To begin group communication, the group leader sends out some session identifier, say $session_{ID} \in \{0,1\}^*$. The basic COMPASS-AGKA protocol runs in a single round of communication, as follows:

Round 1 Each responder $U_i \in \mathcal{R}$ chooses a key contribution $N_i \in_R \{0,1\}^*$ and signs the message $m_i =$"$\mathcal{U} \parallel session_{ID} \parallel N_i$", using the signature scheme $\mathcal{S}$ and their signing key SK_i, to obtain the signature σ_i. Member U_i broadcasts $\langle N_i, \sigma_i \rangle$ to the rest of the group.

Meanwhile, the group leader U_0 chooses a secret contribution $N_0 \in_R \{0,1\}^*$. For each responder $U_i \in \mathcal{U}$, the group leader uses the signcryption scheme $\mathcal{SC}$, her private key SDK_0 and the responder's public key VEK_i, to signcrypt the message $m_0 =$"$\mathcal{U} \parallel session_{ID} \parallel N_0$", producing the ciphertext c_i. She then broadcasts the set of signcrypted ciphertexts $\langle c_1, ..., c_n \rangle$, along with some list indicating which ciphertext c_i corresponds to which responder U_i.

Key Computation Upon receiving the set of signcrypted values from the group leader, each responder U_i extracts their specific signcrypted value c_i and, using their private key SDK_i, the public key VEK_0 of the leader and the signcryption scheme $\mathcal{SC}$, designcrypts it to recover the leader's authenticated secret contribution N_0.

Each group member $U_i \in \mathcal{U}$ verifies the signature σ_j for every other member U_j, $1 \leq j \leq n,\ j \neq i$, using their verification key VK_j. If the verification succeeds, they compute the session key as a function of the public key contributions $\{N_1, ..., N_n\}$ of the responders and the secret key N_0 of the group leader, given by

$$K = H(N_1 \parallel ... \parallel N_n) \oplus H(\mathcal{U} \parallel session_{ID} \parallel N_0),$$

where $H : \{0,1\}^* \rightarrow \{0,1\}^*$ is some cryptographic hash function.

We see that the basic COMPASS framework, as presented above, provides the same communication benefits as the CKA protocol. We require each member to broadcast one message in only a single round of communication, which is the lower bound for group key agreement protocols, as shown in [BW98]. By requiring that the group leader encrypt her secret contribution using a signcryption scheme, rather than using signature and encryption schemes as in the CKA protocol, we can increase the computational efficiency of our protocol. Furthermore, by requiring that the public key contributions of the responders are authenticated as well, we can ensure the properties of verifiable contributiveness and key integrity, as we will see in Section 6.5.1. In addition, by requiring that the session key computation include the multicast group $\mathcal{U}$ and the session identifier $session_{ID}$, we can ensure that the COMPASS framework is not vulnerable to the unknown key share attack on the CKA protocol, described in Section 2.3.6.

However, there are some clear disadvantages of the basic COMPASS framework. Each member U_i is required to store two sets of key pairs, $(\mathsf{SK}_i, \mathsf{VK}_i)$ and $(\mathsf{SDK}_i, \mathsf{VEK}_i)$, which complicates the management of public key certificates. For each responder $U_i \in \mathcal{R}$, the group leader must first verify the public key certificate of the responder's designcryption key VEK_i, then use the key to signcrypt her secret contribution, resulting in n certificate verifications and n executions of the Sign/Encrypt algorithm.

In the key computation round, each member will have to first verify the public key certificate of every other member U_j's verification key VK_j, then verify the signature for each key contribution, resulting in $n-1$ certificate verifications and $n-1$ signature verifications for each responder and n of each type of verification for the group leader. Thus the computational costs may be quite expensive in comparison to the CKA protocol, and may create performance bottlenecks in both the first round and the key computation round. Finally, we note that, since the responders only need their private signing/decryption key to recover the group leader's secret contribution, the disclosure of any responder's key will compromise the key of every session in which that responder was a participant. Thus, similar to the CKA protocol, the COMPASS framework does not provide perfect forward secrecy.

6.2 Optimizations for COMPASS-AGKA

In this section, we describe several optimizations for the COMPASS framework that can be achieved by choosing signcryption schemes that satisfy some specific properties, defined in Section 4.3.3. In particular, we investigate the use of ID-based signcryption in Section 6.2.1 and multi-receiver signcryption in Section 6.2.2, discuss issues dealing with signature verification in Section 6.2.3 and address the argument of forward security versus message recovery in Section 6.2.4. For each criterion, we emphasize the effect on the COMPASS framework and discuss which signcryption approach ($\mathcal{S}t\mathcal{E}$, $\mathcal{E}t\mathcal{S}$ or monolithic $\mathcal{SC}$) best supports the criterion. We note that by $\mathcal{S}t\mathcal{E}$ and $\mathcal{E}t\mathcal{S}$, we refer to the more efficient cooperative approach, discussed in Section 4.3.1.

6.2.1 ID-based Signcryption

We can improve and simplify the COMPASS framework from an administrative perspective by using an identity-based signature (IBS) scheme, as given in Definition 4.1 of Section 4.1.1, and an identity-based signcryption (IBSC) scheme, as given in Definition 4.4 of Section 4.3.2. As we mentioned in Section 3.3, we can simplify the management of public key certificates by using an identity-based public key infrastructure (ID-PKI), such as the familiar ID-PKI of Boneh and Franklin from Section 3.4.

Once the ID-PKI is established, the group members can use the identity-based public/private key pairs of the other members without the burden of verifying public key certificates.

Furthermore, the same ID-based key pair can be used for both the signature and signcryption schemes. This not only reduces the storage space required for public/private key pairs, but also eliminates any issues that may arise with having to manage multiple key pairs.

Finally, we note that, due to the benefits of identity-based cryptography, many of the signcryption schemes proposed in the literature, such as those in [ML02, Boy03, LQ03, NR03, CYHC04, LQ04a, BLMQ05, CML05, CLLC06, YYHZ07, SVG+08b], are based on an ID-PKI. The nature of the COMPASS framework is that we are producing efficient protocols from input signcryption schemes. The more signcryption schemes that we have at our disposal, the more freedom we have to choose an input scheme that satisfies some specific criteria. Hence, by considering identity-based signcryption, we have a wealth of schemes from which we can construct efficient COMPASS-AGKA protocols. As a result, the ID-PKI is the ideal public key environment for the COMPASS framework.

Although no single approach provides better support for identity-based signcryption, a survey of identity-based signcryption schemes from the literature shows that many authors opt for the sign-then-encrypt approach, over the encrypt-then-sign and monolithic signcrypt style of signcryption. Of the IBSC schemes considered, only those proposed in [CYHC04, CLLC06] and [LQ03, Schemes 1 and 2] take a cooperative $\mathcal{E}t\mathcal{S}$ approach, while the schemes proposed in [ML02, Boy03, NR03, LQ04a, BLMQ05, CML05, YYHZ07, SVG+08b] are all cooperative $\mathcal{S}t\mathcal{E}$ schemes. The protocol from [LQ03, Scheme 3] is the only one that takes a true monolithic approach, in the same manner as Zheng's original signcryption scheme from [Zhe97].

6.2.2 Multi-Receiver Signcryption

By using a signcryption scheme that efficiently supports multiple recipients, as discussed in Section 4.3.3, we can reduce the computational costs of the group leader. By definition, signcryption is more efficient than encryption and signature schemes separately. However, this does not imply that signcryption for multiple recipients is

more efficient than $\mathcal{S}t\mathcal{E}$ or $\mathcal{E}t\mathcal{S}$ schemes.

Zheng first proposed a multi-receiver extension of his monolithic two-party signcryption scheme in [Zhe97] where the sender individually signcrypts the message for each recipient. Most monolithic schemes provide multi-receiver support in the same manner. However, this approach does not provide efficient scalability as it requires the sender to execute the two-party signcryption scheme for each recipient, inducing n executions of the **Sign/Encrypt** algorithm.

As discussed by An *et al.* in [ADR02], the $\mathcal{E}t\mathcal{S}$ approach compromises message confidentiality when extended to the multi-receiver model. To protect against this attack, the authors include the identity of the recipient in the **Sign** algorithm and the identity of the sender in the **Encrypt** algorithm. Unfortunately, this requires n executions of both the **Sign** and **Encrypt** algorithms, one for each individual recipient. This is more computationally expensive than the monolithic approach.

In [Boy03], Boyen showed that his $\mathcal{S}t\mathcal{E}$ scheme provides maximum scalability and is better suited for multi-receiver signcryption than monolithic schemes. In his protocol, the sender produces a shared signature σ (and some randomization parameter r) and encrypts the signature for each individual recipient, requiring only a single execution of the **Sign** algorithm and n executions of the **Encrypt** algorithm. Following Boyen's approach, most $\mathcal{S}t\mathcal{E}$ schemes can be extended to support multiple recipients. In [ADR02], An *et al.* also showed that multi-receiver $\mathcal{S}t\mathcal{E}$ schemes are vulnerable to surreptitious forwarding (see Section 4.3.3) and proposed that we used the same modification as in the multi-receiver $\mathcal{E}t\mathcal{S}$ scheme. However, in the COMPASS framework, we are not concerned with surreptitious forwarding, since we assume that the group members do not divulge the leader's secret key contribution to members outside of the group. Thus, the $\mathcal{S}t\mathcal{E}$ style of signcryption schemes provide the most efficient support for multiple recipients, in the context of the COMPASS framework.

6.2.3 Signature Verification

Recall from Section 4.3.3 that an $\mathcal{S}$-verifiable signcryption scheme produces a detachable signature that can be verified using the signature scheme $\mathcal{S}$. By using an $\mathcal{S}$-verifiable signcryption scheme, along with the signature scheme $\mathcal{S}$, in the COMPASS

framework, we can verify the detachable signature for the leader's secret contribution and the signatures for the responders' public contributions using the same verification algorithm. A nice consequence of this property is that we can bind the signcryption scheme $\mathcal{SC}$ of the group leader to the signature scheme $\mathcal{S}$ of the responders, and view the input to the COMPASS framework as a single scheme $\mathcal{SC}$, rather than a pair of schemes $(\mathcal{S}, \mathcal{SC})$. For the remainder of this section, we suggest further optimizations that can be obtained by choosing an $\mathcal{S}$-verifiable signcryption scheme for which the signature scheme $\mathcal{S}$ satisfies some criteria.

In Section 4.1.5, we presented the ZLK-IBS scheme that Zhang *et al.* used to provide authentication in Joux's protocol (see Section 3.2), producing the ZLK protocol. Rather than having the participants broadcast their key contribution and a signature for their contribution, Zhang *et al.* simply used a part of the signature as the participant's public key contribution, reducing the size of the broadcast message. Most signature schemes from the literature require the input of some randomized ephemeral data, say r, and produce a signature containing some element X that is authentically linked to the signer (via some other element in the signature) and is produced as a function of the value r. For example, in the ChCh-IBS scheme of Section 4.1.3, a signer with public/private key pair $(Q_{\mathsf{ID}}, d_{\mathsf{ID}})$ produces a signature of the form $\sigma_{\mathsf{ChCh}} = (X, Z)$, where $X = rQ_{\mathsf{ID}}$ is authentically linked to the signer through the value $Z = (r + H_1(X, m))d_{\mathsf{ID}}$, where $H_1 : \mathbb{G}_1 \times \{0,1\}^* \rightarrow \mathbb{Z}_q^\times$. If our signatures from $\mathcal{S}$ contain such a value X, we can modify the COMPASS framework so that the responders use the value X as their publicly broadcasted key contribution, rather than choosing a random string N_i and producing a signature for N_i. As a result, we can reduce the size of the broadcast messages, as in the ZLK protocol.

Finally, the initial COMPASS framework from Section 6.1 requires each member to individually verify the signed contribution of every other member in the group, in order to provide the properties of verifiable contributiveness and key integrity. However, this drastically increases the computational load of the members, which can outweigh the advantages of the contributory properties and the asymmetric approach of the COMPASS framework. Rather than verifying each signature individually, we can verify all the signatures at once and substantially reduce the computational costs by requiring that the signature scheme $\mathcal{S}$ supports batch verification, discussed in

Section 4.2. Furthermore, we want $\mathcal{S}$ to provide strong batch verification. The weaker notion of signature screening does not guarantee the authenticity of all key contributions and thus the resulting COMPASS-AGKA protocol will not provide verifiable contributiveness or key integrity, which was our motivation for verifying all of the signatures in the first place.

When comparing the various approaches, we simply consider which approach best provides the $\mathcal{S}$-verifiability property. Using a monolithic signcryption approach, the message is signed and encrypted for a specific recipient. In general, this approach does not naturally produce a detachable signature which can be verified by all the members of the group. Similarly, the $\mathcal{E}t\mathcal{S}$ approach requires the signer to produce a signature for the encrypted contribution rather than the contribution itself, and thus does not provide $\mathcal{S}$-verifiability. Conversely, $\mathcal{S}$-verifiability is inherently provided using the $\mathcal{S}t\mathcal{E}$ approach, since the leader's contribution is first signed using some signature scheme, then encrypted for the intended recipient. Thus we must only find a $\mathcal{S}t\mathcal{E}$ scheme in which the underlying signature scheme $\mathcal{S}$ provides strong batch verification.

6.2.4 Forward Security vs. Message Recovery

As defined in Section 4.3.3, a signcryption scheme provides past message recovery, if a ciphertext can be designcrypted using only the sender's private key, and forward security, otherwise. We note that neither of these properties have any effect on the efficiency of the resulting COMPASS-AGKA protocol. However, given a forward secure signcryption scheme, the corresponding COMPASS-AGKA protocol will at least have partial forward secrecy, since we may disclose the private key of the group leader without compromising her secret contribution. We suggest using a signcryption scheme with forward security, rather than message recovery, in the COMPASS framework.

We note that we cannot provide perfect forward secrecy in the COMPASS framework through our choice of signature or signcryption schemes. The very nature of signcryption is that the receivers can recover the message using only their long-term private key, which in turn compromises the session key in the COMPASS framework. However, in Chapter 7, we present a forward secure ID-based authenticated group key agreement protocol that illustrates that, for some signcryption schemes, the resulting

COMPASS-AGKA protocol can be modified to provide perfect forward secrecy, at the expense of requiring an additional round of communication.

6.3 The COMPASS Framework

Applying the optimizations of Section 6.2 to the basic COMPASS framework presented in Section 6.1, we obtain the optimized COMPASS framework.

Let $\mathcal{SC}$ = (**Setup**, **Extract**, **Sign/Encrypt**, **Decrypt/Verify**) be a provably-secure identity-based signcryption scheme, as in Definition 4.4 of Section 4.3.2, providing the following properties:

(i) efficient support for multiple recipients,

(ii) $\mathcal{S}$-verifiability, for a batch verifiable identity-based signature scheme $\mathcal{S}$ = (**Setup**, **Extract**, **Sign**, **Verify**, **Batch**),

(iii) forward security.

Suppose a group of $n+1$ members $\mathcal{U} = \{U_0, ..., U_n\}$ want to agree upon a shared session key, where member U_0 is the group leader and the rest of the members belong to the group of responders $\mathcal{R}$. Using the **Setup** and **Extract** algorithms, each group member $U_i \in \mathcal{U}$ with identity ID_i is associated with the public/private key pair (ID_i, d_i). To begin group communication, the group leader sends out some session identifier, say $session_{ID} \in \{0,1\}^*$. The COMPASS-AGKA protocol runs in a single round of communication, as follows:

Round 1 Each responder $U_i \in \mathcal{R}$ signs the message "$\mathcal{U} \parallel session_{ID}$" using the signature scheme $\mathcal{S}$ to obtain the signature σ_i, with ephemeral public value X_i that simultaneously fulfills the role of the responder's key contribution. Member U_i broadcasts the signature σ_i to the rest of the group.

The group leader U_0 chooses a secret contribution $N \in_R \{0,1\}^*$ and signcrypts the message "$\mathcal{U} \parallel session_{ID} \parallel N$" for each responder using the signcryption scheme $\mathcal{SC}$, producing the ciphertext set $\langle c_1, ..., c_n \rangle$. She then broadcasts the set of signcrypted ciphertexts, along with some list indicating which ciphertext c_i corresponds to which responder U_i.

Key Computation Upon receiving the set of signcrypted values from the group leader, each responder U_i extracts their specific signcrypted value c_i and, using their long-term private key d_i, the long-term public key of the leader ID_0 and the signcryption scheme $\mathcal{SC}$, decrypts it to recover the leader's secret contribution N and the detachable signature σ_0.

Each group member batch verifies the signatures $\{\sigma_0, ..., \sigma_n\}$ using the **Batch** algorithm. If the verification succeeds, they compute the session key as a function of the public contributions $\{X_1, ..., X_n\}$ of the responders and the secret contribution N of the group leader. The session key is given by

$$K = H_1(X_1, ..., X_n) \oplus H_2(\mathcal{U} \parallel session_{ID} \parallel N),$$

for some cryptographic hash functions H_1 and H_2.

A summary of the protocol is given in Table 6.1.

COMPASS-AGKA
Round 1 $U_i : \sigma_i \leftarrow \mathsf{Sign}(\text{“}\mathcal{U} \parallel session_{ID}\text{”}, d_i),\ 1 \leq i \leq n$ $U_i \xrightarrow{B} \mathcal{U} : \sigma_i,\ 1 \leq i \leq n$ $U_0 : N \xleftarrow{r} \{0,1\}^*$ $U_0 : c_i \leftarrow \mathsf{Sign/Encrypt}(\text{“}\mathcal{U} \parallel session_{ID} \parallel N\text{”}, d_0, \mathsf{ID}_i), 1 \leq i \leq n$ $U_0 \xrightarrow{B} \mathcal{U} : \langle c_1, ..., c_n \rangle$
Key Computation $U_i : \langle \text{“}\mathcal{U} \parallel session_{ID} \parallel N\text{”}, \sigma_0 \rangle \leftarrow \mathsf{Decrypt/Verify}(c_i, d_i, \mathsf{ID}_0), 1 \leq i \leq n$ $U_i : \mathsf{Batch}((\text{“}\mathcal{U} \parallel session_{ID} \parallel N\text{”}, \sigma_0, \mathsf{ID}_0),$ $(\text{“}\mathcal{U} \parallel session_{ID}\text{”}, \sigma_1, \mathsf{ID}_1), ..., (\text{“}\mathcal{U} \parallel session_{ID}\text{”}, \sigma_n, \mathsf{ID}_n)),\ 0 \leq i \leq n$ $K = H_1(X_1, ..., X_n) \oplus H_2(\mathcal{U} \parallel session_{ID} \parallel N)$

Table 6.1. The COMPASS-AGKA Protocol

6.4 Efficiency

We compare the efficiency of the COMPASS framework with the tree-based TGDH protocol of Section 2.3.4, the BD protocol of Section 2.3.5 and the CKA protocol from Section 2.3.6 in terms of communication costs and key agreement properties in Table 6.1. In Section 6.6.1, we propose a concrete example of the COMPASS-AGKA protocol for the YYHZ-IBSC scheme of Section 4.3.6, and provide a more thorough account of the computational costs in this case.

As discussed in Section 6.1, the COMPASS protocol requires only a single round of communication, which is the optimal round complexity for any key agreement protocol. Similarly, the CKA protocol runs in a single round of communication, while the BD protocol requires 2 rounds of communication. However, the TGDH protocol does not provide constant round complexity, requiring as many rounds as the height $h = \lceil \log_2 (n+1) \rceil$ of the binary key tree, as discussed in Section 5.1.6.

As mentioned in Section 6.1, the COMPASS and CKA protocols do not provide perfect forward secrecy, as the secret key contribution of the leader is encrypted for each responder using their long-term private key. Since the BD and TGDH protocols use ephemeral keys to establish the session key, they can provide inherent perfect forward secrecy by re-executing the protocol for each session. As mentioned in Section 5.1.6, the TGDH protocol also proposes auxiliary key agreement protocols to rekey the group following a membership change.

Since each member verifies the signature of all other members, the COMPASS protocol provides the strongest possible form of authentication. Furthermore, since the members only use the authenticated contributions to compute the session key, the COMPASS protocol provides the properties of group integrity, verifiable contributiveness and key integrity. As discussed in the analyses of Chapter 5, the TGDH protocol does not provide authentication and cannot guarantee any of the contributory properties. While the BD protocol does provide authentication, it cannot guarantee any of the contributory properties. The CKA protocol provides group integrity, since the key is computed using only member contributions. However, since the protocol does not require authentication of member contributions (indicated by $\Diamond$ in Table 6.2), it does not provide verifiable contributiveness or key integrity.

Finally, the COMPASS protocol requires only one broadcast message by each

group member, which meets the optimal value for group key agreement, as shown in [BW98]. While the CKA protocol achieves the same lower bound, the TGDH and BD protocols require twice as many messages to be sent.

		COMPASS	TGDH	BD	CKA
Rounds (Max)		1	$\lceil \log_2 (n+1) \rceil$	2	1
Perfect Forward Secrecy		§7	✓	✓	×
AKA Protocols		§7	✓	×	×
Contributory Key		✓	✓	✓	✓
Authentication		✓	–	✓	$\checkmark^{\diamond}$
Group Integrity		✓	×	×	✓
Verifiable Contributiveness		✓	×	×	×
Key Integrity		✓	×	×	×
Messages (Total)	Unicast	–	–	–	–
	Multicast	$n+1$	$2n$	$2(n+1)$	$n+1$

Table 6.2. Communication Costs of the COMPASS-AGKA Protocol.

6.5 Security Proofs

We claim that the proposed COMPASS-AGKA protocol is a secure authenticated group key agreement protocol, as given in Definition 2.2 of Section 2.2.2, that provides the contributory properties of verifiable contributiveness (and thus group integrity) and key integrity, as defined in Section 2.2.1. In this section, we provide proofs to support these claims. In Section 6.5.1, we give short, informal proofs of the contributory properties. In Section 6.5.2, we prove the semantic security of the COMPASS-AGKA protocol in the random oracle model.

6.5.1 Contributivity Proofs

The proposed COMPASS-AGKA protocol produces a fresh key from randomly chosen member contributions in each session and thus trivially provides the property of key independence. It remains to show that the protocol provides the contributory properties of verifiable contributiveness and key integrity.

Claim 6.1 *The proposed* COMPASS-AGKA *protocol provides verifiable contributiveness.*

Proof: In order to provide verifiable contributiveness, we must show that member U_i is assured that his contribution X_i and the contribution X_j of every other member U_j, $j \neq i$ is used in the computation of the session key. In addition, each member should have knowledge of when another member has *not* contributed to the key.

Recall that each member U_j is required to output a signature that authentically links them to their contribution X_j. After receiving all the signatures (and recovering the leader's contribution), member U_i batch verifies the signatures of all members (including his own), using the identity-based public keys of the members. At this point, U_i will become aware if another member has not sent a contribution for the session key.

If the verification holds, member U_i computes $K = H_1(X_1, ..., X_n) \oplus H_2(\mathcal{U} \parallel session_{ID} \parallel N)$ as a product of the authenticated contributions $\{X_1, ..., X_n, N\}$ that were received during the protocol. Through this step, member U_i is assured that the session key K has been authentically contributed to by every member in the group, providing verifiable contributiveness. ∎

Claim 6.2 *The proposed* COMPASS-AGKA *protocol provides key integrity.*

Proof: To prove key integrity, we must show that the session key K consists only of contributions of the group members. An adversary $\mathcal{A}$ can attack the key integrity property by introducing an extraneous contribution to the session key or by tampering with the contributions of the intended members.

Following the argument of the previous claim, we see that each responder signs the session identifier (producing their contribution X_i), while the leader signs her randomly chosen key contribution N. Since the signature and signcryption schemes provide unforgeability, the batch verification stage ensures that all contributions are authentic and thus have not been tampered with by an adversary. If we had required the weaker notion of screening from [BGR98], then the verification of the batch may pass, although all signatures may not necessarily pass individual verification.

Furthermore, since each group member is required to compute the session key as a product of *only* the authenticated contributions $\{X_1, ..., X_n, N\}$, they are assured that $\mathcal{A}$ has not introduced any extraneous contributions. Thus each group member is assured that the session key is produced using only the contributions of the intended group members, providing key integrity. ∎

6.5.2 Semantic Security

To prove that the COMPASS-AGKA protocol with input signcryption scheme $\mathcal{SC}$ is a secure authenticated group key agreement protocol, we must show that it satisfies the validity and indistinguishability requirements of Definition 2.2 in Section 2.2.2. Since the validity of the protocol is straightforward, we will show that the COMPASS-AGKA protocol satisfies the indistinguishability requirement.

As in the group key agreement security model of Section 2.2.2, we prove the semantic security of the COMPASS-AGKA protocol using a security game between an adversary $\mathcal{A}$ and an infinite set of oracles $\{\Pi_U^\alpha\}$, $U \in \mathcal{U}$ and $\alpha \in \mathbb{N}$. Since the COMPASS framework is defined in terms of a general signcryption scheme, we cannot directly reduce the security to some hard mathematical problem. In the COMPASS framework the only secret key contribution is that of the group leader and the confidentiality of her contribution is ensured by the signcryption scheme. Following the approach of Boyd and González Nieto in [BN03], we show that an adversary $\mathcal{A}$ with a non-negligible advantage $\mathrm{Adv}_{\mathsf{COMPASS-AGKA}}$ in breaking a simulation of the COMPASS-AGKA protocol can be used to break the security of the underlying signcryption scheme. We note that we need not reduce the security to the responder signature scheme $\mathcal{S}$, since a forgery of the responder contribution does not give the adversary any advantage in distinguishing between the Test keys of the semantic security game.

We assume that $\mathcal{SC}$ is an $\mathcal{S}$-verifiable identity-based signcryption scheme that is both EUF-IBSC-CMA secure (i.e. existentially unforgeable against chosen message attacks) as defined in Definition 4.6 of Section 4.3.4, and IND-IBSC-CCA2 secure (i.e. semantically secure under adaptive chosen-ciphertext attacks), as defined in

Definition 4.5 of Section 4.3.4, in the multi-receiver signcryption model. Informally speaking, we simultaneously play the role of the adversary in the EUF-IBSC-CMA or IND-IBSC-CCA2 security game, while also playing the role of the set of oracles in the AGKA security game with the COMPASS adversary. If the adversary has any advantage, we use it to gain an advantage in the signcryption security games.

Since we prove the security against an active adversary, we allow $\mathcal{A}$ to issue Send, Execute, Reveal, Corrupt and Test queries, as given in Section 2.2.2. In order to properly answer the queries of $\mathcal{A}$, we issue queries of our own to the challenger in the signcryption security games. For example, if the adversary gains an advantage by forging a signature within a ciphertext in $\mathcal{SC}$, we use the Extract oracle of the EUF-IBSC-CMA security game to answer the Corrupt queries of the adversary.

To simplify the proof, we assume that the multicast group $\mathcal{U} = \{U_0, ..., U_n\}$ remains the same from one session to another. The dynamic case is handled similarly. We allow the adversary to choose the leader from the group $\mathcal{U}$ for each session. To indicate the different possible roles of the members, we use the notation $\{U_{i_0}, U_{i_1}, ..., U_{i_n}\}$ to represent the multicast group of the i^{th} session, where each U_{i_j}, $0 \leq j \leq n$, corresponds to some $U_k \in \mathcal{U}$. To be as general as possible, we assume that we do not have prior knowledge of the adversary's choice for the group leader.

Let $\mathrm{Adv}_{\mathsf{COMPASS-AGKA}}(t, q_{ex})$ denote the maximum advantage of any adversary attacking the COMPASS-AGKA protocol in running time t and making q_{ex} queries to the Execute oracle. We suppose that the hash functions H_0, H_1 and H_2 are modelled as random oracles.

Theorem 1 *The proposed* COMPASS-AGKA *protocol with identity-based signcryption scheme $\mathcal{SC}$ is a secure AGKA protocol. Specifically,*

$$\mathrm{Adv}_{\mathsf{COMPASS-AGKA}}(t, q_{ex}) \leq (n+1)\, q_{ex}\, \mathrm{Adv}_{\mathcal{SC}}^{\mathsf{Distinguish}}(t) + (n+1)\, \mathrm{Adv}_{\mathcal{SC}}^{\mathsf{Forge}}(t),$$

where $\mathrm{Adv}_{\mathcal{SC}}^{\mathsf{Forge}}(t)$ is the maximum advantage of any EUF-IBSC-CMA forger $\mathcal{F}$ of the signcryption scheme $\mathcal{SC}$ and $\mathrm{Adv}_{\mathcal{SC}}^{\mathsf{Distinguish}}(t)$ is the maximum advantage of any IND-IBSC-CCA2 algorithm $\mathcal{D}$ attacking the indistinguishability of the signcryption scheme $\mathcal{SC}$, all running in time t.

Following the approach of Boyd and González Nieto from [BN03], we divide the

proof into two separate cases. We assume that the adversary gains her advantage by (1) forging the leader's signature within the ciphertext for the signcryption scheme $\mathcal{SC}$ or (2) breaking the protocol without altering authentication transcripts. In case (1), we use the adversary $\mathcal{A}$ to construct a forging algorithm $\mathcal{F}$ for the signcryption scheme $\mathcal{SC}$. In case (2), we use the adversary to build a distinguishing algorithm $\mathcal{D}$ to attack the semantic security of $\mathcal{SC}$.

Since we are considering identity-based signcryption schemes, our proofs are given in these terms. In particular, we assume the adversary has knowledge of the identities of all users and can obtain public keys using the random oracle H_0 and, when specified, private keys using the Extract query.

Signature within the Ciphertext Forgery on $\mathcal{SC}$

Suppose an adversary $\mathcal{A}$ gains her advantage by forging the signed contribution (within the ciphertext) of the group leader, thereby fooling the responders into sharing a session key with him. We can use the adversary to construct a forger $\mathcal{F}$ for the signcryption scheme $\mathcal{SC}$ with non-negligible advantage in the EUF-IBSC-CMA security game of Section 4.3.4, adapted for the multi-receiver model. To account for the $\mathcal{S}$-verifiability of the signcryption scheme $\mathcal{SC}$, we modify the Decrypt/Verify oracle to output a valid message-signature pair, rather than just the message itself.

The task of the forger $\mathcal{F}$ is as follows: for a user U_{ID^*}, with identity ID^* and public key Q_{ID^*}, and a set of receivers $\{\mathsf{ID}_1^*, ..., \mathsf{ID}_n^*\}$, produce an existential forgery of a signcrypted ciphertext c^* such that, under the private key of some receiver U_i with identity ID_i, $1 \leq i \leq n$, the Decrypt/Verify$(c^*, \mathsf{ID}^*, \mathsf{ID}_i)$ oracle returns a valid message-signature pair (m, σ) from user U_{ID^*}. The forger is allowed to make following queries:

- Extract(ID_i) to determine the long-term private key d_i for any user with identity $\mathsf{ID}_i \notin \mathsf{ID}^*$.

- Sign/Encrypt$(m, \mathsf{ID}, \{\mathsf{ID}_{i_1}, ..., \mathsf{ID}_{i_n}\})$ to produce the set of signcrypted ciphertexts $\langle c_1, ..., c_n \rangle$ of a message m from sender ID to a set of receivers $\{\mathsf{ID}_{i_1}, ..., \mathsf{ID}_{i_n}\}$.

- Decrypt/Verify$(c, \mathsf{ID}, \mathsf{ID}_i)$ produces the valid message-signature pair (m, σ) for sender ID or outputs $\perp$ if the ciphertext c does not decrypt to a valid message-signature pair under the private key of receiver ID_i.

Naturally, we assume that $\mathcal{F}$ outputs a signcrypted ciphertext that was not the result of a Sign/Encrypt(m, ID*, $\{\mathsf{ID}_{j_1}, ..., \mathsf{ID}_{j_n}\}$) query, for any receivers $\{\mathsf{ID}_{j_1}, ..., \mathsf{ID}_{j_n}\}$. We have the following result.

Lemma 6.1 *Let* Forge *be the event that an adversary $\mathcal{A}$ outputs a valid forgery for the signcryption scheme $\mathcal{SC}$. Then*

$$\Pr[\mathsf{Forge}] \leq (n+1)\,\mathrm{Adv}^{\mathsf{Forge}}_{\mathcal{SC}}(t).$$

Proof: We suppose the adversary $\mathcal{A}$ gains her advantage by forging the signature (within ciphertext) of the group leader U_ρ. The forger $\mathcal{F}$ chooses a random $U_d \in \mathcal{U}$ with identity ID_d as its guess for the leader U_ρ and as the user for which he wishes to forge a signcrypted ciphertext in the EUF-IBSC-CMA security game. $\mathcal{F}$ honestly generates the public/private key pairs for each user with identity different from ID_d using the Extract oracle and, using these keys and the Sign/Encrypt and Decrypt/Verify oracles, simulates the group key oracle queries of $\mathcal{A}$ in the usual way. If at any point, $\mathcal{A}$ issues the Corrupt query on ID_d, then $\mathcal{F}$'s guess for the value of ρ was incorrect and $\mathcal{F}$ aborts the simulation. However, if $\mathcal{A}$ outputs a new ciphertext c that decrypts to a valid message-signature pair for U_d, resulting in the event Forge, $\mathcal{F}$ returns the ciphertext.

Suppose that $\mathcal{A}$ succeeds in forging a signature within ciphertext with probability $\Pr[\mathsf{Forge}]$. The probability that this is a forgery for the user U_d is $1/(n+1)$. Thus, the probability of success for $\mathcal{F}$ is given by $\mathrm{Adv}^{\mathsf{Forge}}_{\mathcal{F},\mathcal{SC}}(t) = \frac{1}{n+1}\Pr[\mathsf{Forge}]$, and since $\mathrm{Adv}^{\mathsf{Forge}}_{\mathcal{F},\mathcal{SC}}(t) \leq \mathrm{Adv}^{\mathsf{Forge}}_{\mathcal{SC}}(t)$, we have $\Pr[\mathsf{Forge}] \leq (n+1)\,\mathrm{Adv}^{\mathsf{Forge}}_{\mathcal{SC}}(t)$. ∎

Indistinguishability Attack on $\mathcal{SC}$

Now suppose that an adversary $\mathcal{A}$ gains her advantage without forging the protocol transcripts. We use $\mathcal{A}$ to build a distinguishing algorithm $\mathcal{D}$ for the signcryption scheme $\mathcal{SC}$ that has non-negligible advantage in the IND-IBSC-CCA2 security game of Section 4.3.4, adapted for the multi-receiver model. Similar to the proof of the previous section, we modify the Decrypt/Verify oracle to account for $\mathcal{S}$-verifiability.

The task of the algorithm $\mathcal{D}$ is as follows: given a signcrypted ciphertext c^* from Sign/Encrypt(m_θ, ID*, $\{\mathsf{ID}_1^*, ..., \mathsf{ID}_n^*\}$) for $\theta \in_R \{0,1\}$, where the messages m_0 and m_1, the identity ID* of the sender U_{ID^*} and the set of receivers $\{\mathsf{ID}_1^*, ..., \mathsf{ID}_n^*\}$ are specified by $\mathcal{D}$, decide whether $\theta = 0$ or $\theta = 1$. The algorithm is allowed to issue the same queries from the previous section, with the exception that $\mathcal{D}$ may issue the Extract(ID_i) query for identity ID*, but not for any receiver identity $\mathsf{ID}_i^* \in \{\mathsf{ID}_1^*, ..., \mathsf{ID}_n^*\}$. As a result, we do not allow the adversary to issue the Corrupt query for any responder in the group. We assume that $\mathcal{D}$ does not issue a Decrypt/Verify query for the ciphertext c^* for any receiver in $\{\mathsf{ID}_1^*, ..., \mathsf{ID}_n^*\}$.

Lemma 6.2 *Let* Distinguish *be the event that an adversary $\mathcal{A}$ can distinguish between ciphertexts of the signcryption scheme $\mathcal{SC}$. Then*

$$\Pr[\mathsf{Distinguish}] \leq (n+1)\, q_{ex}\, \mathrm{Adv}_{\mathcal{SC}}^{\mathsf{Distinguish}}(t),$$

where q_{ex} is the number of Execute *queries issued by $\mathcal{A}$.*

Proof: We suppose the adversary $\mathcal{A}$ gains his advantage by distinguishing between the signcrypted ciphertexts sent by the group leader U_ρ. The algorithm $\mathcal{D}$ chooses a random $U_d \in \mathcal{U}$ with identity ID_d as its guess for the group leader U_ρ and the sender of the signcrypted ciphertext in the IND-IBSC-CCA2 security game, and sets the remaining members of $\mathcal{U}$ as the receivers. The distinguisher $\mathcal{D}$ chooses $m_0, m_1 \in_R \{0,1\}^*$ and outputs the sender identity ID_d, the set of receiver identities $\{\mathsf{ID}_i\}_{i \neq d}$ and the messages $\{m_0, m_1\}$ in IND-IBSC-CCA2 security game. The challenger responds with the challenge signcryption ciphertext c^* = Sign/Encrypt(m_θ, ID_d, $\{\mathsf{ID}_i\}$), which has the form $\langle \gamma_1, ..., \gamma_n \rangle$ since the plaintext message is signcrypted for each individual receiver. We suppose that $\mathcal{A}$ makes a maximum of q_{ex} Execute queries, where q_{ex} is polynomial in the security parameter ℓ. $\mathcal{D}$ chooses a random session identifier $\beta \in [1, q_{ex}]$ for which he will give the adversary the challenge ciphertext c^*.

The algorithm $\mathcal{D}$ models the hash functions H_1 and H_2 as random oracles. To be consistent amongst the adversary's queries, $\mathcal{D}$ maintains the following lists, where α refers to the session number:

List L_1 records entries of the form $(\alpha, X_1, ..., X_n, h_1)$ to maintain consistency in oracle H_1 queries, where $h_1 = H_1(X_1, ..., X_n)$.

List L_2 records entries of the form $(\alpha, \mathcal{U}, session_{ID}, N, h_2)$ to maintain consistency in oracle H_2 queries, where $h_2 = H_2(\mathcal{U} \parallel session_{ID} \parallel N)$.

List L_{AGKA} records entries of the form

$$(\alpha, \mathcal{U}, session_{ID}, ID_{\alpha_1}, \sigma_{\alpha_1}, ..., ID_{\alpha_n}, \sigma_{\alpha_n}, ID_{\alpha_0}, N, \sigma, c_{\alpha_1}, ..., c_{\alpha_n}, h_1, h_2, K)$$

to maintain consistency in Send, Execute and Reveal queries. This list is essentially a transcript of the α^{th} session of the protocol.

We denote the α^{th} entry of a list by $L(\alpha)$. Using these lists, the algorithm $\mathcal{D}$ responds to $\mathcal{A}$'s queries as follows:

Send(Π_U^α, M) Algorithm $\mathcal{D}$ handles Send queries in the following cases:

1. If M ="`init` $\parallel$ `leader`", so that the message is to initiate the protocol with U as the leader, we have the following three subcases:
   (a) If $U = U_d$ and $\alpha = \beta$, then $\mathcal{D}$ returns the signcrypted values $\langle \gamma_1, ..., \gamma_n \rangle$ in place of the leader's signcrypted contribution values $\langle c_{\alpha_1}, ..., c_{\alpha_n} \rangle$.
   (b) If $U \neq U_d$ and $\alpha = \beta$, then $\mathcal{D}$'s choice for the group leader was incorrect and the algorithm fails.
   (c) Otherwise, $\mathcal{D}$ chooses a random contribution $N \in \{0,1\}^*$ and responds with the ciphertexs $\langle c_{\alpha_1}, ..., c_{\alpha_n} \rangle$ obtained from Sign/Encrypt("$\mathcal{U} \parallel session_{ID} \parallel N$", ID_U, $\{\mathsf{ID}_{\alpha_1}, ..., \mathsf{ID}_{\alpha_n}\}$). The algorithm records ($\mathcal{U}$,$session_{ID}$,ID_U,N,c_{α_1},...,c_{α_n}) in $L_{AGKA}(\alpha)$.
2. If M ="`init` $\parallel$ `responder`", so that the message is to initiate the protocol with U as the responder, we have the following two subcases:
   (a) If $U = U_d$ and $\alpha = \beta$, then $\mathcal{D}$'s choice for the group leader was incorrect and the algorithm fails.
   (b) Otherwise, $\mathcal{D}$ runs the protocol normally as a responder. We note that the security game does not give $\mathcal{D}$ access to the Sign query of Section 4.1.2. However, we can still sign responder contributions due to

the $\mathcal{S}$-verifiability of the signcryption scheme. The algorithm queries Sign/Encrypt($session_{ID}$, ID_U, ID'_i) for some identity ID'_i to obtain a ciphertext c' and subsequently queries Decrypt/Verify(c', ID_U, ID'_i) to obtain $(session_{ID}, \sigma_U)$, where σ_U is a valid signature on $session_{ID}$ for the responder U. We note that there is no restriction on queries of this form in the IND-IBSC-CCA2 security game of Section 4.3.4. $\mathcal{D}$ returns the signature and records $(session_{ID}, ID_U, \sigma_U)$ in $L_{AGKA}(\alpha)$.

3. If M is not to initiate the protocol, then Π_U^α accepts the message M and responds as follows. If M represents the signed contribution of a responder, then $\mathcal{D}$ records (ID_U, σ_U) in $L_{AGKA}(\alpha)$. In addition, we have the following subcases:

   (a) If U is the group leader and M represents the last responder contribution σ_{α_i} for session α, then $\mathcal{D}$ records $(ID_{\alpha_i}, \sigma_{\alpha_i})$ in $L_{AGKA}(\alpha)$, verifies the signatures of all responders and accepts the session key if the verification holds. $\mathcal{D}$ outputs the outcome of acceptance.

   (b) If U is a responder and M represents the signcrypted contribution $(c_{\alpha_1}, ..., c_{\alpha_n})$ of the leader for session α, then $\mathcal{D}$ accepts the information, provided it is in the expected format. As long as $\alpha \neq \beta$, $\mathcal{D}$ obtains the valid message-signature pair ("$\mathcal{U} \parallel session_{ID} \parallel N$", σ) by querying Decrypt/Verify(c_{α_i}, ID_{α_i}, ID_U). Algorithm $\mathcal{D}$ verifies all responder signatures and the detachable signature, outputs the outcome of acceptance and records ($\mathcal{U}$, $session_{ID}$, ID_U, N, σ, c_{α_1}, ..., c_{α_n}) in $L_{AGKA}(\alpha)$.

Execute($U_{i_0}, U_{i_1}, ..., U_{i_n}$) $\mathcal{D}$ executes the protocol for the leader U_{i_0} and the responders $\mathcal{R} = \{U_{i_1}, ..., U_{i_n}\}$, according to the steps outlined in the Send query.

Reveal(Π_U^α) We first assume that oracle Π_U^α has accepted the session key and thus, Π_U^α has collected all responder contributions $\{X_{\alpha_1}, ..., X_{\alpha_n}\}$ and the leader contribution N. The session keys are given by the bitwise addition of the outputs h_1 and h_2 of the random oracles H_1 and H_2, respectively. If the key for session α has not yet been revealed, $\mathcal{D}$ obtains h_1 and h_2 from $L_{AGKA}(\alpha)$ if they exist, or queries H_1 and H_2 otherwise. $\mathcal{D}$ returns $K = h_1 \oplus h_2$ as the α^{th} session key

and adds K to $L_{AGKA}(\alpha)$. If the key for session α has been revealed before, $\mathcal{D}$ returns K from $L_{AGKA}(\alpha)$.

Corrupt(U) If U is the group leader, $\mathcal{D}$ responds with the private key obtained from the Extract query. Otherwise, $\mathcal{D}$ aborts the simulation.

Test(Π_U^α) If $\alpha = \beta$ and the group leader of the α^{th} session is U_d, then $\mathcal{D}$ returns a random string as the session key. Otherwise, the algorithm fails.

At some point during the game, $\mathcal{A}$ will output a guess for the value of b. Recall that $\mathcal{A}$ queries the hash functions H_1 and H_2 to compute the session key. Since these hash functions are modelled as random oracles, $\mathcal{A}$ has no advantage in guessing session keys that were not amongst these oracle queries. The algorithm $\mathcal{D}$ examines all queries $H_2(\mathcal{U} \parallel session_{ID} \parallel N)$ made by $\mathcal{A}$ to the oracle H_2, which $\mathcal{D}$ has recorded in the list L_2. If at any point, $\mathcal{D}$ finds a query with $N = m_0$, then it outputs $\theta = 0$ as its guess. Otherwise, $\mathcal{D}$ returns $\theta = 1$.

Suppose that $\mathcal{A}$ succeeds in distinguishing between the signcrypted ciphertexts with probability Pr [Distinguish]. The probability that the simulation does not fail (i.e. $\alpha = \beta$ and $U_\rho = U_d$) is $1/(n+1)\,q_{ex}$. Thus, the probability of success for $\mathcal{D}$ is given by $\mathrm{Adv}_{\mathcal{D},\mathcal{SC}}^{\mathsf{Distinguish}}(t) = \frac{1}{(n+1)q_{ex}} \Pr[\mathsf{Distinguish}]$, and since $\mathrm{Adv}_{\mathcal{D},\mathcal{SC}}^{\mathsf{Distinguish}}(t) \leq \mathrm{Adv}_{\mathcal{SC}}^{\mathsf{Distinguish}}(t)$, we have $\Pr[\mathsf{Distinguish}] \leq (n+1)\,q_{ex}\,\mathrm{Adv}_{\mathcal{SC}}^{\mathsf{Distinguish}}(t)$. ∎

6.6 Illustration of the COMPASS Framework

6.6.1 The YYHZ-COMPASS-AGKA Protocol

In Section 4.3.6, we presented an efficient multi-receiver variant (YYHZ-IBSC) of the Chen and Malone-Lee ID-based signcryption scheme (CML-IBSC) of Section 4.3.5. Although not formally proven, the CML-IBSC and YYHZ-IBSC schemes provide $\mathcal{S}$-verifiability using the Cha-Cheon signature scheme (ChCh-IBS) of Section 4.1.3 and forward security. In Section 6.2.3, we showed that the value X in the ChCh-IBS signature $\sigma_{\mathsf{ChCh}} = (X, Z)$ can also serve the purpose of the responder's key contribution. Furthermore, the ChCh-IBS scheme was shown to support efficient strong

batch verification in Section 4.2.2. As a result, the YYHZ-IBSC scheme is an optimal candidate for the COMPASS framework.

Using the **Setup** algorithm, the PKG generates the public system parameters

$$\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_{pub}, R, \theta, H_0, H_1, H_2, H_3, H_4, H_5 \rangle .$$

where $R \in_R \mathbb{G}_1^\times$, $\theta = e(R, P_{pub})$ and $\langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_{pub} \rangle$ are as defined in the YYHZ-IBSC scheme, and $\{H_0, H_1, H_2, H_3, H_4, H_5\}$ are the hash functions defined in Table 6.3.

$H_0 : \{0,1\}^* \rightarrow \mathbb{G}_1^\times$
$H_1 : \mathbb{G}_1 \times \{0,1\}^* \rightarrow \mathbb{Z}_q^\times$
$H_2 : \mathbb{G}_1^2 \times \{0,1\}^* \rightarrow \mathbb{Z}_q^\times$
$H_3 : \mathbb{G}_2 \rightarrow \{0,1\}^*$
$H_4 : \mathbb{G}_1^n \rightarrow \{0,1\}^*$
$H_5 : \{0,1\}^* \rightarrow \{0,1\}^*$

Table 6.3. Hash functions used in the YYHZ-COMPASS-AGKA protocol.

Using the **Setup** and **Extract** algorithms of the YYHZ-IBSC, each group member $U_i \in \mathcal{U}$ with identity ID_i is associated with the public/private key pair (Q_i, d_i). To begin group communication, the group leader makes public the session identifier $session_{ID} \in \{0,1\}^*$, which may be taken from some preexisting list. The YYHZ-COMPASS-AGKA protocol is executed in a single round as follows:

Round 1 (Responders)

Using the ChCh-IBS scheme, each responder $U_i \in \mathcal{R}$ signs the session identifier $session_{ID}$ by:

1. Choosing $r_i \in_R \mathbb{Z}_q^\times$,
2. Computing
   i) $X_i = r_i Q_i$,

ii) $h_i = H_1(X_i, \text{“}session_{ID}\text{”})$,

iii) $Z_i = (r_i + h_i)d_i$.

3. Broadcasting the signature $\sigma_i = (X_i, Z_i)$ to the rest of the group.

Round 1 (Group Leader)

Meanwhile, the group leader U_0 chooses a secret key contribution $N \in_R \{0,1\}^*$ and uses the YYHZ-IBSC scheme to signcrypt her contribution for the group of responders by

1. Choosing $r_0, k_0 \in_R \mathbb{Z}_q^\times$.

2. Computing

   i) $T = k_0 P$,

   ii) $X_0 = r_0 Q_0$,

   iii) $h_0 = H_2(T, X_0, \text{“}\mathsf{ID}_0 \parallel \mathcal{U} \parallel session_{ID} \parallel N\text{”})$,

   iv) $Z_0 = (r_0 + h_0)d_0$,

   v) $\omega = e(Z_0, P)$,

   vi) $y = (\text{“}\mathcal{U} \parallel session_{ID} \parallel N\text{”} \parallel \sigma_0) \oplus H_3(\omega)$, where $\sigma_0 = (X_0, Z_0)$,

   vii) $W = \theta^{k_0}\omega$,

   viii) $c_i = k_0(Q_i + R)$, for each responder $U_i \in \mathcal{R}$.

3. Broadcasting the ciphertext set $\langle y, T, W, c_1, c_2, ..., c_n \rangle$, along with some list indicating which ciphertext c_i corresponds to which responder U_i.

Key Computation

Upon receiving the ciphertext c_i from the group leader, each responder U_i recovers the group leader's contribution N and the detachable signature $\sigma_0 = (X_0, Z_0)$ by computing:

i) $\omega = W \cdot e(T, d_i) \cdot e(P_{pub}, c_i)^{-1}$ as in the YYHZ-IBSC scheme,

ii) "$\mathcal{U} \parallel session_{ID} \parallel N$" $\parallel \sigma_0 = y \oplus H_3(\omega)$.

Using the batch verification algorithm described in Section 4.2.2, each group member chooses a vector $(\delta_0, \delta_1, ..., \delta_n) \in_R \left(\mathbb{Z}_q^{\times}\right)^{n+1}$ and batch verifies the set of signatures $\{\sigma_0, \sigma_1, ..., \sigma_n\}$ by checking that

$$e(\sum_{i=0}^{n} \delta_i Z_i, Q) = e(\sum_{i=0}^{n} \delta_i \left(X_i + h_i Q_i\right), P_{pub}),$$

where $h_0 = H_2(T, X_0, \text{"}\mathsf{ID}_0 \parallel \mathcal{U} \parallel session_{ID} \parallel N\text{"})$, $h_i = H_1(X_i, \text{"}session_{ID}\text{"})$ for $1 \leq i \leq n$. If the verification holds, they compute the session key as

$$K = H_4(X_1, ..., X_n) \oplus H_5(\mathcal{U} \parallel session_{ID} \parallel N).$$

6.6.2 Efficiency of the YYHZ-COMPASS-AGKA Protocol

The communication costs for the resulting YYHZ-COMPASS-AGKA protocol are given in Section 6.4. We provide an accurate account of the computational costs in Table 6.4 below, in terms of pairing computations $\mathcal{P}$ and scalar multiplications $\mathcal{M}$. We compare the costs of the YYHZ-COMPASS-AGKA protocol with the LKKR-GKA protocol of Section 5.1.4, the CHL-ID-AGKA protocol of Section 5.2.2 and the KNKW-ID-AGKA protocol of Section 5.3.1. Similar to the analyses of previous chapters, we drop the protocol suffixes in our discussion.

To allow for a proper comparison of the AGKA protocols, we modify the costs of the LKKR, CHL and KNKW protocols given in Chapter 5, so that they reflect the costs required to provide implicit key authentication (see Section 2.2.1) and resilience to attack. Since the computational costs for the LKKR protocol do not include the costs associated with authentication, we include the costs incurred by substituting Joux's tripartite key exchange with Zhang *et al.*'s authenticated tripartite key exchange (ZLK) protocol from Section 4.1.5. To ease in our comparison, we assume that $n+1 = 3^h$ for some $h \in \mathbb{Z}^+$. That is, the LKKR key tree is a perfect tree of height h. We present the costs of the modified CHL protocol proposed in [CHL08], so that it is protected against the insider impersonation attacks discussed in Section 5.2.2. Since the protocol uses a variation of the H-IBS scheme from Section 4.1.4 to authenticate the first round contributions, we assume that the same scheme is used to authenticate

the second round contributions. Furthermore, the modified protocol requires the verification of the second round contribution of all members using signature screening. Finally, the computational costs given for the KNKW protocol do not include the costs associated with authentication of the leader's contribution, which is necessary for implicit key authentication. Since the protocol does not specify the use of a particular signature scheme, we include the computational costs for the ChCh-IBS variant scheme of Cheon *et al.* from [CKY04], which is used to authenticate responder contributions in the KNKW protocol. Our modified computational costs are given in parentheses in Table 6.4 and are included in the total computational costs for the protocols.

From Table 6.4, we see that the YYHZ-COMPASS protocol requires the least number of pairing computations when $n > 1$. Although the YYHZ-COMPASS protocol appears to be more computationally expensive than the other protocols in terms of multiplications $\mathcal{M}$, we note that these computations are induced by the strong authentication of the COMPASS framework. As discussed in Sections 5.2.4 and 5.3.5, the LKKR, CHL and KNKW protocols only provide partial authentication. In particular, the LKKR protocol only requires the distinguished representatives to sign the blinded keys of non-leaf nodes. The CHL protocol only requires a member to authenticate the contributions of adjacent group members. Similar to the CKA protocol, the KNKW protocol only requires the leader to authenticate responder contributions. Furthermore, the CHL and KNKW protocols only employ signature screening, rather than the strong batch verification of the YYHZ-COMPASS protocol.

Protocol	Computations (GKA / Authentication)			
	$\mathcal{P}$		$\mathcal{M}$	
	u_0	u_i	u_0	u_i
YYHZ–COMPASS-AGKA	$1 \,/\, 2$	$2 \,/\, 2$	$n+1 \,/\, 3n+5$	$- \,/\, 3n+5$
Total	$4n+3$		$3n^2+9n+6$	
LKKR-GKA (§5.1.4)	$hn+h \,/\, (4hn+4h)$		$\frac{3}{2}n \,/\, ((2h+3)n+2h)$	
Total	$(5hn+5h)$		$\left(\frac{(4h+9)}{2}n+2h\right)$	
CHL-ID-AGKA (§5.2.2)	$2 \,/\, 2+(2)$		$2 \,/\, 5+(n+3)$	
Total	$(6n+6)$		$(n^2+11n+10)$	
KNKW-ID-AGKA (§5.3.1)	$n \,/\, n+1$	$1 \,/\, (2)$	$3 \,/\, n+(2)$	$- \,/\, 3+(1)$
Total	$(5n+1)$		$(5n+5)$	

Table 6.4. Computational Costs of the YYHZ-COMPASS-AGKA Protocol.

6.7 Summary

In this chapter, we proposed a novel framework for constructing authenticated group key agreement protocols from signcryption schemes, which we call the COMPASS framework, and reduced the semantic security of the COMPASS-AGKA protocols to the security of the underlying signcryption schemes in the random oracle model. We presented a basic COMPASS framework and suggested several optimizations that can be achieved by choosing signcryption schemes that satisfy some desirable properties. We showed that the resulting COMPASS-AGKA protocols provide the optimal level of communication efficiency for GKA protocols and provide more contributory properties than each of the GKA classes considered in Chapter 5. Furthermore, while providing an illustration of the COMPASS framework using the YYHZ-IBSC scheme of Section 4.3.6, we produced an ID-AGKA that is more computationally efficient than all of the pairing-based GKA protocols from Chapter 5.

In the next section, we modify the COMPASS protocol obtained from the CML-IBSC scheme to produce an identity-based authenticated group key agreement protocol with perfect forward secrecy. In addition, we present protocols for auxiliary key agreement that still maintain perfect forward secrecy.

Chapter 7

A Forward Secure ID-AGKA Protocol

The COMPASS framework of the previous chapter produces efficient and secure computationally asymmetric authenticated group key agreement (AGKA) protocols from identity-based signcryption schemes. While the COMPASS-AGKA protocols provide many of the desirable contributory properties defined in Section 2.2.1, they do not provide perfect forward secrecy since the disclosure of the long-term private keys of all group members compromises previous session keys. However, for some signcryption schemes, we can modify the COMPASS-AGKA protocol to provide inherent perfect forward secrecy, at the expense of an additional round of communication.

In this chapter, we propose a novel identity-based authenticated group key agreement protocol (ID-AGKA) that is based on the COMPASS-AGKA protocol generated from the identity-based signcryption scheme of Chen and Malone-Lee (CML-IBSC) from Section 4.3.5. However, the protocol is more intricate than the resulting CML-COMPASS-AGKA protocol and depends heavily on our unique observation of the symmetry of the identity-based signature schemes of Cha and Cheon (ChCh-IBS) from Section 4.1.3 and Hess (H-IBS) from Section 4.1.4. As a nod to Cha-Cheon and Hess, we refer to the proposed protocol as the ChChH-ID-AGKA protocol. [1]

[1] Alternatively, one could refer to our protocol as the C_2H_3-ID-AGKA, or the *vinyl* protocol, as C_2H_3 is the chemical formula for the vinyl radical.

The ChChH-ID-AGKA protocol provides perfect forward secrecy in two rounds of communication, while preserving the communication efficiency and contributory benefits of the COMPASS framework. Furthermore, we propose efficient auxiliary key agreement (AKA) protocols that maintain the forward security of the protocol, without requiring members to store ephemeral keys, which is, to the best of our knowledge, the first group key agreement protocol to achieve this feat.

In Section 7.1, we prepare for the ChChH-ID-AGKA protcol by presenting the similarities in the ChCh-IBS and H-IBS schemes and discussing how the CML-IBSC scheme can be modified to provide perfect forward secrecy. In Section 7.2 we present the ChChH-ID-AGKA protocol and in Section 7.3, we show that it is a provably secure authenticated group key agreement protocol in the random oracle model. In Section 7.4, we discuss the concept of auxiliary key agreement in the ChChH-ID-AGKA protocol and propose efficient protocols to refresh the session key after members join or leave the multicast group. Furthermore, we show that the ChChH-ID-AGKA protocol maintains perfect forward secrecy when these auxiliary algorithms are employed. Finally, we compare the communication and computational efficiency of the ChChH-ID-AGKA protocol with the pairing-based GKA protocols from Chapter 5 in Section 7.5 and summarize the results of the chapter in Section 7.6.

7.1 Preparations for the ChChH-ID-AGKA Protocol

In this section, we discuss some preliminary issues in preparation for the ChChH-ID-AGKA protocol. In Section 7.1.1, we show that signatures generated using the ChCh-IBS scheme and a particular case of the H-IBS scheme may be verified using the same algorithm. This interesting result is at the heart of the ChChH-ID-AGKA protocol, as we will see in Section 7.2. In Section 7.1.2, we discuss why the COMPASS framework cannot provide perfect forward secrecy and show how some COMPASS-AGKA protocols can be modified to become forward secure.

7.1.1 Cha-Cheon and Hess Verification

In [Hes03, Thm 1], Hess suggests that the signing algorithm of the H-IBS scheme can be further optimized if the signer takes $P_1 = d_{\mathsf{ID}}$, producing the signature $\sigma = (R, Z)$, where $R = e(d_{\mathsf{ID}}, P)^r$ and $Z = (r + h_1)\, d_{\mathsf{ID}}$. As a result of this optimization, signatures of the ChCh-IBS and H-IBS schemes can be verified using the same verification algorithm, while still providing the signature unforgeability of the original schemes. We change the notation of the ChCh-IBS and H-IBS schemes to coincide with our presentation of the ChChH-ID-AGKA protocol.

Suppose the signer with identity ID and public/private key pair $(Q_{\mathsf{ID}}, d_{\mathsf{ID}})$ wants sign a message $m \in \mathcal{M}$.

ChCh-IBS To produce a ChCh-IBS signature $\sigma_{\mathsf{ChCh}} = (X, Z)$, the sender chooses $a \in_R \mathbb{Z}_q^\times$ and computes:

i) $X = aQ_{\mathsf{ID}}$,

ii) $h_1 = H_1(X, m)$ using $H_1 : \mathbb{G}_1 \times \{0,1\}^* \to \mathbb{Z}_q^\times$,

iii) $Z = (a + h_1) d_{\mathsf{ID}}$.

H-IBS To produce an H-IBS signature $\sigma_{\mathsf{H}} = (r, Z)$, the sender chooses $b \in_R \mathbb{Z}_q^\times$ and computes:

i) $r = e(d_{\mathsf{ID}}, P)^b$,

ii) $h_2 = H_2(r, m)$ using $H_2 : \mathbb{G}_2 \times \{0,1\}^* \to \mathbb{Z}_q^\times$,

iii) $Z = (b + h_2) d_{\mathsf{ID}}$.

In order to verify a ChCh-IBS signature $\sigma_{\mathsf{ChCh}} = (X_1, Z_1)$ for a sender ID_1 on a message m_1 and an H-IBS signature $\sigma_{\mathsf{H}} = (r_2, Z_2)$ for a sender ID_2 on a message m_2, the recipient computes $h_1 = H_1(X_1, m_1)$ and $h_2 = H_2(r_2, m_2)$, and checks that

$$e(Z_1 + Z_2, P) = e(X_1 + h_1 Q_1 + h_2 Q_2, P_{pub}) \cdot r_2,$$

where $Q_i = H_0(ID_i)$, $i = 1, 2$ is computed using the hash function $H_0 : \{0,1\}^* \rightarrow \mathbb{G}_1^\times$. We prove the consistency as follows:

$$\begin{aligned} e(X_1 + h_1Q_1 + h_2Q_2, P_{pub}) \cdot r_2 &= e(a_1Q_1 + h_1Q_1 + h_2Q_2, P_{pub}) \cdot e(d_2, P)^{b_2} \\ &= e(a_1d_1 + h_1d_1 + h_2d_2, P) \cdot e(b_2d_2, P) \\ &= e(a_1d_1 + h_1d_1 + h_2d_2 + b_2d_2, P) \\ &= e((a_1 + h_1)\, d_1 + (b_2 + h_2)\, d_2, P) \\ &= e(Z_1 + Z_2, P). \end{aligned}$$

As a result of this optimization, the signatures of the ChCh-IBS and H-IBS schemes can be efficiently batch verified together using the techniques discussed in Section 4.2.2. Indeed, suppose we want to verify a batch of ChCh-IBS signatures $(\mathsf{ID}_1, m_1, \sigma_{\mathsf{ChCh}_1})$, ..., $(\mathsf{ID}_k, m_k, \sigma_{\mathsf{ChCh}_k})$ where $\sigma_{\mathsf{ChCh}_i} = (X_i, Z_i)$, $1 \leq i \leq k$, and a batch of H-IBS variant signatures $(\mathsf{ID}_{k+1}, m_{k+1}, \sigma_{\mathsf{H}_{k+1}})$, ..., $(\mathsf{ID}_n, m_n, \sigma_{\mathsf{H}_n})$ where $\sigma_{\mathsf{H}_j} = (r_j, Z_j)$, $k+1 \leq j \leq n$. The recipient chooses a vector $(\delta_1, ..., \delta_n) \in_R \left(\mathbb{Z}_q^\times\right)^n$ and checks that

$$e(\sum_{i=1}^{n} \delta_i Z_i, P) = e(\sum_{i=1}^{k} \delta_i \left(X_i + h_iQ_i\right) + \sum_{j=k+1}^{n} \delta_j h_j Q_j \ , \ P_{pub}) \cdot \prod_{j=k+1}^{n} r_j^{\delta_i},$$

where $Q_i = H_0(ID_i)$, $1 \leq i \leq n$. The consistency follows from the two signature case above.

7.1.2 Providing Perfect Forward Secrecy

The secure transmission of the leader's secret contribution in the COMPASS framework relies on the concept of signcryption. Since the only secret value required for designcryption is a responder's long-term private key, the disclosure of any responder's private key will compromise the key of each session for which that responder was a participant. To allow for the disclosure of long-term private keys and provide

perfect forward secrecy, the signcryption process must incorporate the ephemeral public/private keys of the responders. Since the COMPASS-AGKA protocol runs in a single round of communication, it is not possible for the group leader to learn the ephemeral public keys of the responders before signcrypting her secret contribution. By including an additional round of communication, some signcryption schemes can be modified to include ephemeral values in the signcryption process, so that the corresponding COMPASS-AGKA protocol will provide perfect forward secrecy. To the best of our knowledge, there are no group key establishment protocols that provide perfect forward secrecy in a single round of communication.

As illustrated in the CML-IBSC scheme of Section 4.3.5 and the multi-receiver variant scheme of Yu *et al.* (YYHZ-IBSC) from Section 4.3.6, most sign-then-encrypt schemes encrypt the plaintext message using some shared "masking" value that can only be computed by the sender and the recipient. Thus many COMPASS-AGKA protocols work in a similar manner to the pairing-based CKA protocols of Section 5.3, where the leader masks her contribution for each individual responder using some shared masking value α. We simply modify this masking process so that it involves both the long-term and ephemeral public/private keys of the responder.

In the proposed ChChH-ID-AGKA protocol, we use a variation of the masking process from the CML-IBSC scheme. In the scheme, the sender A has the long-term public/private key pair (Q_A, d_A) and chooses the ephemeral public/private key pair $(X_A = r_A Q_A, r_A)$, while the recipient B has the long-term public/private key pair (Q_B, d_B). The shared masking value is computed as a function of these values using the pairing e, as

$$\alpha = e(d_A, Q_B)^{r_A} = e(sQ_A, Q_B)^{r_A} = e(r_A Q_A, sQ_B) = e(X_A, d_B).$$

In adapting this process for the ChChH-ID-AGKA protocol, we assume that the recipient also has an ephemeral public/private key pair $(X_B = r_B Q_B, r_B)$. We modify the computation of the masking value α to include this ephemeral key pair. The new

masking value α' is computed as

$$\alpha' = e(d_A, X_B)^{r_A} = e(sQ_A, r_BQ_B)^{r_A} = e(r_AQ_A, sQ_B)^{r_B} = e(X_A, d_B)^{r_B}.$$

Since the recipient needs his long-term private key d_B and the ephemeral private key r_B to compute the masking value, his long-term private key may be disclosed without compromising the security of the session. By using this masking procedure in the ChChH-ID-AGKA protocol, we provide perfect forward secrecy.

7.2 The ChChH-ID-AGKA Protocol

In this section, we present the proposed ChChH-ID-AGKA protocol. We discuss the initial setup, including a discussion of the identity-based public key infrastructure (ID-PKI) in Section 7.2.1, present the ChChH-ID-AGKA protocol in terms of initial key agreement in Section 7.2.2 and offer an analysis of the ChChH-ID-AGKA approach in Section 7.2.3. A list of the hash functions used in the ChChH-ID-AGKA protocol are given in Table 7.1.

$H_0 : \{0,1\}^* \rightarrow \mathbb{G}_1$
$H_1 : \mathbb{G}_2 \rightarrow \{0,1\}^*$
$H_2 : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{Z}_q^\times$
$H_3 : \mathbb{G}_1 \times \mathbb{G}_2^2 \rightarrow \mathbb{Z}_q^\times$

Table 7.1. Hash Functions in the ChChH-ID-AGKA Protocol.

7.2.1 ChChH-ID-AGKA Setup Phase

The proposed ChChH-ID-AGKA protocol requires an ID-PKI similar to that of the Boneh and Franklin identity-based encryption (IBE) scheme of Section 3.4. In particular, the ID-PKI is comprised of the following algorithms:

Setup Given a security parameter ℓ, the private key generator (PKG) generates the public system parameters

$$\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_{pub}, H_0, H_1, H_2, H_3 \rangle ,$$

where $\langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_{pub} \rangle$ are as defined in Section 3.4 and the hash functions are as defined in Table 7.1. As in other ID-PKIs, the PKG keeps the master secret key $s \in \mathbb{Z}_q^\times$ hidden.

Extract As in Section 3.4.

Suppose that a set of $n+1$ group members $\mathcal{U} = \{U_0, U_1..., U_n\}$ wish to agree upon a group session key. Member U_0 has the distinguished position of group leader. All other members U_i are responders and make up the set $\mathcal{R} = \{U_1, ..., U_n\}$. Using the ID-PKI described above, each group member U_i is associated with a public/private key pair (Q_i, d_i), issued by the PKG. We assume that each member U_i can obtain the public key Q_j associated with every other member $U_j \in \mathcal{U}$.

We provide authentication in our protocol using two different signature schemes. The responders use the H-IBS variant scheme from Section 7.1.1, while the leader uses the ChCh-IBS scheme. However, as shown in Section 7.1.1, we can efficiently batch verify signatures from either of these schemes. As a result, we can verify the leader's signature and the responder signatures at the same time, as in the COMPASS framework of Section 6.3. For clarity, we do not require the group members to sign the session identifier $session_{ID}$ and the multicast group $\mathcal{U}$, as in the COMPASS framework. However, in practical implementations, these values should be included in the authentication procedure to protect against insider impersonation and unknown key share attacks.

7.2.2 The ChChH-ID-AGKA Protocol

Before beginning the protocol, each group member $U_i \in \mathcal{U}$ chooses values $a_i, b_i \in_R \mathbb{Z}_q^\times$ and precomputes the following:

(i) $X_i = a_i Q_i$,

(ii) $r_i = e(b_i d_i, P)$,

(iii) $h_i = H_2(X_i, r_i)$ using the hash function H_2.

The ChChH-ID-AGKA protocol runs in the following two rounds of communication:

Round 1 Each responder $U_i \in \mathcal{R}$ computes $Z_i = (b_i + h_i)d_i$ and broadcasts $\langle X_i, \sigma_i \rangle$ to the group, where $\sigma_i = (r_i, Z_i)$ is an H-IBS variant signature.

Round 2 Upon receiving the $\langle X_i, \sigma_i \rangle$ from the responders, the group leader U_0 chooses a vector $(\delta_1, ..., \delta_n) \in_R \left(\mathbb{Z}_q^\times\right)^n$ and batch verifies the signatures $(\sigma_1, ..., \sigma_n)$ by checking that

$$e(\sum_{i=1}^{n} \delta_i Z_i, P) = e(\sum_{i=1}^{n} \delta_i h_i Q_i, P_{pub}) \cdot \prod_{i=1}^{n} r_i^{\delta_i},$$

as shown in Section 7.1.1. If the verification fails, the group leader broadcasts an "`abort`" message. If the equality holds, the group leader computes $Z_0 = (a_0 + h_0)d_0$ to obtain the ChCh-IBS signature $\sigma_0 = (X_0, Z_0)$ and masks her secret key contribution r_0 and the value Z_0 for each individual responder $U_i \in \mathcal{R}$ by computing [2]

$$\begin{aligned} t_i &= e(a_0 d_0, X_i) = e(Q_0, Q_i)^{s a_0 a_i} \\ y_i &= (r_0 \parallel Z_0) \oplus H_1(t_i). \end{aligned}$$

[2] Similar to the CML-IBSC and YYHZ-IBSC schemes, we require some invertible functions $f : \mathbb{G}_1 \to \{0,1\}^*$ and $g : \mathbb{G}_2 \to \{0,1\}^*$ to map Z_0 and r_0, respectively, to strings.

The group leader broadcasts $\langle X_0, y_1, ..., y_n \rangle$.

Key Computation Upon receiving X_0 and y_i from the group leader, each responder U_i recovers the leader's secret contribution r_0 and signature value Z_0 by computing

$$\begin{aligned} t_i &= e(X_0, d_i)^{a_i} = e(Q_0, Q_i)^{sa_0a_i} \\ (r_0 \parallel Z_0) &= y_i \oplus H_1(t_i). \end{aligned}$$

Each responder U_i then chooses a vector $(\delta'_0, ...\delta'_{i-1}, \delta'_{i+1}, ..., \delta'_n) \in_R (\mathbb{Z}_q^\times)^n$ and batch verifies all signatures $(\sigma_0, ...\sigma_{i-1}, \sigma_{i+1}, ..., \sigma_n)$ by checking that

$$e(\sum_{j \neq i} \delta'_j Z_j, P) = e(\delta'_0 (X_0 + h_0 Q_0) + \sum_{j \neq i} \delta'_j h_j Q_j, P_{pub}) \cdot \prod_{j \neq i} r_j^{\delta'_j}.$$

If the verification fails, the responder broadcasts an "`abort`" message. If the verification holds, they compute

$$K_{\mathcal{R}} = \prod_{i=1}^{n} r_i = \prod_{i=1}^{n} e(b_i d_i, P) = e(b_1 Q_1 + ... + b_n Q_n, P_{pub}).$$

and the final session key as

$$K = H_1(K_{\mathcal{R}}) \oplus H_1(r_0).$$

Remark 7.1 *When using the auxiliary key agreement protocols proposed in Section 7.4, all group members store the public part of the session key $K_{\mathcal{R}}$, and the private part of the session key r_0, for use in the next session. In addition, the group leader must store the individual contribution r_i of each responder $U_i \in \mathcal{R}$.*

ChChH-ID-AGKA

Round 1

$U_i \in \mathcal{R} : a_i, b_i \xleftarrow{r} \mathbb{Z}_q^{\times}$

$U_i \xrightarrow{B} \mathcal{U} : \langle X_i, \sigma_i = (r_i, Z_i) \rangle, \ 1 \leq i \leq n$

$$X_i = a_i Q_i$$
$$r_i = e(b_i d_i, P)$$
$$Z_i = (b_i + H_2(X_i, r_i))d_i$$

Round 2

U_0 batch verifies $(\sigma_1, ..., \sigma_n)$ by checking

$$e(\sum \delta_i Z_i, P) = e(\sum \delta_i H_2(X_i, r_i) Q_i, P_{pub}) \cdot \prod r_i^{\delta_i}$$

$U_0 : a_0, b_0 \xleftarrow{r} \mathbb{Z}_q^{\times}$

$U_0 \xrightarrow{B} \mathcal{U} : \langle X_0, y_1, ..., y_n \rangle$

$$X_0 = a_0 Q_0$$
$$r_0 = e(b_0 d_0, P)$$
$$Z_0 = (a_0 + H_2(X_0, r_0))d_0$$
$$y_i = (r_0 \parallel Z_0) \oplus H_1(e(a_0 d_0, X_i)), \ 1 \leq i \leq n$$

Key Computation

$U_i \in \mathcal{R}$ computes $y_i \oplus H_1(e(X_0, d_i)^{a_i}) = (r_0 \parallel Z_0)$

$U_i \in \mathcal{R}$ batch verifies $(\sigma_0, ...\sigma_{i-1}, \sigma_{i+1}, ..., \sigma_n)$ by checking

$e(\sum \delta_i' Z_i, P) = e(\delta_0' (X_0 + H_2(X_0, r_0) Q_0) + \sum \delta_i' H_2(X_i, r_i) Q_i, P_{pub}) \cdot \prod r_i^{\delta_i'}$

$$K = H_1(e(b_1 Q_1 + b_1 Q_1 + ... + b_n Q_n, P_{pub})) \oplus H_1(r_0)$$

Table 7.2. The ChChH-ID-AGKA Protocol

7.2.3 Analysis of ChChH-ID-AGKA Protocol

The beauty of the ChChH-ID-AGKA protocol is a direct result of the symmetric nature of the ChCh-IBS and H-IBS signature schemes. In the protocol, all group members precompute the same pair of values (X_i, r_i). However, the group members employ these values differently, depending on their role (i.e. leader or responder) in the group. For a responder $U_i \in \mathcal{R}$, the value X_i is used *only* in the masking of the leader's contribution in the second round of the protocol, while the value r_i is used as a part of the H-IBS variant signature $\sigma_i = (r_i, Z_i)$ *and* as the responder's key contribution. Since the responders' key contributions are publicly broadcasted in the computationally asymmetric model, this does not jeopardize the security of our protocol. A similar optimization is used in the COMPASS framework, as discussed in Section 6.2.3. For the group leader U_0, the value X_0 is used as a part of the ChCh-IBS signature $\sigma_0 = (X_0, Z_0)$ *and* in the unmasking process of the key computation phase, while the value r_0 is used *only* as the group leader's secret key contribution.

In the COMPASS framework, the responders and the group leader all send their contributions at the same time, ensuring that no member can predetermine or control the value of the final session key. This is the notion of no key control or key freshness discussed in Section 2.1.3. Although the ChChH-ID-AGKA protocol allows the group leader to collect the public key contributions of the responders before sending her secret key contribution, the session key computation formula ensures that she cannot control the value of the final session key. Indeed, this would require the cryptographic hash function H_1 to be invertible. Thus the ChChH-ID-AGKA protocol provides the no key control property.

Finally, we note that, since the ChChH-ID-AGKA protocol uses strong batch verification of signatures and computes the session key using only authenticated key contributions, we still provide the nice contributory properties of group integrity, verifiable contributiveness and key integrity, as in the COMPASS-AGKA protocols. The proof of these claims follows from the proofs for the COMPASS-AGKA protocol in

Section 6.5.2.

A summary of the group communication flows for the ChChH-ID-AGKA protocol can be found in Table 7.2.

7.3 Security Proof

We claim that the proposed ChChH-ID-AGKA protocol is a secure authenticated group key agreement protocol, as defined in Definition 2.2 of Section 2.2.2. The validity of the protocol (i.e. that members accept the same key in the presence of a passive adversary) is straightforward. It remains to show that the advantage of a probabilistic polynomial time (PPT) active adversary is negligible.

We prove the semantic security of the ChChH-ID-AGKA scheme by showing that, in the course of breaking a simulation of the protocol, the adversary must either produce a forgery for the ChCh-IBS or H-IBS signature schemes, or solve the decisional bilinear Diffie-Hellman (DBDH) problem, defined in Section 3.1. We use the conventional security model for authenticated group key agreement, as given in Section 2.2.2. Since we prove the security against an active adversary, we allow $\mathcal{A}$ to issue Send, Execute, Reveal, Corrupt and Test queries. Since our protocol is identity-based, we assume that the Corrupt query performs the same tasks as the Extract query, and allows the adversary to obtain the long-term identity-based private keys of users that are not in the multicast group.

When reducing the security of the protocol to the existential unforgeability of the signature schemes, we play the role of the forger in the EUF-CMA security game of Section 4.1.2 and use the adversary to gain a non-negligible advantage in the game. As in the security proof for the COMPASS framework in Section 6.5.2, we model the adversary's queries by issuing our own queries to the oracles of the EUF-CMA game and we allow the adversary to fool the responders into sharing a session key with her by forging the leader's signature for the ChCh-IBS scheme. However, in the ChChH-

ID-AGKA protocol, the adversary also gains an advantage by forging a responder's signature for the H-IBS scheme. Indeed, since the leader uses the authenticated responder values X_i when computing the masked values t_i, the adversary gains an advantage in the protocol by forging the signature for the X_i values.

When reducing the security of the protocol to the DBDH problem, we follow the approach of Katz and Yung from [KY03]. We note that this is a common approach, used to prove the security of both classical and pairing-based variations of GKA protocols, including the BD-based protocols of [KY03, CHL04] and the computationally asymmetric protocols of [KNKW05, NLKW05, ZSM06, CSCW07, HLL07]. We assume that the adversary obtains the joint distribution (T, K) from an Execute query, where T refers to the transcript of the execution of the protocol and K refers to the corresponding session key. We consider the following two distributions of the protocol: one which results from an honest execution of the protocol, which we call Real; and another in which the masked leader contribution values $\{y_i\}$ are uniformly distributed in $\{0,1\}^*$ and the session key K is independent from the protocol transcript T, which we call Rand. We show that if the adversary has a non-negligible advantage in distinguishing between the distributions Real and Rand, which in turn relates to her ability to distinguish between the Test keys in the security game, we can construct a distinguishing algorithm $\mathcal{D}$ that solves the DBDH problem with non-negligible advantage $\mathrm{Adv}^{DBDH}_{\mathbb{G}_1,\mathbb{G}_2,e}$, defined by

$$\mathrm{Adv}^{DBDH}_{\mathbb{G}_1,\mathbb{G}_2,e} = \left| \begin{array}{l} \Pr\left[\mathcal{D}(aP, bP, cP, e(P,P)^{abc}) = 1 \mid P \leftarrow \mathbb{G}_1;\ a,b,c \leftarrow \mathbb{Z}_q^\times\right] \\ -\Pr\left[\mathcal{D}(aP, bP, cP, e(P,P)^{d}) = 1 \mid P \leftarrow \mathbb{G}_1;\ a,b,c,d \leftarrow \mathbb{Z}_q^\times\right] \end{array} \right|.$$

Let $\mathrm{Adv}_{\mathsf{ChChH-ID-AGKA}}(t, q_{ex})$ denote the maximum advantage of any adversary attacking the ChChH-ID-AGKA protocol in running time t and making q_{ex} Execute queries. We suppose that the hash functions H_0, H_1 and H_2 are random oracles.

Theorem 2 *The* ChChH-ID-AGKA *protocol is a secure AGKA protocol achieving per-*

fect forward secrecy. Specifically,

$$\mathrm{Adv}_{\mathsf{ChChH-ID-AGKA}}(t, q_{ex}) \leq 2q_{ex} \cdot \mathrm{Adv}^{DBDH}_{\mathbb{G}_1,\mathbb{G}_{2,e}}(t') + (n+1)\,\mathrm{Adv}^{\mathsf{Forge}}_{\mathsf{ChCh}}(t) + (n+1)\,\mathrm{Adv}^{\mathsf{Forge}}_{\mathsf{H}}(t),$$

where $\mathrm{Adv}^{\mathsf{Forge}}_{\mathsf{ChCh}}(t)$ *is the maximum advantage of any forger* $\mathcal{F}_{\mathsf{ChCh}}$ *of the Cha-Cheon signature scheme (*ChCh-IBS*) running in time* t*,* $\mathrm{Adv}^{\mathsf{Forge}}_{\mathsf{H}}(t)$ *is the maximum advantage of any forger* $\mathcal{F}_{\mathsf{H}}$ *of the Hess signature scheme (*H-IBS*) running in time* t *and* $\mathrm{Adv}^{DBDH}_{\mathbb{G}_1,\mathbb{G}_{2,e}}(t')$ *is the maximum advantage of any distinguishing algorithm* $\mathcal{D}$ *running in time* $t' = t + 2nt_{\mathcal{M}} + (2n-1)t_{\mathcal{E}} + t_{\mathcal{P}}$*, where* $t_{\mathcal{M}}$ *represents the time to compute a scalar multiplication in* $\mathbb{G}_1$*,* $t_{\mathcal{E}}$ *represents the time to compute an exponentiation in* $\mathbb{G}_2$ *and* $t_{\mathcal{P}}$ *represents the time to compute a pairing operation.*

Suppose $\mathcal{A}$ is an adversary with non-negligible advantage $\mathrm{Adv}_{\mathsf{ChChH-ID-AGKA}}$ in attacking the ChChH-ID-AGKA protocol. Following the approach of Boyd and González Nieto from [BN03] and our security proof for the COMPASS-AGKA protocol, we divide our proof into three separate cases. We assume that $\mathcal{A}$ gains her advantage by: (1) forging the leader signature for the ChCh-IBS scheme, (2) forging a responder signature for the H-IBS scheme or (3) breaking the protocol without altering transcripts. In cases (1) and (2), we use the adversary A to construct forging algorithms $\mathcal{F}_{\mathsf{ChCh}}$ and $\mathcal{F}_{\mathsf{H}}$ for the respective signature schemes. In case (3), we use the adversary to build a distinguishing algorithm $\mathcal{D}$ that has non-negligible advantage in solving the DBDH problem.

For each proof, we allow $\mathcal{A}$ to choose the group leader U_0, the subgroup of responders $\mathcal{R} = \{U_1, ..., U_n\}$ and the responder subgroup size n.

Existential Forgery for the Cha-Cheon or Hess Signature Schemes

We follow the approach used in Section 6.5.2 to reduce the security of the COMPASS-AGKA protocol to the existential unforgeability of the underlying signcryption scheme. If the adversary $\mathcal{A}$ gains her advantage by forging the signed contribution of the group

leader, then we use $\mathcal{A}$ to construct a forger $\mathcal{F}_{\mathsf{ChCh}}$ for the ChCh-IBS scheme. Similarly, if $\mathcal{A}$ gains her advantage by forging the signed contribution of a responder, we use $\mathcal{A}$ to construct a forger $\mathcal{F}_{\mathsf{H}}$ for the H-IBS scheme. In either case, the task of the forger $\mathcal{F}$ in the EUF-CMA security game of Section 4.1.2 is to produce an existential forgery of a signature for some user U_{ID^*} with identity ID^* and public key Q_{ID^*}. The forger may make the following queries:

- $\mathsf{Extract}(\mathsf{ID}_i)$ to determine the long-term private key d_i for any user with identity $\mathsf{ID}_i \neq \mathsf{ID}^*$.

- $\mathsf{Sign}(m, \mathsf{ID})$ to produce the signature for sender ID on the message m.

We assume that the forgery for a message m^* that was not the result of query of the form $\mathsf{Sign}(m^*,\mathsf{ID}^*)$. Using the EUF-CMA security game, we prove the following results.

Lemma 7.1 *Let* Forge_1 *be the event that* $\mathcal{A}$ *outputs a valid forgery for the* ChCh-IBS *scheme. Then*

$$\Pr[\mathsf{Forge}_1] \leq (n+1)\,\mathrm{Adv}^{\mathsf{Forge}}_{\mathsf{ChCh}}(t).$$

Proof: Suppose the adversary $\mathcal{A}$ gains her advantage by forging the ChCh-IBS signature of the group leader $U_\rho \in \mathcal{U}$. The forger $\mathcal{F}_{\mathsf{ChCh}}$ chooses a random $U_d \in \mathcal{U}$ with identity ID_d as its guess for the group leader and the identity for which he will forge a ChCh-IBS signature in the EUF-CMA security game (i.e. $\mathsf{ID}_d = \mathsf{ID}^*$). For all users with identity different from ID_d, $\mathcal{F}_{\mathsf{ChCh}}$ honestly generates the public/private key pair using the $\mathsf{Extract}_{\mathsf{ChCh}}$ query. $\mathcal{F}_{\mathsf{ChCh}}$ then simulates the oracle queries from $\mathcal{A}$ in the usual way. If $\mathcal{A}$ issues the $\mathsf{Corrupt}(\mathsf{ID}_d)$ query, then $\mathcal{F}_{\mathsf{ChCh}}$ aborts the simulation. Otherwise, if $\mathcal{A}$ generates a new and valid message pair $(m, \sigma = (X, Z))$ for the identity ID_d, resulting in the event Forge, $\mathcal{F}$ returns the valid forgery. The probability of success for $\mathcal{F}_{\mathsf{ChCh}}$ is given by $\mathrm{Adv}^{\mathsf{Forge}}_{\mathcal{F},\mathsf{ChCh}}(t) = \frac{1}{n+1}\Pr[\mathsf{Forge}_1]$, and since $\mathrm{Adv}^{\mathsf{Forge}}_{\mathcal{F},\mathsf{ChCh}}(t) \leq \mathrm{Adv}^{\mathsf{Forge}}_{\mathsf{ChCh}}(t)$, we have $\Pr[\mathsf{Forge}_1] \leq (n+1)\,\mathrm{Adv}^{\mathsf{Forge}}_{\mathsf{ChCh}}(t)$. ∎

Lemma 7.2 *Let* Forge_2 *be the event that* $\mathcal{A}$ *outputs a valid forgery for the* H-IBS *scheme. Then*

$$\Pr[\mathsf{Forge}_2] \leq (n+1)\,\mathrm{Adv}_{\mathsf{H}}^{\mathsf{Forge}}(t).$$

Proof: The proof follows from the result of Lemma 7.1. The only difference in this case is that the forger $\mathcal{F}_{\mathsf{H}}$ chooses a random $U_d \in \mathcal{U}$ with identity ID_d as its guess for the responder U_ρ. The probability of success is determined by the probability that U_d was not the group leader of the session and was indeed the responder for which A produced a forgery, which is given by

$$\mathrm{Adv}_{\mathcal{F},\mathsf{H}}^{\mathsf{Forge}}(t) = \left(\frac{n}{n+1}\right)\left(\frac{1}{n}\right)\Pr[\mathsf{Forge}_2] = \frac{1}{n+1}\Pr[\mathsf{Forge}_2].$$

Thus, following the approach in Lemma 7.1, we have $\Pr[\mathsf{Forge}_2] \leq (n+1)\,\mathrm{Adv}_{\mathsf{H}}^{\mathsf{Forge}}(t)$. ∎

Indistinguishability Attack on the ChChH-ID-AGKA Protocol

Suppose that the adversary $\mathcal{A}$ attacks the ChChH-ID-AGKA protocol without altering authentication transcripts and makes q_{ex} Execute queries. We prove that if $\mathcal{A}$ attacks the protocol with non-negligible advantage $\epsilon(t)$ (i.e. $\mathcal{A}$ correctly guesses the value of the bit b with probability $1/2 + \epsilon$), we can construct a PPT algorithm $\mathcal{D}$ that can solve the DBDH problem in $(\mathbb{G}_1, \mathbb{G}_2, e)$ with probability ϵ/q_{ex}.

We first review the system parameters and the values that determine the transcript and the final session key of the protocol. In the initial setup of the protocol,

we have

$$\mathsf{params} = \left\{ \begin{array}{c} (\mathbb{G}_1, \mathbb{G}_2, e) \leftarrow \mathcal{G}_{BDH}(1^\ell);\ P \leftarrow \mathbb{G}_1^\times;\ s \leftarrow \mathbb{Z}_q^\times;\ P_{pub} = sP; \\ Q_0, Q_1, ..., Q_n \leftarrow \mathbb{G}_1^\times;\ d_0 = sQ_0, ..., d_n = sQ_n; \end{array} \right\},$$

where $(\mathbb{G}_1, \mathbb{G}_2, e, P, P_{pub})$ are the public system parameters, which are used in the computation of the public/private key sets $\{(Q_0, d_0), ..., (Q_n, d_n)\}$. In an execution of the protocol, the following values are computed

$$\left\{ (T, K) \left| \begin{array}{l} a_0, a_1, \dots, a_n, b_0, b_1, \dots, b_n \leftarrow \mathbb{Z}_q^\times; \\ X_0 = a_0 Q_0, ..., X_n = a_n Q_n; \\ r_0 = e(Q_0, P)^{b_0 s},\ r_1 = e(Q_1, P)^{b_1 s},\ \dots, r_n = e(Q_n, P)^{b_n s}; \\ Z_0 = (a_0 + h_0) d_0, Z_1 = (b_1 + h_1) d_1, \dots, Z_n = (b_n + h_n) d_n; \\ h_0 = H_2(X_0, r_0), \dots, h_n = H_2(X_n, r_n); \\ t_1 = e(Q_0, Q_1)^{a_0 a_1 s}, \dots, t_n = e(Q_0, Q_n)^{a_0 a_n s}; \\ \alpha_1 = H_1(t_1), ..., \alpha_n = H_1(t_n); \\ y_1 = (r_0 \parallel Z_0) \oplus \alpha_1, \dots, y_n = (r_0 \parallel Z_0) \oplus \alpha_n. \end{array} \right. \right\},$$

and determine the protocol transcript, given by $T = (X_0, X_1, ..., X_n, r_1, ..., r_n, Z_1, ..., Z_n, y_1, ..., y_n)$, and session key, given by $K = H_1(K_{\mathcal{R}}) \oplus H_1(r_0)$, where $K_{\mathcal{R}} = \prod_{i=1}^{n} r_i$.

We model perfect forward secrecy by allowing the adversary $\mathcal{A}$ to recover the long-term private key d_i for all users $U_i \in \mathcal{U}$ by issuing multiple $\mathsf{Corrupt}(U_i)$ queries. Recall that in this case, we assume the adversary does not gain an advantage by forging authentication transcripts. The values $(Z_1, ...Z_n)$ and $(r_1, ..., r_n)$ from T are only used for authentication purposes and do not help $\mathcal{A}$ to compute the masking values $\{t_i\}$ and recover the leader's contribution r_0. Consequently, we may omit the authentication values from the transcript, to obtain the reduced transcript $T^* = (X_0, ..., X_n, y_1, ..., y_n)$. Furthermore, we note that the public part of the session key $K_{\mathcal{R}}$ is publicly available to the adversary and thus has no effect on the security of the session. Thus the adversary's advantage in distinguishing between keys from the

Test query is equivalent to her advantage in distinguishing between the leader's secret contribution r_0 and a random value from $\mathbb{G}_2^\times$. We simplify our proof by reducing the session key K to the portion of the session key computed using the leader's secret contribution, $K^* = H_1(r_0)$, in our distribution. In this case, our only concern is that the adversary recovers the r_0 value and not that it can be properly authenticated. Thus we choose a random value from $\mathbb{G}_1^\times$ to represent the leader's Z_0 value.

In a real execution of the protocol, the distribution of our reduced transcript and session key are given by the following:

$$\mathsf{Real} = \left\{ (T^*, K^*) \;\middle|\; \begin{array}{l} a_0, a_1, a_2, \ldots, a_n \leftarrow \mathbb{Z}_q^\times;\ Z_0 \leftarrow \mathbb{G}_1^\times;\ r_0 \leftarrow \mathbb{G}_2^\times; \\ X_0 = a_0 Q_0, X_1 = a_1 Q_1, X_2 = a_2 Q_2, \ldots, X_n = a_n Q_n; \\ t_1 = e(Q_0, Q_1)^{a_0 a_1 s}, \ldots, t_n = e(Q_0, Q_n)^{a_0 a_n s}; \\ \alpha_1 = H_1(t_1), \ldots, \alpha_n = H_1(t_n); \\ y_1 = (r_0 \parallel Z_0) \oplus \alpha_1, \ldots, y_n = (r_0 \parallel Z_0) \oplus \alpha_n. \end{array} \right\}$$

where $T^* = (X_0, \ldots, X_n, y_1, \ldots, y_n)$ and $K^* = H_1(r_0)$.

We define the distribution Rand as follows: the values $\{a_0, a_1, a_2, \ldots, a_n\}$ are selected at random from $\mathbb{Z}_q^\times$ and used along with the public keys $\{Q_i\}$ to compute the X_i values, exactly as in Real. However, the masking values $\{\alpha_i\}$ are chosen uniformly at random from $\{0,1\}^*$, rather than computed as the hash of the t_i values, as in Real. Thus, we obtain the following distribution:

$$\mathsf{Rand} = \left\{ (T^*, K^*) \;\middle|\; \begin{array}{l} a_0, a_1, a_2, \ldots, a_n \leftarrow \mathbb{Z}_q^\times;\ Z_0 \leftarrow \mathbb{G}_1^\times;\ r_0 \leftarrow \mathbb{G}_2^\times; \\ \alpha_1, \ldots, \alpha_n \leftarrow \{0,1\}^*; \\ X_0 = a_0 Q_0, X_1 = a_1 Q_1, X_2 = a_2 Q_2, \ldots, X_n = a_n Q_n; \\ y_1 = (r_0 \parallel Z_0) \oplus \alpha_1, \ldots, y_n = (r_0 \parallel Z_0) \oplus \alpha_n. \end{array} \right\},$$

where T^* and K^* are as defined in Real.

Lemma 7.3 *For an algorithm $\mathcal{A}$ running in time t, we have*

$$\begin{aligned} &|\Pr[\mathcal{A}(T^*, K^*) = 1 \mid (T^*, K^*) \leftarrow \mathsf{Real}] - \Pr[\mathcal{A}(T^*, K^*) = 1 \mid (T^*, K^*) \leftarrow \mathsf{Rand}]| \\ \leq\ & \mathrm{Adv}^{DBDH}_{\mathbb{G}_1, \mathbb{G}_2, e}(t + 2nt_{\mathcal{M}} + (2n-1)t_{\mathcal{E}} + t_{\mathcal{P}}), \end{aligned}$$

where $t_{\mathcal{M}}$ represents the time to compute a scalar multiplication in $\mathbb{G}_1$, $t_{\mathcal{E}}$ represents the time to compute an exponentiation in $\mathbb{G}_2$ and $t_{\mathcal{P}}$ represents the time to compute a pairing operation.

Proof: Suppose an adversary $\mathcal{A}$ can distinguish between the distributions Real and Rand with non-negligible probability. Since the hash function H_1 is modelled as a random oracle and the only difference in the two distributions is in the computation of α_i, the adversary must determine at least one of the t_i from Real in order to distinguish the corresponding α_i value from Rand. We claim that $\mathcal{A}$ can be used to construct an algorithm with non-negligible advantage in solving the DBDH problem. We achieve this task by embedding the given DBDH protocol into a simulation of the protocol.

Let $\mathcal{D}$ be a distinguishing algorithm with input tuple $(aP, bP, cP, e(P,P)^d)$, where P is a fixed generator of $\mathbb{G}_1$. Recall that the task of the algorithm is to determine whether $(aP, bP, cP, e(P,P)^d)$ is a BDH tuple (i.e. $d = abc$) or a random value. To begin the simulation of the protocol, algorithm $\mathcal{D}$ outputs the system parameters

$$\mathsf{params} = \left\{(\mathbb{G}_1, \mathbb{G}_2, e) \leftarrow \mathcal{G}_{BDH}(1^\ell);\ P \leftarrow \mathbb{G}_1^\times;\ s \leftarrow \mathbb{Z}_q^\times;\ P_{pub} = cP;\right\},$$

so that the PKG's master secret key is the value c and the master public key is the DBDH value cP. Using these public parameters, $\mathcal{D}$ simulates the adversary's H_0 and Corrupt queries in the following manner. We assume that A makes the H_0 query before Corrupt queries.

Simulator $H_0(\mathsf{ID}_U)$ $\mathcal{D}$ maintains a list L_0 with entries of the form $(\mathsf{ID}_U, Q_U, d_U, l_U)$, which is initially empty. $\mathcal{D}$ interacts with $\mathcal{A}$ as follows:

- Choose some l_U uniformly at random from $\mathbb{Z}_q^\times$;
- Compute $Q_U = l_U P$ and $d_U = l_U P_{pub} = l_U cP$;
- Store $(\mathsf{ID}_U, Q_U, d_U, l_U)$ in L_0 and return Q_U.

Simulator $\mathsf{Corrupt}(\mathsf{ID}_U)$ Search the list L_0 for the entry $(\mathsf{ID}_U, Q_U, d_U, l_U)$ corresponding to ID_U and return d_U.

As a result, the algorithm produces the public/private key pairs as

$$\{(Q_0 = l_0 P, d_0 = cl_0 P), (Q_1 = l_1 P, d_1 = cl_1 P), ..., (Q_n = l_n P, d_n = cl_n P)\}.$$

The algorithm D embeds the remainder of the DBDH problem (i.e. the input values aP, bP and $e(P,P)^d$) in the protocol simulation to generate (T^*, K^*). The protocol distribution Simulation is defined in such a way that it is indistinguishable from the distribution Real if $(aP, bP, cP, e(P,P)^d)$ is a BDH tuple, and indistinguishable from the distribution Rand otherwise. The distribution is defined as follows:

$$\mathsf{Simulation} = \left\{ (T^*, K^*) \;\middle|\; \begin{array}{l} \beta_2, \dots, \beta_n, \gamma_2, \dots, \gamma_n \leftarrow \mathbb{Z}_q^\times;\ Z_0 \leftarrow \mathbb{G}_1^\times;\ r_0 \leftarrow \mathbb{G}_2^\times; \\ a_2 = a\beta_2 + \gamma_2, ..., a_n = a\beta_n + \gamma_n; \\ X_0 = l_0 bP,\ X_1 = l_1 aP,\ X_2 = a_2 l_2 P, ..., X_n = a_n l_n P; \\ t_1 = e(Q_0, Q_1)^d, t_2 = e(Q_0, Q_2)^{d\beta_2 + bc\gamma_2}, \\ \quad ..., t_n = e(Q_0, Q_n)^{d\beta_n + bc\gamma_n}; \\ \alpha_1 = H_1(t_1), ..., \alpha_n = H_1(t_n); \\ y_1 = (r_0 \parallel Z_0) \oplus \alpha_1, \dots, y_n = (r_0 \parallel Z_0) \oplus \alpha_n. \end{array} \right\},$$

where T^* and K^* are as defined in the distributions above. In the distribution, the leader's secret value a_0 is replaced with b, the responder U_1's secret value a_1 is replaced with a and the value a_i for each remaining responder is computed as $a\beta_i + \gamma_i$. We note that these values don't need to be explicitly computed in the distribution. We need

only be able to compute the X_i and t_i values in the distribution. We compute the X_i values as $X_0 = l_0(bP)$, $X_1 = l_1(aP)$ and $X_i = \beta_i l_i(aP) + \gamma_i l_i P$ for $2 \leq i \leq n$ and the t_i values as $t_1 = e(P,P)^{dl_0 l_1}$ and $t_i = e(P,P)^{d\beta_i l_0 l_i} \cdot e(bP, cP)^{\gamma_i l_0 l_i}$, for $2 \leq i \leq n$.

We first consider the case where $(aP, bP, cP, e(P,P)^d)$ is a BDH tuple (i.e. $d = abc$). Then we have,

$$t_1 = e(P,P)^{dl_0 l_1} = e(P,P)^{abcl_0 l_1} = e(Q_0, Q_1)^{abc}$$

and

$$t_i = e(Q_0, Q_i)^{d\beta_i + bc\gamma_i} = e(Q_0, Q_i)^{abc\beta_i + bc\gamma_i} = e(Q_0, Q_i)^{bc(a\beta_i + \gamma_i)} = e(Q_0, Q_i)^{bca_i},$$

for all $2 \leq i \leq n$. Since the t_i values, and thus the α_i values, are computed exactly as in Real, we have Simulation $\equiv$ Real from $\mathcal{A}$'s point of view. Thus we can conclude that

$$\begin{aligned} & \Pr\left[\mathcal{D}(aP, bP, cP, e(P,P)^{abc}) = 1 \mid a, b, c \leftarrow \mathbb{Z}_q^\times\right] \qquad (7.3.1) \\ = \; & \Pr\left[\mathcal{A}(T^*, K^*) = 1 \mid (T^*, K^*) \leftarrow \mathsf{Real}\right]. \end{aligned}$$

Now we consider the case where $(aP, bP, cP, e(P,P)^d)$ is a random tuple. The values X_0 and X_1 are computed exactly as in Rand and it is clear from Simulation that the remaining $\{X_i\}$ values are uniformly and independently distributed in $\mathbb{G}_1$. It remains to show that the t_i values are uniformly distributed and independent from the X_i values and from each other, so that the corresponding α_i values are uniformly and independently distributed in $\{0,1\}^*$, as in Rand. It is clear that this holds for t_1, since $d \in_R \mathbb{Z}_q^\times$. Following the approach in [KY03], we prove this for the remaining t_i values using a representative, say t_2, as an example. Since β_2 and γ_2 are only used

in the computation of X_2 and t_2, the joint distribution of these values is given by

$$\begin{aligned} \log_P X_2 &= a\beta_2 + \gamma_2 \\ \log_E t_2 &= d\beta_2 + bc\gamma_2, \end{aligned}$$

where $E = e(P,P)$ and β_2, γ_2 are uniformly and independently distributed in $\mathbb{Z}_q^\times$. Since $d \neq abc$, the above equations are linearly independent and thus t_2 is uniformly distributed in $\mathbb{G}_2$ and independent from X_2 and all other values. It follows that the t_i values are uniformly and independently distributed in $\mathbb{G}_2$ and, consequently, the α_i values are uniformly and independently distributed in $\{0,1\}^*$. Thus from $\mathcal{A}$'s point of view, we have Simulation $\equiv$ Rand and we can conclude that

$$\begin{aligned} & \Pr\left[\mathcal{D}(aP, bP, cP, e(P,P)^d) = 1 \mid a,b,c,d \leftarrow \mathbb{Z}_q^\times\right] \qquad (7.3.2) \\ = \; & \Pr\left[\mathcal{A}(T^*, K^*) = 1 \mid (T^*, K^*) \leftarrow \mathsf{Rand}\right]. \end{aligned}$$

The running time of the algorithm $\mathcal{D}$ is the running time of the adversary $\mathcal{A}$ plus the time required for the additional $2n$ scalar multiplications in $\mathbb{G}_1$, $(2n-1)$ exponentiations in $\mathbb{G}_2$ and a single pairing operation to be computed in Simulation. The proof of the lemma follows simply from Equations (7.3.1) and (7.3.2) above and the definition of $\mathrm{Adv}^{DBDH}_{\mathbb{G}_1,\mathbb{G}_2,e}(t)$. ∎

Lemma 7.4 *For any (computationally unbounded) adversary $\mathcal{A}$, we have:*

$$\Pr\left[\mathcal{A}(T^*, K_b^*) = b \mid (T^*, K_1^*) \leftarrow \mathsf{Rand};\ K_0^* \leftarrow \{0,1\}^*;\ b \leftarrow \{0,1\}\right] = 1/2.$$

Proof: We show that an adversary has no advantage in distinguishing between a session key generated from the Rand distribution and a session key chosen at random. In the Rand distribution, the transcript T^* constrains the value $(r_0 \,\|\, Z_0)$ by the

following n equations:

$$\begin{aligned} y_1 &= (r_0 \parallel Z_0) \oplus \alpha_1 \\ &\cdots \\ y_n &= (r_0 \parallel Z_0) \oplus \alpha_n. \end{aligned}$$

Since $(r_0 \parallel Z_0)$ cannot be expressed as a linear combination of any of the equations above, we can conclude that our session key K^* is independent of T^*. So we have

$$\Pr\left[\mathcal{A}(T^*, (r_0)_b) = b \mid (T^*, (r_0)_1) \leftarrow \mathsf{Rand};\ (r_0)_0 \leftarrow \mathbb{G}_2;\ b \leftarrow \{0,1\}\right] = 1/2$$

and since the hash function H_1 is a random oracle, we obtain the result of the lemma. ∎

The construction of the algorithm $\mathcal{D}$ is as follows. We assume that $\mathcal{A}$ makes the Test query to an oracle that is produced by the ρ^{th} Execute query, $1 \leq \rho \leq q_{ex}$. The algorithm $\mathcal{D}$ chooses some $m \in [1, q_{ex}]$ as its guess for the value of ρ. $\mathcal{D}$ calls upon $\mathcal{A}$ and responds to all queries in the usual way, except if it is the m^{th} Execute query. In the latter case, $\mathcal{D}$ embeds the DBDH problem into the transcript by generating (T^*, K^*) from the Simulation distribution (from Lemma 7.3) and responds to the m^{th} Execute query of $\mathcal{A}$ with the transcript T^*. In response to the Test query, the algorithm $\mathcal{D}$ outputs the session key K^* if $m = \rho$ and a random session key otherwise. When $\mathcal{A}$ finally outputs its guess, $\mathcal{D}$ outputs 1 if $b = b'$ and 0 otherwise.

Applying Lemmas 7.3 and 7.4 to the fact that $\Pr[m = \rho] = 1/q_{ex}$, we obtain

$$\Pr\left[\mathcal{A}(T^*, K_b^*) = b \mid (T^*, K_1^*) \leftarrow \mathsf{Real};\ K_0^* \leftarrow \{0,1\}^*;\ b \leftarrow \{0,1\}\right] = 1/2 + \epsilon$$

and

$$\mathrm{Adv}^{DBDH}_{\mathbb{G}_1, \mathbb{G}_2, e}(t') = \epsilon / q_{ex},$$

which satisfies the proof for the third case of Theorem 2.

7.4 Auxiliary Key Agreement

Recall from Section 2.1.1, that auxiliary key agreement (AKA) protocols are used to rekey the group following a membership change. The main purpose of these protocols is to reduce the communication costs from one session to the next and the impact of the 1-affects-n phenomenon. However, we must take care in the design of AKA protocols so that we still provide the same security properties as the initial key agreement protocol.

In this section, we discuss the concept of auxiliary key agreement in the ChChH-ID-AGKA protocol. In Section 7.4.1, we outline an approach for maintaining the properties of key independence, key freshness and perfect forward secrecy in the auxiliary key agreement algorithms of the ChChH-ID-AGKA protocol. In Section 7.4.2, we propose a protocol to be executed when members wish to join the group and in Section 7.4.3, we propose a protocol to be executed when members leave the group. Finally, in Section 7.4.4, we show that the ChChH-ID-AGKA protocol will maintain perfect forward secrecy in the presence of the join and leave protocols.

7.4.1 Maintaining Security in Auxiliary Key Agreement

The central idea behind auxiliary key agreement is that all group members should not be required to broadcast new key contributions after a member joins or leaves the multicast group. We can still provide key independence and key freshness, while allowing the members to use their contribution from the previous session. However, we must show caution in the design of these AKA protocols to ensure that the protocol maintains the property of perfect forward secrecy. We note that it is a non-trivial problem to maintain perfect forward secrecy using auxiliary key agreement protocols. For example, the CKA-based protocols of [BCEP03, KNKW05, CSCW07, BAA+07]

claim to provide perfect forward secrecy, but require the group members to store their ephemeral private keys for long-term use in the AKA protocols. In particular, the responders use the same ephemeral private key to recover the leader's new contribution following a membership change. If the ephemeral private key of any responder is disclosed, it will compromise the group key of every session in which that member was a participant. Consequently, these auxiliary key agreement protocols do not maintain the perfect forward secrecy of the initial key agreement protocol. As far as we know, the Tree-based Group Diffie-Hellman protocol (TGDH) of Section 2.3.4 and its variations, which require short-term storage of ephemeral values for auxiliary key agreement, provide the best approach for maintaining perfect forward secrecy thus far.

In the initial key agreement algorithm of the ChChH-ID-AGKA protocol, the final session key was computed as $K = H_1(K_{\mathcal{R}}) \oplus H_1(r_0)$. We may think of the product of responder contributions $K_{\mathcal{R}}$ as the public part of the session key, while the leader's contribution r_0 is the private part of the session key. If a member joins the group, forming the new set of responders $\mathcal{R}'$, the public part of the new session key $K'_{\mathcal{R}}$ should comprise of the contributions of the previous responders and the new member. As the new member does not know the previous key $K_{\mathcal{R}}$, the group leader broadcasts this value in some authenticated manner. The new member sends out his key contribution as in the initial key agreement protocol. To ensure key independence and key freshness, the group leader must send out a new key contribution r'_0, that serves as the private part of the new session key. If a member leaves the group, that member's contribution should be removed from the public part of the new session key $K'_{\mathcal{R}}$. Similarly, the group leader must choose a new contribution r'_0 to ensure key independence and key freshness.

While auxiliary key agreement reduces the communication costs, a drawback is that our protocol cannot provide the properties of group integrity, verifiable contributiveness or key integrity for members joining the group. In order for a new member

to be assured that the value $K_{\mathcal{R}}$ is computed using only the key contributions of the responders of the previous session, those responders would have to rebroadcast their signed contributions, defeating the purpose of auxiliary key agreement. However, since the group leader sends the public part of the previous session key $K_{\mathcal{R}}$ in an authenticated manner, we still provide a stronger notion of contributivity than the other GKA protocols from Section 2.3 and Chapter 5. Furthermore, since the signed contributions of the responders are publicly broadcasted, the new members could obtain these values from previous session transcripts and thus be assured that $K_{\mathcal{R}}$ is computed using only the contributions of the responders $\mathcal{R}$.

As shown in the security proof of Section 7.3, the ChChH-ID-AGKA protocol provides perfect forward secrecy, provided that the initial key agreement protocol is executed for each new session. This is achieved by bringing the ephemeral public/private key pair of the responder into the leader contribution masking process, as described in Section 7.1.2. We can use the same masking procedure for members joining the group and be sure that their long-term private keys may be disclosed without compromising the leader's secret contribution. However, the previous responders do not choose new ephemeral keys in auxiliary key agreement protocols and, as discussed above, ephemeral keys that are stored for an extended period of time are essentially long-term keys. Ideally, the ephemeral keys should be created and discarded in the same session. To allow the long-term keys of the previous responders to be disclosed without compromising session keys, we propose that the masking process involve the private part of the previous session key (i.e. the leader's secret contribution r_0 from the previous session). In order to keep the contribution hidden from members leaving the group, we must also bring the long-term public/private key of the responders into the masking process. Thus, the masking process will involve the long-term and ephemeral keys of members joining the group and the long-term keys and the private part of the previous session key for current members. In Sections 7.4.2 and 7.4.3, we will see exactly how this masking process is achieved in the ChChH-ID-AGKA protocol.

7.4.2 Join Protocol

Suppose that t group members $\mathcal{J} = \{U_{n+1}, ..., U_{n+t}\}$ wish to join the group $\mathcal{U} = \{U_0, ..., U_n\}$. Each new member $U_j \in \mathcal{J}$ is issued a public/private key pair (Q_j, d_j) by the PKG. The group leader delivers her secret key contribution to each responder $U_i \in \mathcal{R} \cup \mathcal{J}$ in one of two different ways, depending on whether U_i is a pre-existing member or a joining member. The session key is updated in the following two rounds:

Round 1 Each joining member $U_j \in \mathcal{J}$ chooses values $a_j, b_j \in_R \mathbb{Z}_q^\times$, computes $X_j = a_j Q_j$, $r_j = e(b_j d_j, P)$ and $Z_j = (b_j + H_2(X_j, r_j))d_j$ and broadcasts $\langle X_j, \sigma_j \rangle$ to the group, where $\sigma_j = (r_j, Z_j)$ is an H-IBS variant signature.

Round 2 The group leader U_0 chooses a vector $(\delta_{n+1}, ..., \delta_{n+t}) \in_R \left(\mathbb{Z}_q^\times\right)^t$ and verifies the signatures $(\sigma_{n+1}, ..., \sigma_{n+t})$ by checking that

$$e(\sum_{j=n+1}^{n+t} \delta_j Z_j, P) = e(\sum_{j=n+1}^{n+t} \delta_j h_j Q_j, P_{pub}) \cdot \prod_{j=n+1}^{n+t} r_j^{\delta_j},$$

where $h_j = H_2(X_j, r_j)$. If the verification fails, the group leader broadcasts an "`abort`" message. Otherwise, she computes the public part of the session key

$$K'_{\mathcal{R}} = K_{\mathcal{R}} \cdot \prod_{j=n+1}^{n+t} r_j$$

using the public part of the previous session key $K_{\mathcal{R}}$. In addition, she chooses new values $a'_0, b'_0 \in_R \mathbb{Z}_q^\times$ and computes $X'_0 = a'_0 Q_0$, $r'_0 = e(b'_0 d_0, P)$ and $Z'_0 = (a'_0 + H_3(X'_0, r'_0, K'_{\mathcal{R}}))d_0$, using the hash function H_3.

To provide perfect forward secrecy (as we discussed in Section 7.4.1), she masks her new key contribution r'_0 for each member U_i in one of the following ways:

1. If $U_i \in \mathcal{R}$ (i.e. current member) then the group leader computes

$$t_i = r_0 \cdot e(a'_0 d_0, Q_i)$$

$$y_i = (r'_0 \parallel Z'_0) \oplus H_1(t_i).$$

2. If $U_j \in \mathcal{J}$ (i.e. joining member) then the group leader computes

$$t_j = e(a'_0 d_0, X_j)$$
$$y_j = (r'_0 \parallel Z'_0) \oplus H_1(t_j).$$

The group leader broadcasts the set $\langle K_{\mathcal{R}}, X'_0, y_1, ..., y_{n+t} \rangle$ to the group, where $K_{\mathcal{R}}$ is the public part of the previous session key and X'_0 is part of the leader's ChCh-IBS signature $\sigma_0 = (X'_0, Z'_0)$.

Key Computation Upon receiving the ciphertext set, each previous member $U_i \in \mathcal{U}$ verifies that $K_{\mathcal{R}}$ is equal to the public part of the key stored from the previous session. Each responder $U_i \in \mathcal{R} \cup \mathcal{J}$ extracts their ciphertext y_i and recovers the leader's secret contribution r'_0 and the signature value Z'_0 in one of the following two ways:

1. If $U_i \in \mathcal{R}$, they recall the leader's previous contribution r_0 and compute

$$t_i = r_0 \cdot e(X'_0, d_i)$$
$$(r'_0 \parallel Z'_0) = y_i \oplus H_1(t_i).$$

2. If $U_j \in \mathcal{J}$, they compute

$$t_j = e(X'_0, d_j)^{a_j}$$
$$(r'_0 \parallel Z'_0) = y_i \oplus H_1(t_j).$$

Each responder computes the public part of the session key

$$K'_{\mathcal{R}} = K_{\mathcal{R}} \cdot \prod_{j=n+1}^{n+t} r_j = \prod_{i=1}^{n+t} r_i = e(b_1 Q_1 + ... + b_{n+t} Q_{n+t}, P_{pub}),$$

using the leader broadcasted value $K_{\mathcal{R}}$. Each responder then chooses a vector $(\delta'_0, \delta'_{n+1}, ..., \delta'_{n+t}) \in_R (\mathbb{Z}_q^\times)^{t+1}$ and verifies all signatures $(\sigma_0, \sigma_{n+1}, ..., \sigma_{n+t})$ by checking that

$$e(\delta'_0 Z'_0 + \sum_{j=n+1}^{n+t} \delta'_j Z_j, P) = e(\delta'_0 (X'_0 + h_0 Q_0) + \sum_{j=n+1}^{n+t} \delta'_j h_j Q_j \ , \ P_{pub}) \cdot \prod_{j=n+1}^{n+t} r_j^{\delta'_j},$$

where $h_0 = H_3(X'_0, r'_0, K'_{\mathcal{R}})$ and $h_j = H_2(X_j, r_j)$, for $n+1 \leq j \leq n+t$. If the verification fails, they broadcast an "`abort`" message. Otherwise, all members compute the final session key as

$$K' = H_1(K'_{\mathcal{R}}) \oplus H_1(r'_0).$$

7.4.3 Leave Protocol

Suppose that a set of $t < n$ group members $\mathcal{L}$ wish to leave the group $\mathcal{U} = \{U_0, U_1, ..., U_n\}$. Upon a reordering of indices, we may assume that the leaving members form the subgroup $\mathcal{L} = \{U_{n-t+1}, ..., U_n\}$. The new session key for the group $\mathcal{U} \setminus \mathcal{L}$ is updated in a single round, as follows:

Round 1 The group leader computes a key update value for all remaining group members,

$$K_{\mathcal{L}} = \prod_{i=n-t+1}^{n} r_i$$

consisting of the contributions of all leaving members. She chooses new a'_0, $b'_0 \in_R \mathbb{Z}_q^\times$ and computes $X'_0 = a'_0 Q_0$ and $r'_0 = e(b'_0 d_0, P)$. In addition, she computes the public part of the session key $K'_{\mathcal{R}} = K_{\mathcal{R}} \cdot K_{\mathcal{L}}^{-1}$ and the signature value $Z'_0 = (a'_0 + H_3(X'_0, r'_0, K'_{\mathcal{R}}))d_0$.

For each non-leaving responder $U_i \in \mathcal{R} \setminus \mathcal{L}$, the group leader masks her new key

contribution by computing

$$\begin{aligned} t_i &= r_0 \cdot e(a'_0 d_0, Q_i) \\ (r'_0 \parallel Z'_0) &= y_i \oplus H_1(t_i). \end{aligned}$$

The group leader broadcasts the set $\langle K_{\mathcal{L}}, X'_0, y_1, ..., y_{n-t} \rangle$ to the group, where X'_0 is part of the ChCh-IBS signature $\sigma'_0 = (X'_0, Z'_0)$.

Key Computation Upon receiving the ciphertext y_i, each responder $U_i \in \mathcal{R} \setminus \mathcal{L}$ recalls the leader's previous contribution r_0 and computes

$$\begin{aligned} t_i &= r_0 \cdot e(X'_0, d_i) \\ (r'_0 \parallel Z'_0) &= y_i \oplus H_1(t_i) \end{aligned}$$

and

$$K'_{\mathcal{R}} = K_{\mathcal{R}} \cdot K_{\mathcal{L}}^{-1}.$$

Each responder verifies that

$$e(Z'_0, P) = e(X'_0 + H_3(X'_0, r'_0, K'_{\mathcal{R}})Q_0, P_{pub}).$$

If the verification holds, they compute the session key as

$$K' = H_1(K'_{\mathcal{R}}) \oplus H_1(r'_0).$$

7.4.4 Perfect Forward Secrecy

We claim that the ChChH-ID-AGKA protocol maintains perfect forward secrecy when the auxiliary key agreement protocols of Sections 7.4.2 and 7.4.3 are employed. We provide a proof of this claim here.

Claim 7.1 *The* ChChH-ID-AGKA *protocol provides perfect forward secrecy in the presence of the Join and Leave protocols.*

Proof: Suppose an adversary $\mathcal{A}$ is in possession of the long-term private key d_j of each group member $U_j \in \mathcal{U}$ in the i^{th} session. We claim that $\mathcal{A}$ can masquerade as any group member, but cannot determine past session keys. In the attack, we assume that $\mathcal{A}$ attempts to recover the leader's secret contribution r_0^i for the i^{th} session. We have the following two cases:

1. For a joining member U_j, $\mathcal{A}$ is faced with the same problem as in the initial key agreement protocol. Following the argument of Theorem 2, if $\mathcal{A}$ has an advantage in determining the session key, then we can construct an algorithm $\mathcal{D}$ to solve the DBDH problem with some advantage. Thus the adversary has a negligible advantage in determining the key from a joining member.

2. For a current member U_j that joined the group in the l^{th} session, $1 \leq l < i$, an adversary $\mathcal{A}$ obtains d_j from the Corrupt(U_j) query and the values X_0^i and y_j^i from the protocol transcript by issuing an Execute query. In order to recover r_0^i from y_j^i, $\mathcal{A}$ must compute

$$t_j^i = r_0^{i-1} \cdot e(X_0^i, d_j),$$

which requires knowledge of the leader's previous contribution r_0^{i-1}. The adversary obtains X_0^{i-1} and y_j^{i-1} from the protocol transcript and attempts to recover r_0^{i-1} from y_j^{i-1}. However, this requires $\mathcal{A}$ to compute

$$t_j^{i-1} = r_0^{i-2} \cdot e(X_0^{i-1}, d_j),$$

which requires knowledge of the contribution r_0^{i-2}. Continuing in this fashion, $\mathcal{A}$ eventually attempts to recover the leader's contribution r_0^l from y_j^l for the session l in which U_j joined the group. From the previous case, it is infeasible

for the adversary to recover the leader's contribution from a joining member. As a result, $\mathcal{A}$ has a negligible advantage in determining the session key from a current member.

Thus knowledge of the long-term private keys of all group members does not allow an adversary to recover the secret contribution r_0^i for any session i, without some additional hidden information, such as the leader's contribution from a previous session r_0^l or the ephemeral private key a_j of a member joining the l^{th} session for $1 \leq l \leq i$. Thus, since the adversary has no advantage in determining past session keys, our protocol provides perfect forward secrecy. ∎

7.5 Efficiency

In this section, we compare the efficiency of the ChChH-ID-AGKA protocol with the LKKR-GKA, CHL-ID-AGKA and KNKW-ID-AGKA protocols of Chapter 5. We compare the protocols in terms of contributory properties and communication costs in Section 7.5.1 and in terms of computational costs in Section 7.5.2. As in previous analyses, we drop the suffixes of the protocol for our discussion.

7.5.1 Communication Costs

We compare the properties and communication costs of the various protocols in Table 7.3. As mentioned in Section 7.1.2, the ChChH protocol provides perfect forward secrecy at the expense of a second round of communication. Since ephemeral values are used in the computation of the session key, the ChChH protocol provides inherent perfect forward secrecy if the initial key agreement protocol from Section 7.2 is executed for each new session. In addition, the ChChH protocol maintains the property of perfect forward secrecy when the auxiliary key agreement protocols are executed,

as shown in Section 7.4. The LKKR protocol also achieves this feat, but requires $h = \log_3(n+1)$ rounds of communication. The CHL and KNKW protocols provide constant round complexity and both provide inherent perfect forward secrecy. However, the CHL protocol does not propose algorithms for auxiliary key agreement and, while the KNKW protocol does propose AKA protocols, they do not maintain perfect forward secrecy (which is denoted by † in Table 7.3).

As discussed in Section 7.2.3, the ChChH protocol provides the nice contributory properties of the COMPASS-AGKA protocols. The LKKR protocol does not provide authentication and cannot guarantee the contributory properties due to the tree-based nature of the protocol, as discussed in Section 5.1.6. The CHL and KNKW protocols only provide the weaker notion of signature screening (denoted by $*$ in Table 7.3) and thus fail to satisfy many of the properties, as discussed in Sections 5.2.4 and 5.3.5.

Finally, the ChChH protocol still achieves the optimal communication costs of the COMPASS-AGKA protocol, requiring only one broadcast message per member. The KNKW protocol also reaches this lower bound, while the LKKR protocol requires $\frac{3}{2}n$ messages and the CHL protocol requires $2(n+1)$ messages.

		ChChH	LKKR	CHL	KNKW
Rounds (Max)		2	h	2	2
Perfect Forward Secrecy		✓	✓	✓	✓†
AKA Protocols		✓	✓	×	✓
Contributory Key		✓	✓	✓	✓
Authentication		✓	×	✓*	✓*◊
Group Integrity		✓	×	×	×
Verifiable Contributiveness		✓	×	×	×
Key Integrity		✓	×	×	×
Messages (Total)	Unicast	–	–	–	n
	Multicast	$n+1$	$\frac{3}{2}n$	$2(n+1)$	1

Table 7.3. Communication Costs of the ChChH-ID-AGKA Protocol.

7.5.2 Computational Costs

We compare the computational costs of the protocols in terms of pairings $\mathcal{P}$ and scalar multiplications $\mathcal{M}$ in $\mathbb{G}_1$ in Table 7.4. Similar to the analyses of Sections 5.3.5 and 6.6.2, we separate the computations of the leader and the responders in terms of key agreement and authentication. In order to promote a fair comparison of the protocols, we present the modified costs of the LKKR, CHL and KNKW protocols, as given in Section 6.6.2, so that the total costs represent the computations required to provide implicit key authentication and resilience to attacks.

The ChChH protocol is very efficient in terms of the expensive pairing operation, requiring much less pairing computations than the LKKR and CHL protocols and only a single pairing in total more than the KNKW protocol. Similar to our analysis of the YYHZ-COMPASS-AGKA protocol in Section 6.6.1, the ChChH protocol appears to be more computationally expensive than the other AGKA protocols in terms of scalar multiplications. However, we reiterate that the majority of these costs are associated with strong batch verification, which allows for the contributory properties shown in Table 7.3.

Protocol	Computations (GKA / Authentication)			
	$\mathcal{P}$		$\mathcal{M}$	
	u_0	u_i	u_0	u_i
ChChH-ID-AGKA	n / 2	1 / 3	2 / $2n+1$	2 / $2n+2$
Total	$5n+2$		$2n^2+6n+3$	
LKKR-GKA (§5.1.4)	$hn+h$ / $(4hn+4h)$		$\frac{3}{2}n$ / $((2h+3)n+2h)$	
Total	$(5hn+5h)$		$\left(\frac{(4h+9)}{2}n+2h\right)$	
CHL-ID-AGKA (§5.2.2)	2 / $2+(2)$		2 / $5+(n+3)$	
Total	$(6n+6)$		$(n^2+11n+10)$	
KNKW-ID-AGKA (§5.3.1)	n / $n+1$	1 / (2)	3 / $n+(2)$	− / $3+(1)$
Total	$(5n+1)$		$(5n+5)$	

Table 7.4. Computational Costs of the ChChH-ID-AGKA Protocol.

7.6 Summary

In this chapter, we proposed a novel identity-based authenticated group key agreement protocol (ChChH-ID-AGKA) that achieves the difficult task of providing perfect forward secrecy in the presence of auxiliary key agreement protocols. Our ChChH-ID-AGKA protocol illlustrates how the COMPASS framework can be modified to provide perfect forward secrecy for some signcryption schemes, while requiring only one additional round of communication. Since the ChChH-ID-AGKA protocol is a modification of the COMPASS-AGKA protocol of Chapter 6, it provides the lower bound for communication in GKA protocols, requiring a single broadcast message per member, and the desirable contributory properties of group integrity, verifiable contributiveness and key integrity. We showed that the proposed protocol is semantically secure in the random oracle model, provided that the DBDH problem is computationally infeasible, and computationally efficient in comparison to the pairing-based GKA protocols of Chapter 5. We addressed the issue of preserving key security properties, such as key independence, key freshness and perfect forward secrecy, when members join or leave the multicast group and proposed efficient auxiliary key agreement protocols that maintain perfect forward secrecy, without requiring the group members to store ephemeral values for long-term use.

Chapter 8

Conclusions

8.1 Summary of Results

In this thesis, we investigated the use of bilinear pairings in the design of secure and efficient authenticated group key agreement protocols. As a result of our work, we proposed several interesting and original contributions to the field.

In Chapter 5, we presented the first comprehensive literature survey and a novel classification of group key agreement from bilinear pairings. We showed that each protocol can be categorized as a pairing-based variation of either the Tree-based Group Diffie-Hellman protocol, the Burmester-Desmedt protocol or the computationally asymmetric Conference Key Agreement protocol from Section 2.3. Furthermore, we distinguish between the pairing-based variations of each class based on specific criteria, such as group structure, key computation, authentication and vulnerability to attacks. We provided a thorough analysis of the communication and computational costs of the pairing-based GKA protocols and as a result, we propose what we believe to be the most suitable pairing-based variations for each class. In particular, we suggest that the tree-based LKKR-GKA protocol of Lee *et al.* from Section 5.1.4, the BD-based CHL-ID-AGKA protocol of Choi *et al.* from Section 5.2.2 and the CKA-based KNKW-ID-AGKA protocol of Kim *et al.* from Section 5.3.1, provide the most efficient

and secure solutions to pairing-based group key agreement from current literature.

In Chapter 6, we proposed a novel framework, which we call the COMPASS framework, to construct secure and efficient computationally asymmetric authenticated group key agreement protocols from identity-based signcryption schemes. In particular, we showed that the COMPASS-AGKA protocols achieve the lower bound for communication efficiency in group key agreement, requiring each member to broadcast only one message in a single round of communication, and proved the semantic security of the COMPASS-AGKA protocols in the random oracle model. In addition, the COMPASS-AGKA protocols provide stronger notions of security than the GKA classes of Chapter 5 through resilience to the unknown key share attacks of the CKA protocol and the provision of the contributory properties of group integrity, verifiable contributiveness and key integrity. Furthermore, we showed that the COMPASS-AGKA protocol produced from the YYHZ-IBSC scheme of Yu *et al.* from Section 4.3.6 is more computationally efficient than the pairing-based GKA protocols presented in the literature survey of Chapter 5. As a result of the COMPASS framework, we reduce the problem of constructing efficient and secure authenticated group key agreement protocols to designing suitable and provably secure signcryption schemes.

In Chapter 7, we proposed a novel identity-based authenticated group key agreement protocol that, to the best of our knowledge, is the first to maintain perfect forward secrecy in the presence of auxiliary key agreement protocols, while allowing members to generate and discard their ephemeral key pair in the same session. The proposed ChChH-ID-AGKA protocol is variation of the COMPASS-AGKA protocol produced from the CML-IBSC scheme of Chen and Malone-Lee from Section 4.3.5 and consequently provides the same communication efficiency and contributory properties of the COMPASS-AGKA protocols. In addition, we illustrated how the COMPASS framework of Chapter 6 can be modified to provide perfect forward secrecy at the expense of only a second round of communication. Furthermore, we reduced the semantic security of the ChChH-ID-AGKA protocol to the intractability of the DBDH

problem and showed that our protocol is computationally efficient in comparison to the pairing-based GKA protocols of Chapter 5.

The rest of the thesis provided the necessary background to support the understanding of our contributions. In Chapter 2, we presented the primary issues of concern when designing group key establishment and, more specifically, group key agreement protocols and defined the formal security model that was later used to prove the security of the COMPASS-AGKA and ChChH-ID-AGKA protocols. In Chapter 3, we introduced the concept of bilinear pairings and identity-based cryptography. The goal of this chapter was to demonstrate the use of the pairing as a building block in the construction of cryptographic protocols, while presenting specific protocols that arise in subsequent chapters. In particular, many of the pairing-based group key agreement protocols from Chapter 5 were constructed using the key exchange protocols of Joux and Smart, while every identity-based cryptographic protocol considered in this thesis depends on the public key infrastructure of Boneh and Franklin's identity-based encryption scheme.

Arguably, the most instrumental background material was covered in Chapter 4, where we addressed the issue of authentication in identity-based public key cryptography. We discussed the concept of signcryption, upon which the COMPASS framework was constructed, and several of the important signcryption properties used to optimize the framework. We defined the formal security models for identity-based signature and signcryption schemes, which were later used to prove the security of the COMPASS-AGKA and ChChH-ID-AGKA protocols. We discussed the concept of strong batch verification, which was used to achieve several contributory properties in the COMPASS-AGKA and ChChH-ID-AGKA protocols, and discussed its advantages over signature screening. We presented the identity-based signature schemes of Cha-Cheon and Hess, which were used to provide authentication in the pairing-based GKA protocols of Chapter 5 and used in the construction of our ChChH-ID-AGKA protocol, along with the identity-based signcryption scheme of Chen and Malone-Lee. We

also presented the multi-receiver signcryption scheme of Yu *et al.* which was used to illustrate the COMPASS framework in Chapter 6.

Finally, for the mathematically inclined reader, we provided a thorough discussion of the concepts used to formally define and efficiently compute the Weil and Tate pairings in Appendix A.

8.2 Future Research

Many interesting areas of future research have resulted from the work presented in this thesis. We describe these varied research problems here.

Perfect Forward Secrecy We modified the one-round COMPASS framework to provide perfect forward secrecy in the ChChH-ID-AGKA protocol, at the expense of a second round of communication. Is it possible to achieve perfect forward secrecy in a group key agreement protocol requiring only a single round of communication? Furthermore, we maintained perfect forward secrecy using a unique approach in the auxiliary key agreement protocols. We may like to prove the security of the ChChH-ID-AGKA protocol with the auxiliary protocols in the dynamic model of [BCP02]. Can this approach be used to maintain perfect forward secrecy in the auxiliary key algorithms of other group key agreement protocols?

Computationally Asymmetric GKA In this thesis, we referenced several different computationally asymmetric group key agreement protocols in [BCEP03, KNKW05, NLKW05, KKC+06, ZSM06, BAA+07, CSCW07, HLL07, Tse07] and proposed two of our own, the YYHZ-COMPASS-AGKA and ChChH-ID-AGKA protocols. This class of protocol is exceptionally well suited for networks of low power mobile devices, since we can delegate a large portion of the expensive computations to a more powerful server. Are there applications for which we don't want to delegate too much of the computational load to the leader? What

are the ideal computational distributions and exact security requirements for the various applications in this type of group communication model? Moreover, how do the various computationally asymmetric protocols compare in terms of these requirements?

Certificate-based PKI Although pairing- and identity-based cryptography are active areas of research, many recent protocols are still based on the classic Diffie-Hellman key exchange and make use of the traditional certificate-based public key infrastructure. It would be interesting to consider COMPASS-AGKA protocols obtained from DH-based signcryption schemes and see how these protocols compare to the computationally asymmetric, DH-based protocols from [BCEP03, NLKW05, KKC+06, Tse07].

The Standard Model Recently, some cryptographers have chosen to design their protocols to be proven secure in the standard model, without depending on the use of random oracles. In particular, identity-based encryption schemes secure in the standard model were proposed in [BB04, Wat05], which motivated other cryptographers to design and prove the security of identity-based authentication schemes and key exchange protocols in the standard model. Based on these works, can we produce identity-based authenticated group key agreement protocols secure in the standard model? Is it possible to reduce the security of COMPASS-AGKA protocols to the security of identity-based signcryption schemes which are proven secure in the standard model?

COMPASS-AGKA Protocols While we presented the COMPASS-AGKA protocol obtained from Yu *et al.*'s multi-receiver signcryption scheme, it would be interesting to apply the COMPASS framework to other identity-based signcryption schemes from current literature, such as the efficient $\mathcal{S}$-verifiable signcryption scheme of Barreto *et al.* from [BLMQ05]. To produce an efficient group key agreement protocol, their signcryption scheme must be modified to support

multiple recipients and provide batch verification of signatures in the underlying signature scheme, while preserving the provable security of the scheme. Can we construct new identity-based signcryption schemes, which are specifically designed for the COMPASS framework, that are more efficient than the existing signcryption schemes?

Appendix A

The Mathematics of Pairings

In this Appendix, we discuss the background mathematics required to fully understand bilinear pairings. In Section A.1, we define the concept of elliptic curves and show how they form abelian groups under the so-called chord-and-tangent rule. We present the concepts and results from the theory of elliptic curves that are essential to our understanding of pairings. In particular, we investigate the concept of torsion subgroups and rational functions. In Section A.2, we go deeper into a branch of algebraic geometry to study the theory of divisors. While knowledge of elliptic curves is sufficient to understand pairings as bilinear maps, a thorough understanding of divisors is necessary to comprehend the formal definition and computation of pairings. Finally, we arrive at the concept of pairings in Section A.3. We give a detailed presentation of the two most common types of pairings, the Weil and Tate pairings. In Section A.4, we show how these pairings can be modified for use in cryptographic applications and describe the algorithm used to efficiently compute the pairings.

A.1 Elliptic Curves

While the term *elliptic curve* may elicit images of geometrical figures or planetary orbits from other strains of mathematics, the concept is deeply rooted in the fields of

abstract algebra, algebraic geometry and number theory. Since elliptic curves were first (independently) proposed back in 1985 by Koblitz and Miller for cryptographic purposes, they have taken the cryptographic community by storm, being used for practical applications from factorization algorithms to key agreement protocols and spawning new fields of interest, such as hyperelliptic curve cryptography and pairing-based cryptography.

The focus of this section is to provide enough background on elliptic curves to comprehend the forthcoming sections on divisors and pairings. Throughout this section, we present many general definitions and results from the theory of elliptic curves. As our main goal is to understand the use of elliptic curves in pairings, we omit the proofs of these results, which can be found in most introductory textbooks on the subject. In particular, we refer the reader to [Sil86, HPS08, Was08]. Instead, we illustrate the concepts with meaningful examples to be built upon in subsequent sections.

A.1.1 Weierstrass Equations

We begin this section by adapting the general definition of an elliptic curve (see, for example [Sil86, Prop. III.3.1]) to obtain the more simplified version commonly found in cryptographic discussions.

Definition A.1 *A smooth cubic curve C given by a* Weierstrass equation *of the form*

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3, \tag{A.1.1}$$

with coefficients $a_1, ..., a_6 \in K$, is an elliptic curve *over the field K with basepoint $\mathcal{O} = [0, 1, 0]$.*

Remark A.1 *We define the concept of a* smooth curve *in Section A.1.2 and explain its importance in the definition of an elliptic curve.*

By convention, we denote an elliptic curve given in Definition A.1 by E. To ease our notation, we can switch Equation (A.1.1) to non-homogeneous coordinates by substituting $x = X/Z$ and $y = Y/Z$ to yield

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6. \tag{A.1.2}$$

A polynomial given by Equation (A.1.2) with $a_1, ..., a_6 \in K$ is called a *generalized Weierstrass equation.* We note that the basepoint $\mathcal{O} = [0, 1, 0]$ is the only solution to Equation (A.1.1) that is not accounted for by Equation (A.1.2). As a result, we include the basepoint $\mathcal{O}$, which we call the *point-at-infinity*, and view an elliptic curve as the pair $(E, \mathcal{O})$.

If the characteristic of the field K is different from 2 or 3, the general Weierstrass equation can be further reduced to the *short Weierstrass form*

$$y^2 = x^3 + Ax + B, \tag{A.1.3}$$

with $A, B \in K$. For cryptographic purposes, we often choose a field with large prime characteristic. As a result, we will always work with elliptic curves in the short Weierstrass form.

We illustrate the concept of elliptic curves by first defining them over the field of real numbers $\mathbb{R}$.

Definition A.2 *An* elliptic curve $E(\mathbb{R})$ *is the graph of the* Weierstrass equation $y^2 = x^3 + Ax + B$ *with constant coefficients* $A, B \in \mathbb{R}$, *along with the point-at-infinity* $\mathcal{O}$.

To help us visualize our set of points, we consider the equations $y^2 = x^3 - 3x$ and $y^2 = x^3 + 3x$. The first equation has three distinct real roots corresponding to the points at which the curve intersects the x-axis. The second equation only intersects the x-axis at a single point and thus has only one real root. The graphs of these

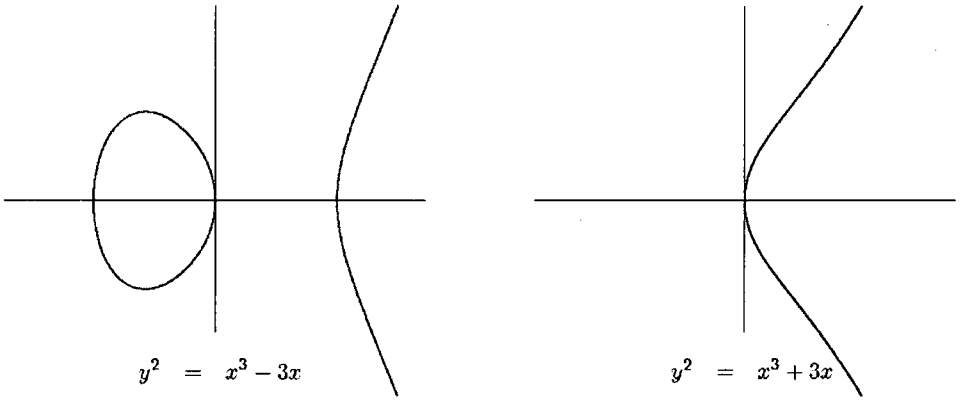

Figure A.1: The Graphs of Weierstrass Equations over $\mathbb{R}$.

equations, shown in Figure A.1, illustrate two basic types of elliptic curves.

From the previous discussion, we see that the set of points on an elliptic curve over $\mathbb{R}$ is given by all real solutions to the Weierstrass equation, along with the special point-at-infinity $\mathcal{O}$. Over $\mathbb{R}$, we may visualize $\mathcal{O}$ to be the point positioned simultaneously at the very top and the very bottom of the y-axis in the Cartesian plane. The importance of the point-at-infinity will be apparent in the Section A.1.2.

Unfortunately, when we aren't working over the field of real numbers, our elliptic curve does not have the nice graphical representation of Figure A.1. In general, we consider an elliptic curve E over a base field K, where K is taken to be one of the following fields: the real numbers $\mathbb{R}$, the rational numbers $\mathbb{Q}$, the complex numbers $\mathbb{C}$, the finite field F_p for prime p or some finite extension field F_q where $q = p^r$. We define the set of points on an elliptic curve E over a base field K to be the set of Cartesian coordinates $(x, y) \in K \times K$ that are solutions to Equation (A.1.3), along with the point-at-infinity $\mathcal{O}$. More formally, we have the following definition from [Was08].

Definition A.3 *The* K-rational points of an elliptic curve E defined over a field K,

or for short, the elliptic curve over K, *with* $\mathrm{char}(K) \neq 2, 3$ *is given by*

$$E(K) = \{\mathcal{O}\} \cup \{(x, y) \in K \times K \mid y^2 = x^3 + Ax + B\}$$

for some $A, B \in K$.

Before delving into the theory of elliptic curves, we should set some notational standards for the remainder of this chapter. For any field K, we let $\overline{K}$ denote a fixed algebraic closure of K. We usually write E rather than $E(K)$ when discussing the elliptic curve over the base field K. However, when working with an extension field $L \supseteq K$, we write $E(L)$ and $E(K)$ to distinguish between the different curves.

In the next few sections, we present the theory of elliptic curves in terms of a general field K. However, since our work is mainly concerned with elliptic curves over finite fields, we discuss some results specific to finite fields when applicable. We provide our own example to illustrate the concept of an elliptic curve over a finite field. The reader should pay particular attention to this example, as we will refer back to it in future sections.

Example A.1 *Let* E *be the elliptic curve over* F_{31} *defined by the Weierstrass equation*

$$y^2 = x^3 + 11.$$

We can construct the elliptic curve $E(F_{31})$ *by determining all the pairs* $(x, y) \in F_{31} \times F_{31}$ *which satisfy* $y^2 = x^3 + 11$. *For example, the pair* $(2, 9) \in F_{31} \times F_{31}$ *satisfies*

$$(9)^2 \equiv (2)^3 + 11 \equiv 19 \mod 31$$

and thus $(2, 9) \in E(F_{31})$. *Continuing in this fashion, we can determine all* 25 *points*

of $E(F_{31})$*, given by*

$$E(F_{31}) = \left\{ \begin{array}{c} \mathcal{O}, (2,9), (2,22), (3,10), (3,21), (6,14), (6,17), (10,9), (10,22), \\ (11,3), (11,28), (13,10), (13,21), (15,10), (15,21), (19,9), (19,22), \\ (24,3), (24,28), (26,17), (26,14), (27,3), (27,28), (30,14), (30,17) \end{array} \right\}.$$

A.1.2 Group Law for Elliptic Curves

The goal of this section is to define an "addition" law on the set of points of an elliptic curve that gives $E(K)$ the structure of an abelian group. While it would certainly be nice to simply add the respective x- and y-coordinates of the two points, this does not often yield a point on the curve. Instead, we define a new group law, known as the *chord-and-tangent rule*, for the addition of points on a curve. We derive the formulas that comprise the various cases of the chord-and-tangent rule, before formally defining the group law. We note that the group law is best explained geometrically, as in Figures A.2 and A.3.

Point Addition

To add two points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, we first determine the equation of the secant line through P and Q,

$$y = \lambda (x - x_1) + y_1,$$

where λ is the slope of the line, given by

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}.$$

Since E is a cubic curve and the intersection of the secant line and E has roots P, Q with coordinates in K, the line will intersect the curve at exactly one other point $R = (x_3, y_3) \in E(K)$, as illustrated in Figure A.2. Substituting the equation of our

line into the curve, we have

$$(\lambda(x - x_1) + y_1)^2 = x^3 + Ax + B,$$

which we can rearrange to the form

$$0 = x^3 - \lambda^2 x^2 + \ldots.$$

At this point, we note that for any cubic equation, we have

$$(x - \alpha_1)(x - \alpha_2)(x - \alpha_3) = x^3 - (\alpha_1 + \alpha_2 + \alpha_3)x^2 + \ldots.$$

Applying this idea to our equation, and noting that the our 3 roots are x_1, x_2 and the unknown x_3, we obtain

$$x_1 + x_2 + x_3 = \lambda^2,$$

which we can rearrange to determine the x-coordinate of R as

$$x_3 = \lambda^2 - x_1 - x_2.$$

To obtain the y-coordinate of R, we simply substitute the x-coordinate into the equation of the line, to obtain

$$y_3 = \lambda(x_3 - x_1) + y_1.$$

Finally, we reflect the point R in the x-axis by multiplying the y-coordinate by -1, to obtain

$$y_3' = \lambda(x_1 - x_3) - y_1$$

and thus the coordinates of the point $R' = P + Q$ are given by (x_3, y_3'). One can verify that $(x_3, y_3') \in K \times K$ is indeed a solution to $y^2 = x^3 + Ax + B$ and thus

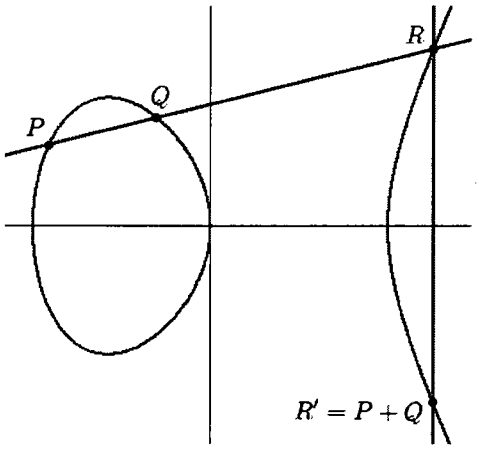

Figure A.2: Point Addition on an Elliptic Curve

$R' = (x_3, y_3') \in E(K)$.

Point Doubling

We may want to add a point $P = (x_1, y_1)$ to itself, yielding the scalar multiple $2P = (x_3, y_3)$. However, we immediately see that the formula from the previous section will not work, since the slope is undefined. Our natural intuition is to determine the secant line intersecting P and Q, while moving the point Q along the curve towards the point P. Eventually, we obtain the tangent line at P, as illustrated in Figure A.3. To determine the slope of the tangent line, we implicitly differentiate Equation (A.1.3) to obtain

$$2y \cdot dy = \left(3x^2 + A\right) \cdot dx.$$

Rearranging the equation, we obtain the slope of the tangent line at $P = (x_1, y_1)$, given by

$$\lambda = \frac{dy}{dx} = \frac{3x_1^2 + A}{2y_1}.$$

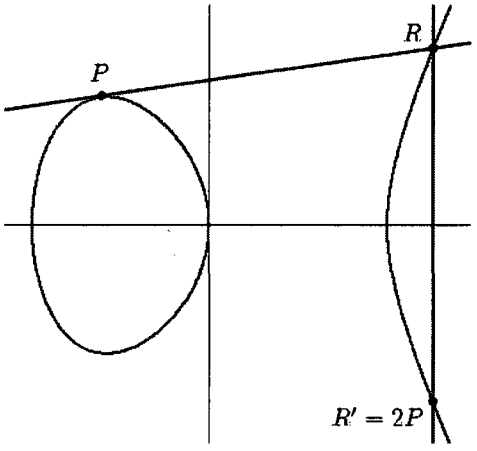

Figure A.3: Point Doubling on an Elliptic Curve

The tangent line of the curve at P is given by

$$y = \lambda (x - x_1) + y_1$$

and only passes through the curve at points P and R. Following the method of the previous section, we can determine the x-coordinate of $2P$ by computing

$$x_3 = \lambda^2 - 2x_1$$

and the y-coordinate by substituting x_3 into the tangent line and reflecting in the y-axis, as

$$y_3' = \lambda (x_1 - x_3) - y_1,$$

to obtain the coordinates of the point $R' = 2P$, given by (x_3, y_3'). Similarly, one can verify that $(x_3, y_3') \in K \times K$ is indeed a solution to $y^2 = x^3 + Ax + B$ and thus $R' = (x_3, y_3') \in E(K)$.

Example A.2 *Let E be the elliptic curve over F_{31} defined by the Weierstrass equation*

$$y^2 = x^3 + 11.$$

Suppose we are given a point $P = (11, 3) \in E$ *and want to find the point* $2P = (x_3, y_3')$. *We first find the tangent line at* P, *given by*

$$y = \frac{3x_1^2 + A}{2y_1}(x - x_1) + y_1 = \frac{3(11)^2 + (0)}{2(3)}(x - 11) + 3 = 14(x - 11) + 3 = 14x + 4.$$

Intersecting the tangent line with the curve E, *we have*

$$\begin{aligned} (14x + 4)^2 &= x^3 + 11 \\ 10x^2 + 19x + 16 &= x^3 + 11 \end{aligned}$$

which we can reduce to

$$x^3 + 21x^2 + 12x + 26 = (x - 11)^2(x - 19) = 0.$$

Since the line is tangent to the curve at P, *the intersection has root* $x = 11$ *with multiplicity* 2 *at the point* $P = (11, 3)$. *Since the only points of intersection on the line are* P *and* R, *we have* $x_3 = 19$. *Substituting this value into the equation of the line, we get* $y = 22$ *and reflecting in the* y*-axis we get* $y_3' = 9$. *Thus the point* $2P$ *is given by* $(19, 9)$.

Identity and Inverse Addition

If $(x, y) \in K \times K$ is a solution to the Weierstrass equation $y^2 = x^3 + Ax + B$, then clearly $(x, -y)$ is also a solution. Working from the previous sections, we see that $R = (x_3, y_3)$ and $R' = (x_3, -y_3)$ are both points on the elliptic curve E, as illustrated in Figure A.2. However, to add these two points, we cannot apply the results of the previous sections since the slope of the vertical line through R and R' is undefined. This is where the special *point-at-infinity* $\mathcal{O}$ discussed in Section A.1.1 comes into play. Following the method of point addition, the vertical line intersecting R and R' also goes through the point-at-infinity $\mathcal{O}$, which by Definition A.3 is an element of

our elliptic curve. Thus we define the addition of the points R and R' as $R+R' = \mathcal{O}$. To introduce the point-at-infinity into our group law, we must also define the addition of $\mathcal{O}$ and a point P. We set $P + \mathcal{O} = P$ so that $\mathcal{O}$ is treated as the identity element of our group. Consequently, the points R and R' are viewed as inverse elements in $E(K)$, and thus we write $R' = -R$.

We summarize the chord-and-tangent rule for the addition of elliptic curve points in the following definition.

Definition A.4 *Let E be an elliptic curve over a field K as given in Definition A.3 and let P, Q be points on E. We define the addition law $P + Q = R'$ as follows:*

1. *If $P = \mathcal{O}$ (resp. $Q = \mathcal{O}$), then $\mathcal{O} + Q = Q$ (resp. $P + \mathcal{O} = P$).*
2. *Otherwise, let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$. We have the following cases:*

   i) If $x_1 \neq x_2$ (i.e. Point Addition with $P \neq Q$), then $R' = (x_3, y_3')$ is given by

$$x_3 = \lambda^2 - x_1 - x_2 \qquad y_3' = \lambda(x_1 - x_3) - y_1 \tag{A.1.4}$$

   where $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$.

   ii) If $x_1 = x_2$ and $y_1 = y_2 \neq 0$ (i.e. Point Doubling with $P = Q$), then $R' = (x_3, y_3')$ is given by

$$x_3 = \lambda^2 - 2x_1 \qquad y_3' = \lambda(x_1 - x_3) - y_1 \tag{A.1.5}$$

   where $\lambda = \frac{3x_1^2 + A}{2y_1}$.

   iii) If $x_1 = x_2$ and $y_1 = -y_2$ or $y_1 = y_2 = 0$ (i.e. Inverse Addition with $Q = -P$), then $R' = \mathcal{O}$.

Applying the addition law from Definition A.4 to our set of points on the elliptic curve E, we obtain an abelian group, as illustrated by the following theorem. For a proof of the theorem, we refer the reader to [Was08, Theorem 2.1].

Theorem 3 *Let E be an elliptic curve over the field K. The addition law on E has the following properties:*

Commutativity $P + Q = Q + P$ *for all* $P, Q \in E$.

Identity $P + \mathcal{O} = \mathcal{O} + P = P$ *for all* $P \in E$.

Inverse $P + (-P) = \mathcal{O}$ *for all* $P \in E$.

Associativity $P + (Q + R) = (P + Q) + R$ *for all* $P, Q, R \in E$.

In other words, under the group addition law, the elliptic curve E forms an abelian group.

Using Definition A.4, we define scalar multiplication on the elliptic curve E using the *multiplication-by-m map*, as follows:

$$\begin{aligned} mP &= (P) + (P) + ... + (P) && \text{for } m > 0 \\ 0P &= \mathcal{O} \\ mP &= (-P) + (-P) + ... + (-P) && \text{for } m < 0. \end{aligned}$$

In elliptic curve cryptography, we often compute scalar multiples of points. Clearly computing mP with $m - 1$ separate additions is very computationally inefficient. We can compute scalar multiples much faster using what's known as the *successive doubling method.* For example, to compute the point $11P$, we would compute

$$2P = P + P \qquad 4P = 2P + 2P \qquad 8P = 4P + 4P \qquad 11P = 8P + 2P + P,$$

requiring 3 point doubles and 2 point additions, as opposed to 10 point additions. To perform the above computation from an algorithmic point of view, we use the

Double-and-Add algorithm. We first find the binary representation of the integer $m = (m_s m_{s-1} ... m_1 m_0)_2$, where $m_i \in \{0, 1\}$ and $m_s = 1$ so that

$$m = m_0 + m_1 \cdot 2^1 + m_2 \cdot 2^2 + ... + m_s \cdot 2^s.$$

The Double-and-Add algorithm, as described in [Sil86, §XI.1], is given by the following.

Algorithm A.1 (Double-and-Add)

```
set Q ⟵ P, R ⟵ O if m_0 = 0 or R ⟵ P if m_0 = 1 and s = ⌊log_2 n⌋.
for i = 1, 2, ..., s do
{
  set Q ⟵ 2Q // Point Doubling
  if m_i = 1, then
  {
    set R ⟵ R + Q // Point Addition
  }
}
return R;
```

Working from the previous example, we have $11 = (1011)_2$, $(Q, R) = (P, P)$ and $s = 3$ for initial conditions. We obtain the pair $(2P, 3P)$ after the first round, $(4P, 3P)$ after the second and $(8P, 11P)$ after the third, to obtain the final value $11P$.

Remark A.2 *We note that improving upon the Double-and-Add algorithm is an active area of research. In particular, the algorithm can be improved by using a signed binary representation of the scalar m and computing the sums and differences of various 2-powers. For more information, we refer the reader to* [HPS08, §5.3.1].

Since $E(K)$ is a group, we obtain the usual group-theoretic definitions. In particular, the *order of an elliptic curve* $E(K)$, denoted $\#E(K)$, is given by the number of elements in the group. We note that when K is not a finite field, the group $E(K)$ may possibly have infinite order. The *order of a point* $P \in E$ is defined to be the smallest positive integer m such that $mP = \mathcal{O}$. If no such integer m exists, the point is said to have infinite order. For groups of finite order, the order of the

point always divides the order of the group. As we are interested in elliptic curves over finite fields, our elliptic curve group will always have finite order. The following important result by Hasse gives an upper and lower bound on the order of an elliptic curve over a finite field.

Theorem 4 (Hasse) *Let E be an elliptic curve defined over a finite field F_q of characteristic* $\text{char}(F_q) = p$. *The order of $E(F_q)$ is given by*

$$\#E(F_q) = q + 1 - t,$$

where $|t| \leq 2\sqrt{q}$ and the value $t = q + 1 - \#E(F_q)$ is called the trace of Frobenius for $E(F_q)$.

We conclude this section with a few remarks about the chord-and-tangent rule. While the group law given in Definition A.4 may seem very abstract, it follows naturally from the theory of divisors, discussed in Section A.2, and the Picard group. We refer the reader to the Doctoral thesis of Déchène in [Déc05, §3.4] for an excellent explanation of this concept.

It turns out that the addition law for elliptic curves does not always work when the Weierstrass equation has multiple roots. To be sure that our elliptic curve $E(K)$ is an abelian group, we require that the Weierstrass equation

$$y^2 = x^3 + Ax + B = (x - \alpha_1)(x - \alpha_2)(x - \alpha_3),$$

has distinct roots $\alpha_1, \alpha_2, \alpha_3 \in K$. A *smooth (or non-singular) cubic curve* is defined to be a curve with cubic equation that contains no multiple roots. In order to obtain a smooth curve, we must ensure that the *discriminant* of the curve, given by

$$\Delta = 4a^3 + 27b^2,$$

is non-zero. This illustrates the requirement that our elliptic curve E be defined by a smooth cubic curve, as in Definition A.1. A detailed explanation is beyond the focus of our work. For more information, we refer the reader to [Was08, §2.4]. Throughout the rest of our work, we always assume that we are working with a smooth elliptic curve.

A.1.3 Torsion Points

The torsion points of an elliptic curve are a fundamental part of the definition of a bilinear pairing. Consequently, the definitions and results of this section are essential to our discussion of pairings in Section A.3.

Let E be an elliptic curve over K and m be a positive integer. The *torsion points* of an elliptic curve are the points of finite order. A point $P \in E$ is called an *m-torsion point* if $mP = \mathcal{O}$. In other words, P is an m-torsion point if it has order dividing m. The set of all m-torsion points

$$E[m] = \{P \in E(\overline{K}) \mid mP = \mathcal{O}\}$$

forms a subgroup of $E(\overline{K})$, called the *m-torsion subgroup*. The m-torsion subgroup is simply the kernel of the multiplication-by-m map on the elliptic curve $E(\overline{K})$.

Remark A.3 *It may often happen that $E[m] \subsetneq E(K)$ and thus we define $E[m] = E(\overline{K})[m]$ to be the full m-torsion subgroup. When referring to the m-torsion points of $E(K)$, we will write $E(K)[m]$. In Section A.3 we will show how to find the smallest extension field $L \supseteq K$ such that $E[m] \subseteq E(L)$.*

The following statements illustrate some of the important results related to m-torsion subgroups. For proofs of these results, we refer the reader to [Was08, Theorem 3.2].

Theorem 5 *Let E be an elliptic curve over K and let m be a positive integer.*

1. If $\mathrm{char}(K) = 0$ or m is prime to $\mathrm{char}(K)$, then

$$E[m] \simeq \mathbb{Z}_m \times \mathbb{Z}_m.$$

2. If $\mathrm{char}(K) = p$ and $p \mid m$, we write $m = p^r m'$ with $p \nmid m'$. Then either

$$E[m] \simeq \mathbb{Z}_{m'} \times \mathbb{Z}_{m'} \qquad \text{or} \qquad E[m] \simeq \mathbb{Z}_m \times \mathbb{Z}_{m'}.$$

Remark A.4 *We note that for the first case, we can choose a basis $\{P_1, P_2\}$ for $E[m]$, such that every point can be expressed in the form $aP_1 + bP_2$ for distinct $a, b \in \mathbb{Z}_m$.*

Furthermore, if m is a prime, then $E[m]$ may be thought of as a 2-dimensional vector space over the field $\mathbb{Z}_m$.

Corollary A.1 *If $\mathrm{char}(K) = p$ for a prime p, then for all integers $r \geq 1$, either*

$$E[p^r] \simeq \mathbb{Z}_{p^r} \qquad \text{or} \qquad E[p^r] \simeq \{\mathcal{O}\}.$$

Using the results of the corollary above, we can classify the elliptic curves into two distinct categories. Suppose we have an elliptic curve E over a field K of prime characteristic p. If $E[p^r] \simeq \{\mathcal{O}\}$ we say that the curve is *supersingular* and if $E[p^r] \simeq \mathbb{Z}_{p^r}$ we say that the curve is *ordinary*. This classification is of the utmost importance when using elliptic curves for cryptographic purposes, as we will see in Section A.3. When considering elliptic curves over finite fields, we have an alternate definition relating the trace of Frobenius for $E(F_q)$ to the concept of supersingularity, as shown in the following definition from [BSe05, §IX.10].

Definition A.5 *Let E be an elliptic curve over the finite field F_q of characteristic $\mathrm{char}(F_q) = p$, with order $\#E(F_q) = q+1-t$. We say that E is a supersingular curve if one of the following equivalent conditions holds:*

1. *$p \mid t$ (i.e. $\#E(F_q) \equiv 1 \mod p$).*
2. *E has no points of order p in $\overline{F_q}$.*
3. *The endomorphism ring of E over $\overline{F_q}$ is non-commutative.*

Otherwise, we say that E is an ordinary elliptic curve.

We stress that singular curves, briefly discussed in the previous section, and supersingular curves are completely unrelated concepts and should not be confused due to their unfortunate nomenclatural similarities. We conclude this section with an example illustrating the torsion subgroup of an elliptic curve.

Example A.3 *Let E be the elliptic curve over F_{31} defined by*

$$y^2 = x^3 + 11$$

and let $m = 5$. *Since* 5 *is prime to* $\text{char}(F_{31}) = 31$, *we have* $E[5] \simeq \mathbb{Z}_5 \times \mathbb{Z}_5$. *In fact, for this particular example, we can say more about the* 5*-torsion subgroup. Recall from Example A.1 that the group* $E(F_{31})$ *has order* $\#E(F_{31}) = 25$. *Consequently, we know that either* $E(F_{31}) \simeq \mathbb{Z}_{25}$ *or* $E(F_{31}) \simeq \mathbb{Z}_5 \times \mathbb{Z}_5$. *We can partition the group into* 6 *subgroups of order* 5, *each containing distinct non-identity points, given by*

$$\{\mathcal{O}, (11,3), (19,9), (19,22), (11,28)\}, \{\mathcal{O}, (24,3), (2,9), (2,22), (24,28)\},$$
$$\{\mathcal{O}, (27,3), (10,9), (10,22), (27,28)\}, \{\mathcal{O}, (15,21), (26,17), (26,14), (15,10)\},$$
$$\{\mathcal{O}, (13,21), (6,17), (6,14), (13,10)\}, \{\mathcal{O}, (3,21), (30,17), (30,14), (3,10)\}.$$

Since $E(F_{31})$ *contains no points of order* 25, *we can conclude that* $E(F_{31}) \simeq \mathbb{Z}_5 \times \mathbb{Z}_5$ *and thus we have that* $E(F_{31}) = E[5]$.

A.1.4 Rational Functions on Elliptic Curves

This final subsection deals with rational functions on elliptic curves. In order to comprehend the definitions of the Weil and the Tate pairings, we need to be able to evaluate a rational function at point on the curve and we require a special map that sends a function to its order of vanishing at a given point.

Definition A.6 *Let* E *be an elliptic curve over a field* K *defined by the Weierstrass equation* $y^2 = x^3 + Ax + B$. *The field of fractions of* $\overline{K}[x,y]/ < f >$ *where* $f(x,y) = x^3 + Ax + b - y^2 = 0$, *denoted* $\overline{K}(E)$, *is called the* field of rational functions of E.

We evaluate a rational function $f(x,y) \in \overline{K}(E)$ at an elliptic curve point $P = (x,y) \in E$ in the most intuitive way, by simply substituting the x- and y-coordinates into the function. Unfortunately, evaluating a rational function at the point-at-infinity $\mathcal{O}$ is a bit trickier. A common solution is to convert the rational function to homogeneous coordinates (as in Definition A.1) and evaluate the function at the basepoint $\mathcal{O} = [0,1,0]$. For the purposes of our work, we can always modify our computations so that we do not have to evaluate a function at $\mathcal{O}$. We'll discuss this further in Section A.4.5.

We say that the function has a *zero* at the point $P \in E$ if $f(P) = 0$ and a *pole* at P if $f(P) = \infty$. The concept of zeros and poles of a function in $\overline{K}(E)$ is

analogous to that of real functions. For example, suppose that we have the real function $g(x) = \frac{(x-1)^2}{(x+1)^3} \in \mathbb{R}[x]$. We say the function g has a *zero* of order 2 at the point $x = 1$ and a *pole* of order 3 at the point $x = -1$.

Remark A.5 *We note that a rational function $f \in \overline{K}(E)^\times$ has a finite number of zeros and poles, as shown in* [Sil86, Proposition II.1.2].

As algebraists, we would like to keep track of the zeros and poles of a rational function. In Section A.2, we will investigate a powerful tool from algebraic geometry that provides a convenient way of cataloguing the zeros and poles of a function along with their associated multiplicities. In preparation for this algebraic tool, we define a special map on the function field $\overline{K}(E)$ that will indicate whether a given function $f \in \overline{K}(E)$ has a pole or a zero at the point $P \in E$ by mapping f to its order of vanishing at P. More specifically, given a non-zero function $f \in \overline{K}(E)$, we define the map $ord_P : \overline{K}(E) \to \mathbb{Z} \cup \infty$ as follows:

$$ord_P(f) = \begin{cases} n > 0 \text{ if } f \text{ has a zero of order } n \text{ at } P. \\ n < 0 \text{ if } f \text{ has a pole of order } n \text{ at } P. \\ 0 \text{ if } f \text{ has neither a zero nor a pole, but is defined at } P. \\ \infty \text{ if } f = 0. \end{cases}$$

Remark A.6 *For a more thorough exposition of the ord_P map, including a discussion of discrete valuations, we refer the reader to* [Sil86, §II.1].

A.2 Divisors

In the previous section, we alluded to an efficient method for the recording the zeros and poles of a function. Our so-called "bookkeeper" is what's known as a divisor, and it plays an even more significant role in the definition of bilinear pairings. We begin with a few elementary definitions, adapted from [Sil86] and [Was08].

A.2.1 Definitions

Throughout this section, we take E to be an elliptic curve over a field K and write E for $E(\overline{K})$. The following results hold true for any algebraic curve C and have interesting implications in the study of algebraic varieties and Jacobians of curves. We limit our discussion to elliptic curves as they are of most relevance to our work, but the interested reader is referred to [Déc05] for a thorough treatment of these concepts.

For each point $P \in E$, we define a symbol (P). A *divisor* D is the formal sum of scalar multiples of these symbols,

$$\sum_{P \in E} n_P(P)$$

where $n_P \in \mathbb{Z}$ and $n_P = 0$ for all but finitely many points $P \in E$. In other words, a divisor is an element of the free abelian group generated by the points on E with identity $\sum_{P \in E(\overline{K})} 0(P) = \mathbf{0}$ and addition rule

$$\sum_{P \in E} m_P(P) + \sum_{P \in E} n_P(P) = \sum_{P \in E} (m_P + n_P)(P),$$

which we call the *divisor group* of E, denoted $\mathrm{Div}(E)$.

The *degree* of a divisor D is given by the sum of integers

$$\deg(D) = \sum_{P \in E} n_P.$$

The *sum* of a divisor D is given by

$$\mathrm{sum}(D) = \sum_{P \in E} n_P P \in E(K)$$

and corresponds to a point on the elliptic curve $E(K)$.

The divisors of degree zero are of particular interest to our work. They form a subgroup of $\mathrm{Div}(E)$, denoted by

$$\mathrm{Div}^0(E) = \{D \in \mathrm{Div}(E) \mid \deg(D) = 0\}.$$

The *support* of a divisor $D = \sum_{P \in E} n_p(P)$ is the collection of points P such that $n_P \neq 0$. More formally,

$$\text{Supp}(D) = \{P \in E \mid n_P \neq 0\}.$$

If D and D' are divisors such that $\text{Supp}(D) \cap \text{Supp}(D') = \varnothing$, that is, the divisors have disjoint supports, we say that D is *prime* to D'.

A.2.2 Divisors of Functions

In Section A.1.4, we defined a map ord_P that sent a rational function $f \in \overline{K}(E)$ on an elliptic curve E to its order of vanishing at the point $P \in E$. Using this map, we can define the divisor associated with a function, which we will call the *divisor of the function.*

Definition A.7 *Let E be an elliptic curve defined over K. The* divisor of a function *$f \in \overline{K}(E)^\times$ is given by*

$$\text{div}(f) = \sum_{P \in E} ord_P(f)(P).$$

With this definition, we see how the divisor of a function provides a record of the zeros and poles of the function, along with their associated multiplicities. The divisor of a function has a number of useful properties, as outlined in the proposition below and proven in [Sil86, Proposition II.3.1].

Proposition A.1 *Let E be an elliptic curve defined over K and f, g be functions from $\overline{K}(E)^\times$. Then,*

1. $\text{div}(f) = 0$ *if and only if* $f \in \overline{K}^\times$.
2. $\text{div}(f \cdot g) = \text{div}(f) + \text{div}(g)$.
3. $\text{div}\left(\frac{f}{g}\right) = \text{div}(f) - \text{div}(g)$.

Remark A.7 *From Proposition A.1 above, we see that a function $f \in \overline{K}(E)^\times$ satisfies $\text{div}(f) = 0$ if and only if f is a constant in $\overline{K}^\times$. If $\text{div}(f) = \text{div}(g)$, then $\text{div}(\frac{g}{f}) = 0$ and g is a constant multiple of f. Thus, given $\text{div}(f)$, the function f is unique up to multiplication by a non-zero constant $c \in \overline{K}^\times$.*

So we have just defined the concept of a divisor of a function and have seen some of the interesting properties of these divisors. We know how to evaluate a function at a given point on an elliptic curve (as seen in Section A.1.4), but we have yet to determine how these functions will be evaluated at a given divisor.

Definition A.8 *Let E be an elliptic curve defined over K. Given a divisor $D = \sum_{P\in E} n_P(P) \in Div(E)$ and a function $f \in \overline{K}(E)^\times$ such that D is prime to* $\mathrm{div}(f)$*, we define*

$$f(D) = \prod_{P\in E} f(P)^{n_P} = \prod_{P\in Supp(D)} f(P)^{n_P}.$$

From the remark above, we may wonder to what extent the functions f and $g = cf$, $c \in \overline{K}^\times$ are equivalent, when evaluated at the same divisor D. We have

$$g(D) = \prod_{P\in E)} (cf)(P)^{n_P} = \prod_{P\in E} c^{n_P} f(P)^{n_P} = c^{\Sigma n_P} \prod_{P\in E} f(P)^{n_P} = c^{\Sigma n_P} f(D)$$

and thus if $D \in \mathrm{Div}^0(E)$, we have $g(D) = f(D)$. From this point forward, if it is clear that we are evaluating functions at divisors of degree zero, we will treat $f = g$.

With the ability to evaluate functions at divisors, we obtain a wonderful result that will be used to prove several properties of pairings in Section A.3.

Proposition A.2 (Weil Reciprocity Law) *Let E be an elliptic curve defined over K. If $f, g \in \overline{K}(E)^\times$ are functions such that* $\mathrm{div}(f)$ *is prime to* $\mathrm{div}(g)$*, then*

$$f(\mathrm{div}(g)) = g(\mathrm{div}(f)). \tag{A.2.1}$$

As the proof of the Weil reciprocity law requires a significant amount of busy work, we refer the reader to [BSe05, Theorem IX.3]. We provide an example to illustrate the beauty of this result below.

Example A.4 *Let E be the elliptic curve over F_{31} defined by*

$$y^2 = x^3 + 11$$

and let $f, g \in \overline{K}(E)^\times$ be rational functions defined by $f(x,y) = \dfrac{x-11}{x-6}$ and $g(x,y) =$

$\frac{x-13}{x-3}$. *We have*

$$\operatorname{div}(f) = D_{(11,3)} + D_{(11,28)} - D_{(6,14)} - D_{(6,17)}$$

and

$$\operatorname{div}(g) = D_{(13,10)} + D_{(13,21)} - D_{(3,21)} - D_{(3,10)},$$

where $\operatorname{div}(f)$ *and* $\operatorname{div}(g)$ *have disjoint supports. We see that*

$$\begin{aligned}
f(\operatorname{div}(g)) &= f(D_{(13,10)} + D_{(13,21)} - D_{(3,21)} - D_{(3,10)}) \\
&= f(13,10) \cdot f(13,21) \cdot f(3,21)^{-1} \cdot f(3,10)^{-1} \\
&= \left(\frac{13-11}{13-6}\right) \cdot \left(\frac{13-11}{13-6}\right) \cdot \left(\frac{3-11}{3-6}\right)^{-1} \cdot \left(\frac{3-11}{3-6}\right)^{-1} \\
&= \left(\frac{13-11}{13-6}\right)^{2} \cdot \left(\frac{3-11}{3-6}\right)^{-2} \\
&= \frac{(13-11)^2 \cdot (3-6)^2}{(13-6)^2 \cdot (3-11)^2}
\end{aligned}$$

and

$$\begin{aligned}
g(\operatorname{div}(f)) &= g(D_{(11,3)} + D_{(11,28)} - D_{(6,14)} - D_{(6,17)}) \\
&= g(11,3) \cdot f(11,28) \cdot f(6,14) \cdot f(6,17) \\
&= \left(\frac{11-13}{11-3}\right) \cdot \left(\frac{11-13}{11-3}\right) \cdot \left(\frac{6-13}{6-3}\right)^{-1} \cdot \left(\frac{6-13}{6-3}\right)^{-1} \\
&= \left(\frac{11-13}{11-3}\right)^{2} \cdot \left(\frac{6-13}{6-3}\right)^{-2} \\
&= \frac{(11-13)^2 \cdot (6-3)^2}{(11-3)^2 \cdot (6-13)^2} \\
&= \frac{(13-11)^2 \cdot (3-6)^2}{(13-6)^2 \cdot (3-11)^2} \\
&= f(\operatorname{div}(g)),
\end{aligned}$$

as required by the Weil reciprocity law.

As seen in the example above, we can easily find the divisor of a function by simply determining the zeros and poles of the function. While all rational functions

may be associated with some (not necessarily unique) divisor, not all divisors may be associated with a function and in general, it is often challenging to find the function associated with these divisors. In the next section, we define the type of divisors that may be associated with functions and in Section A.4.5, we look at an efficient algorithm to compute these functions.

A.2.3 Principal Divisors

As discussed in the previous section, not all divisors may be associated with a function and thus the set of divisors of functions is a proper subset of the entire collection of divisors. From Proposition A.1, it is easy to see that the set of divisors of functions actually forms a subgroup of the group of divisors. We give a name to this subgroup and its elements, as they play an important role in our understanding of pairings.

Definition A.9 *A divisor $D \in Div(E)$ is called a* principal divisor *if there exists a function $f \in \overline{K}(E)^\times$ such that $D = \mathrm{div}(f)$. We denote the group of principal divisors on an elliptic curve E by*

$$\mathrm{Princ}(E) = \{D \in \mathrm{Div}(E) \mid D \text{ is principal}\}.$$

From this point on, we will refer to divisors of functions as principal divisors. The following theorem, due to Abel and Jacobi, tells us exactly when a divisor $D \in \mathrm{Div}(E)$ is a principal divisor.

Theorem 6 (Abel-Jacobi) *Let E be an elliptic curve over K and let $D \in Div(E)$ be a divisor $D = \sum_{P \in E} n_P(P)$. Then,*

$$D \text{ is principal if and only if } \deg(D) = 0 \text{ and } sum(D) = \mathcal{O}.$$

Proof: For a proof and thorough discussion of the Abel-Jacobi theorem, we refer the reader to [Déc05, §3.3.5]. ∎

The Abel-Jacobi theorem illustrates a number of interesting results. In particular, we see that only divisors of degree zero may be associated with a function. Since

all principal divisors have degree zero and the principal divisors form a subgroup of the divisors, we have that $\text{Princ}(E)$ is a subgroup of $\text{Div}^0(E)$. Before considering other important results of this powerful theorem, we illustrate the Abel-Jacobi theorem and the nice properties of principal divisors with the following example.

Example A.5 *Suppose that P, Q and $R = P + Q$ are points on an elliptic curve $E(K)$. Let $l(x, y) \in \overline{K}(E)^\times$ be the equation of the line through P and Q. The function l has zeros at P, Q and $-R$. Since $\text{div}(l)$ is a principal divisor, the Abel-Jacobi theorem implies that it has degree zero and sum$(\text{div}(l)) = \mathcal{O}$. Thus, we have*

$$\text{div}(l) = (P) + (Q) + (-R) - 3(\mathcal{O}).$$

Similarly, if $v(x, y)$ is the equation of the vertical line through the point R, then the function v has divisor

$$\text{div}(v) = (R) + (-R) - 2(\mathcal{O}).$$

Given the properties above, we have

$$\begin{aligned} \text{div}(\frac{l}{v}) &= \text{div}(l) - \text{div}(v) \\ &= \{(P) + (Q) + (-R) - 3(\mathcal{O})\} \\ &\quad - \{(R) + (-R) - 2(\mathcal{O})\} \\ &= (P) + (Q) - (R) - (\mathcal{O}). \end{aligned}$$

Furthermore, we see that

$$\deg\left(\text{div}\left(\frac{l}{v}\right)\right) = 0 \quad \textit{and} \quad \textit{sum}\left(\text{div}\left(\frac{l}{v}\right)\right) = P + Q - R - \mathcal{O} = \mathcal{O},$$

and the principal divisor $\text{div}(\frac{l}{v})$ indeed satisfies the Abel-Jacobi theorem.

An important corollary of the Abel-Jacobi theorem helps us to define a relation on the set of divisors.

Definition A.10 *Let E be an elliptic curve and let $D_1, D_2 \in Div(E)$ be given. We say that D_1 is* linearly equivalent *to D_2, denoted $D_1 \sim D_2$, if there exists a non-zero*

function $f \in \overline{K}(E)^{\times}$ *such that* $D_1 = D_2 + \mathrm{div}(f)$. *This relation partitions* $\mathrm{Div}(E)$ *into sets of equivalence classes, called* divisor classes. *We will denote the equivalence class of the divisor* D_1 *by* $[D_1]$.

In particular, two divisors $D_1 = \sum_{P \in E} m_P(P)$ and $D_2 = \sum_{P \in E} n_P(P)$ are linearly equivalent if $D_1 - D_2$ is a principal divisor. The Abel-Jacobi theorem implies that $D_1 - D_2 = \sum_{P \in E} (m_P - n_p)(P)$ will be a principal divisor if and only if $\deg(D_1 - D_2) = 0$ and $\sum_{P \in E} (m_P - n_P) P = \mathcal{O}$. This result is summarized in the corollary below.

Corollary A.2 *Let* E *be an elliptic curve over* K *and let* $D_1, D_2 \in Div(E)$ *be given divisors, as defined above. Then*

$$D_1 \sim D_2 \text{ if and only if } \deg(D_1) = \deg(D_2) \text{ and } \sum_{P \in E} m_P P = \sum_{P \in E} n_P P.$$

We conclude this section with an example that ties together many of the ideas presented above to find the function corresponding to a principal divisor.

Example A.6 *Let* E *be the elliptic curve over* F_{31} *defined by*

$$y^2 = x^3 + 11.$$

Let P *correspond to the point* $(11, 3)$ *of order* 5 *on* $E(F_{31})$. *The divisor given by*

$$D = 5(P) - 5(\mathcal{O})$$

has degree zero and $sum(D) = \mathcal{O}$. *By the Abel-Jacobi theorem,* D *is a principal divisor and thus the divisor of some function* $f \in \overline{K}(E)^{\times}$. *In Example A.2, we showed that the tangent line at* P *is given by* $y = 14x + 4$. *From Example A.5 above, we know that*

$$\mathrm{div}(14x - y + 4) = 2(P) + (-2P) - 3(\mathcal{O}).$$

The vertical line through the point $2P = (19, 9)$ *is given by* $x - 19 = 0$ *and has divisor*

$$\mathrm{div}(x - 19) = (2P) + (-2P) - 2(\mathcal{O}).$$

Computing

$$\begin{aligned} 2\,\mathrm{div}(14x - y + 4) - 2\,\mathrm{div}(x - 19) &= \{4(P) + 2(-2P) - 6(\mathcal{O})\} \\ &\quad - \{2(2P) + 2(-2P) - 4(\mathcal{O})\} \\ &= 4(P) - 2(2P) - 2(\mathcal{O}) \end{aligned}$$

we have

$$D = (P) + 2(2P) - 3(\mathcal{O}) + 2\,\mathrm{div}(14x - y + 4) - 2\,\mathrm{div}(x - 19).$$

Similarly, we can easily determine the tangent line at $2P$, given by the equation $y = 24x + 18$, and its associated divisor

$$\mathrm{div}(24x - y + 18) = 2(2P) + (-4P) - 3(\mathcal{O}).$$

Since $P = -4P$, we have

$$\begin{aligned} D &= \mathrm{div}(24x - y + 18) + 2\,\mathrm{div}(14x - y + 4) - 2\,\mathrm{div}(x - 19) \\ &= \mathrm{div}\left(\frac{(24x - y + 18)(14x - y + 4)^2}{(x - 19)^2}\right) \end{aligned}$$

After some grueling calculations using the division algorithm, this function can be simplified to

$$\frac{(24x - y + 18)(14x - y + 4)^2}{(x - 19)^2} = 21x^2 + (10 - y)x + 7(1 - y).$$

Therefore

$$D = \mathrm{div}\left(21x^2 + (10 - y)x + 7(1 - y)\right).$$

A.3 Pairings

At long last, we arrive at the building block of our cryptographic protocols and the key to identity-based cryptography: the almighty pairing. The pairing is the proverbial "man behind the curtain" of identity-based cryptography. Since the landmark paper

of Boneh and Franklin, there has been a spur of interest in pairings and pairing-based cryptography, as this thesis will attest. Indeed, the pairing serves as the mathematical foundation to our work.

This section will build upon the elementary definition of a pairing, as given in Chapter 3, to reveal two of the most celebrated types of pairings: the *Weil pairing* and the *Tate pairing*. We give general definitions of these pairings and look at the properties that make them so desirable to cryptographers.

Throughout this section, we always work with an elliptic curve E over a field K of characteristic p. We also assume that m is a positive integer coprime to p such that $m \mid \#E$.

A.3.1 Preliminary Definitions

In Chapter 3, we described a cryptographic pairing as a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ for cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of large prime order q. For cryptographic purposes, the group $\mathbb{G}_1$ is often chosen to be a subgroup of the group of points of an elliptic curve. Before defining the Weil and Tate pairings, we build upon concepts from Section A.1 on elliptic curves and introduce a few definitions from [BSe05, §IX.3] to aid in our understanding of the pairings.

Recall from Theorem 5 that if $\gcd(m, p) = 1$, then the m-torsion subgroup $E[m]$ consisting of the points $P \in E(\overline{K})$ such that $mP = \mathcal{O}$ has structure

$$E[m] \cong \mathbb{Z}_m \times \mathbb{Z}_m.$$

The subgroup $E[m]$ is of exponent m, since $mP = \mathcal{O}$ for all $P \in E[m]$.

We define the subgroup mE by

$$mE = \{mP \mid P \in E\}.$$

Given two points $P_1, P_2 \in E$, we define the equivalence relation $P_1 \equiv P_2$ if and only if $P_1 - P_2 \in mE$. Now we may consider the quotient group E/mE to be the set of equivalence classes under this relation. This quotient group is also of exponent m.

The set of m^{th} *roots of unity* is given by

$$\mu_m = \left\{ x \in \overline{K}^\times \mid x^m = 1 \right\}$$

and forms a cyclic group of order m whose generators are called *primitive* m^{th} *roots of unity.* Let $L = K(\mu_m)$ be the extension field of K generated by the m^{th} roots of unity. We define the subgroup

$$(L^\times)^m = \{a^m \mid a \in L^\times\}.$$

Given two elements $a, b \in L^\times$, we define the relation of equivalence modulo m^{th} powers by $a \equiv b$ if and only if $\frac{a}{b} \in (L^\times)^m$. The quotient group $L^\times/(L^\times)^m$ corresponds to the set of equivalence classes under this relation.

Using the results from Section A.2 on divisors and the concepts above, we formally define the Weil and the Tate pairing.

A.3.2 The Weil Pairing

The Weil pairing enjoys several different, but equivalent definitions (see [CC90] or [How96]) in the mathematical community. We look at the more familiar version as seen in [BSe05, §IX.6] and discuss the nice cryptographic properties of the Weil pairing.

Definition of the Weil Pairing

In general, the Weil pairing maps a pair of m-torsion points to an m^{th} root of unity. Let L be a field extension of K such that $E[m] \subseteq E(L)$. Suppose we have a pair of points $P, Q \in E[m]$. Let D_P and D_Q be divisors with disjoint supports, such that $D_P \sim (P) - (\mathcal{O})$ and $D_Q \sim (Q) - (\mathcal{O})$. By the Abel-Jacobi Theorem, we have that $mD_P \sim m(P) - m(\mathcal{O})$ is a principal divisor and thus there exists a function $f_P \in \overline{L}(E)^\times$ such that $\mathrm{div}(f_P) = mD_P$. Similarly, there exists $f_Q \in \overline{L}(E)^\times$ such that $\mathrm{div}(f_Q) = mD_Q$. The Weil pairing can be formally defined in terms of these divisors and functions.

Definition A.11 *Let E be an elliptic curve defined over a field K of characteristic p, m be a positive integer such that $\gcd(m,p) = 1$ and L be a field extension of K such that $E(\overline{K})[m] \subseteq E(L)$. For points $P, Q \in E[m]$, the* Weil pairing of P and Q *is a map*

$$w_m : E[m] \times E[m] \to \mu_m \subseteq L^\times$$

given by

$$w_m(P,Q) = \frac{f_P(D_Q)}{f_Q(D_P)}.$$

Remark A.8 *The Weil pairing is often written as $e(P,Q)$. However, as this form is also used for the Tate pairing, we use $w(P,Q)$ to denote the Weil pairing and $e(P,Q)$ when speaking of a pairing in general.*

To be concrete, we must be sure that the value of the Weil pairing is independent of our choice of the divisors D_P and D_Q. That is, we would like the Weil pairing to be well-defined.

Proposition A.3 *The Weil pairing is well-defined.*

Proof: Let D_P and D'_P be elements of the equivalence class $[(P) - (\mathcal{O})]$ and D_Q and D'_Q be elements of $[(Q) - (\mathcal{O})]$ with D_P prime to D_Q and D'_P prime to D'_Q. By definition, $D'_P = D_P + \mathrm{div}(h_P)$ and $D'_Q = D_Q + \mathrm{div}(h_Q)$ for some $h_P, h_Q \in \overline{L}(E)^\times$. Then there exist functions $f'_P = f_P \cdot h_P^m$ and $f'_Q = f_Q \cdot h_Q^m \in \overline{L}(E)^\times$ such that $\mathrm{div}(f'_P) = mD'_P$ and $\mathrm{div}(f'_Q) = mD'_Q$. We have

$$\begin{aligned}
\frac{f'_P(D'_Q)}{f'_Q(D'_P)} &= \frac{f'_P(D_Q + \mathrm{div}(h_Q))}{f'_Q(D_P + \mathrm{div}(h_P))} \\
&= \frac{f'_P(D_Q) \cdot f'_P(\mathrm{div}(h_Q))}{f'_Q(D_P) \cdot f'_Q(\mathrm{div}(h_P))} \\
&= \frac{(f_P(D_Q) \cdot h_P(D_Q)^m) \cdot (f_P(\mathrm{div}(h_Q)) \cdot h_P(\mathrm{div}(h_Q))^m)}{(f_Q(D_P) \cdot h_Q(D_P)^m) \cdot (f_Q(\mathrm{div}(h_P)) \cdot h_Q(\mathrm{div}(h_P))^m)} \\
&= \frac{f_P(D_Q)}{f_Q(D_P)} \cdot \frac{h_P(D_Q)^m \cdot f_P(\mathrm{div}(h_Q))}{h_Q(D_P)^m \cdot f_Q(\mathrm{div}(h_P))} \cdot \frac{h_Q(\mathrm{div}(h_P))^m}{h_Q(\mathrm{div}(h_P))^m} \quad \text{from Weil reciprocity} \\
&= \frac{f_P(D_Q)}{f_Q(D_P)} \cdot \frac{h_P(D_Q)^m \cdot h_Q(\mathrm{div}(f_P))}{h_Q(D_P)^m \cdot h_P(\mathrm{div}(f_Q))} \quad \text{from Weil reciprocity} \\
&= \frac{f_P(D_Q)}{f_Q(D_P)} \cdot \frac{h_P(D_Q)^m \cdot (h_Q(mD_P))}{h_Q(D_P)^m \cdot (h_P(mD_Q))}
\end{aligned}$$

$$= \frac{f_P(D_Q)}{f_Q(D_P)} \cdot \frac{h_P(D_Q)^m \cdot h_Q(D_P)^m}{h_Q(D_P)^m \cdot h_P(D_Q)^m}$$
$$= \frac{f_P(D_Q)}{f_Q(D_P)}.$$

It remains to show that $w_m(P,Q) = \frac{f_P(D_Q)}{f_Q(D_P)}$ is an m^{th} root of unity. Indeed, we have

$$\begin{aligned} w_m(P,Q)^m &= \frac{f_P(D_Q)^m}{f_Q(D_P)^m} \\ &= \frac{f_P(mD_Q)}{f_Q(mD_P)} \\ &= \frac{f_P(mD_Q)}{f_Q(\mathrm{div}(f_P))} \\ &= \frac{f_P(mD_Q)}{f_P(\mathrm{div}(f_Q))} \quad \text{from Weil reciprocity law} \\ &= \frac{f_P(mD_Q)}{f_P(mD_Q)} \\ &= 1. \end{aligned}$$

Thus the value of the Weil pairing is independent of the choice of divisors D_P and D_Q, and so the Weil pairing is well-defined. ∎

Properties of the Weil Pairing

The following properties make the Weil pairing useful for cryptographers.

Proposition A.4 *The Weil pairing $w_m : E(L)[m] \times E(L)[m] \to \mu_m$ enjoys the following properties:*

1. **Bilinearity :** For all $P, P_1, P_2, Q, Q_1, Q_2 \in E(L)[m]$,

   (a) $w_m(P_1 + P_2, Q) = w_m(P_1, Q) w_m(P_2, Q)$.

   (b) $w_m(P, Q_1 + Q_2) = w_m(P, Q_1) w_m(P, Q_2)$.

   i.e. $w_m(aP, bQ) = w_m(P,Q)^{ab}$ for all $a, b \in \mathbb{Z}_m^\times$, $P, Q \in E(L)[m]$.

2. **Non-degeneracy**: For all $P \neq \mathcal{O} \in E(L)[m]$, there exists some $Q \in E(L)[m]$ such that $w_m(P,Q) \neq 1$ and for all $Q \neq \mathcal{O} \in E(L)[m]$, there exists some $P \in E(L)[m]$ such that $w_m(P,Q) \neq 1$.

3. **Alternating (or Identity)**: For all $P \in E(L)[m]$, $w_m(P,P) = 1$.

4. **Skew-symmetry (or Anti-symmetry):** If $P, Q \in E(L)[m]$, then

$$w_m(P,Q) = w_m(Q,P)^{-1}.$$

5. **Compatibility:** If $P \in E(L)[mn]$ and $Q \in E(L)[m]$, then

$$w_{mn}(P,Q) = w_m(nP,Q).$$

Proof: We provide a proof for property 1. For a proof of the other properties, see [Sil86, Prop. III.8.1].

1. We prove the case of linearity in the second position. The proof of linearity in the first position is similar. Suppose that $Q_1 + Q_2 = Q_3$ and for $1 \leq i \leq 3$, let $f_{Q_i} \in \overline{L}(E)^\times$ be the function with divisor $\mathrm{div}(f_{Q_i}) = m(Q_i) - m(\mathcal{O})$. From Example A.5, we see that there exists a function $g \in \overline{L}(E)^\times$ such that

$$(Q_3) - (\mathcal{O}) = (Q_1) - (\mathcal{O}) + (Q_2) - (\mathcal{O}) + \mathrm{div}(g).$$

Multiplying by m, we have

$$\begin{aligned} \mathrm{div}(f_{Q_3}) &= m(Q_3) - m(\mathcal{O}) \\ &= m(Q_1) - m(\mathcal{O}) + m(Q_2) - m(\mathcal{O}) + m\,\mathrm{div}(g) \\ &= \mathrm{div}(f_{Q_1}) + \mathrm{div}(f_{Q_2}) + \mathrm{div}(g^m) \end{aligned}$$

and thus

$$f_{Q_3} = f_{Q_1} f_{Q_2} g^m.$$

Suppose we have a divisor $D_P \sim (P) - (\mathcal{O})$ with support disjoint from Q_1, Q_2,

Q_3, $\mathcal{O}$. Then

$$\begin{aligned}
w_m(P, Q_1 + Q_2) &= w_m(P, Q_3) \\
&= \frac{f_P(D_{Q_3})}{f_{Q_3}(D_P)} \\
&= \frac{f_P(D_{Q_3})}{f_{Q_1} f_{Q_2} g^m(D_P)} \\
&= \frac{f_P((Q_1) - (\mathcal{O}) + (Q_2) - (\mathcal{O}) + \mathrm{div}(g))}{f_{Q_1}(D_P) \cdot f_{Q_2}(D_P) \cdot g^m(D_P)} \\
&= \frac{f_P(D_{Q_1}) \cdot f_P(D_{Q_2}) \cdot f_P(\mathrm{div}(g))}{f_{Q_1}(D_P) \cdot f_{Q_2}(D_P) \cdot g^m(D_P)} \\
&= \frac{f_P(D_{Q_1})}{f_{Q_1}(D_P)} \cdot \frac{f_P(D_{Q_2})}{f_{Q_2}(D_P)} \cdot \frac{g(mD_P)}{g(mD_P)} \quad \text{from Weil reciprocity law} \\
&= w_m(P, Q_1) w_m(P, Q_2).
\end{aligned}$$

Thus $w_m(P, Q_1+Q_2) = w_m(P, Q_1) w_m(P, Q_2)$ and we have linearity in the second position of the Weil pairing.

∎

Recall that we often have to look to the algebraic closure $\overline{K}$ to find the m-torsion points $E[m]$ and m^{th} roots of unity μ_m. As consequence of the Weil pairing, we have the following proposition, proven in [Was08, Cor. 3.11], that illustrates a connection between these two groups when all m-torsion points have coordinates in the base field K.

Proposition A.5 *If $E[m] \subseteq E(K)$, then $\mu_m \subseteq K$.*

This proposition will be useful when we define pairings for cryptographic applications in Section A.4. We conclude this section with a discussion of the Tate pairing.

A.3.3 The Tate Pairing

The Tate pairing (also known as the Tate-Lichtenbaum pairing) was first introduced to the cryptographic community by Frey and Rück in [FR94]. The Tate pairing

enjoys some advantages over the Weil pairing, as we will see in Section A.4.6, and as a result is often favoured by cryptographers.

We first give a general definition of the Tate pairing over a general field K, then discuss the cryptographic properties.

Definition of the Tate Pairing

In Section 3.1, we briefly discussed the concept of symmetric and asymmetric pairings. Recall that a symmetric pairing has the form $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. For example, the Weil pairing discussed in the previous section is a symmetric pairing. Conversely, an asymmetric pairing has the form $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. As we will soon see, the Tate pairing is an example of an asymmetric pairing and thus does not satisfy our definition of a pairing given in Section 3.1. However, the Tate pairing can be modified to satisfy the needs of practical applications, as will be seen in Section A.4.4.

Here, we let L denote the field extension of K generated by the m^{th} roots of unity. In general, the Tate pairing maps an m-torsion point on an elliptic curve $E(L)$ and an element of the quotient group $E(L)/mE(L)$ to an element of the quotient group $L^\times/(L^\times)^m$. Suppose we are given $P \in E(L)[m]$ and $Q \in E(L)$. We find divisors D_P and D_Q and a function f_P, as we did with the Weil pairing, and define the Tate pairing using these concepts.

Definition A.12 *Let E be an elliptic curve defined over a field K of characteristic p and m be a positive integer such that* $\gcd(m, p) = 1$. *For points $P \in E(L)[m]$ and $Q \in E(L)$, where $L = K(\mu_m)$, the* Tate pairing of P and Q *is a map*

$$t_m : E(L)[m] \times E(L)/mE(L) \rightarrow L^\times/(L^\times)^m$$

given by

$$t_m(P, Q) = f_P(D_Q).$$

Remark A.9 *The Tate pairing may also be written as $\langle P, Q \rangle_m$ in mathematical literature.*

It is important to note that Q does not represent an element of $E(L)$, but an equivalence class in $E(L)/mE(L)$. As a result, the value of the Tate pairing depends

on the particular equivalence class in which our point Q is contained. While the Tate pairing is not well defined as an element of $L^\times$, it can be shown to be well-defined as equivalence classes in $L^\times/(L^\times)^m$.

Proposition A.6 *The Tate pairing is well-defined.*

Proof: A proof for the well-definedness of the Tate pairing can be found in [BSe05, Lemmas IX.4 and IX.5]. ∎

Properties of the Tate Pairing

The Weil pairing does not share all of its properties with the Tate pairing. In particular, the Tate pairing does not always have the alternating or compatibility properties. We also note that "=" may refer to equivalence modulo m^{th} powers.

Proposition A.7 *The Tate pairing $t_m : E(L)[m] \times E(L)/mE(L) \to L^\times/(L^\times)^m$ enjoys the following properties:*

1. ***Bilinearity:*** *For all $P, P_1, P_2 \in E(L)[m]$ and $Q, Q_1, Q_2 \in E(L)/mE(L)$,*

   (a) $t_m(P_1 + P_2, Q) = t_m(P_1, Q)t_m(P_2, Q)$

   (b) $t_m(P, Q_1 + Q_2) = t_m(P, Q_1)t_m(P, Q_2)$.

   i.e. $t_m(aP, bQ) = t_m(P, Q)^{ab}$ *for all* $a, b \in \mathbb{Z}_m^\times$, $P, Q \in E(L)[m]$.

2. ***Non-degeneracy:*** *For all $P \neq \mathcal{O} \in E(L)[m]$, there exists some $Q \in E(L)$ such that $t_m(P, Q) \neq 1$ and for all $Q \notin mE(L)$, there exists some $P \in E(L)[m]$ such that $t_m(P, Q) \neq 1$.*

Proof: The bilinearity follows from the proof of the Weil pairing. For a proof of the non-degeneracy property, we refer the reader to [Hes04]. ∎

Remark A.10 *We note that the Weil and Tate pairings are families of maps, one for each value m that satisfies the given conditions.*

A.4 Pairings for Cryptography

Our motivation for studying pairings on elliptic curves is their application to cryptographic protocols, specifically for use in identity-based public key cryptography. For most practical applications, we consider elliptic curves over finite fields. Unfortunately, the previous definitions for the Weil and Tate pairing do not satisfy the requirements of our original cryptographic pairing definition, as given in Section 3.1. In this section, we discuss the embedding degrees of both the finite field and the elliptic curve and how they're related to the concept of distortion maps. We illustrate the complications that arise when the pairings from the previous section are used for cryptographic purposes and show how we can construct modified versions of the Weil and Tate pairing that suit our needs. We present Miller's algorithm, which satisfies the *computability* portion of the criteria for a cryptographic pairing and compare the advantages of the two different pairings when used for cryptography.

For the remainder of this section, we let E be an elliptic curve over a finite field F_q of characteristic $p > 0$ and m be an integer such that $\gcd(m, p) = 1$ and $m \mid \#E$.

A.4.1 Embedding Degrees

From Theorem 5, we see that our curve E has m^2 points of m-torsion, which may lie in some larger field. We define the *embedding degree of the curve $E(F_q)$ with respect to q and m* to be the smallest integer $k > 0$ such that

$$E(F_{q^k})[m] \cong \mathbb{Z}_m \times \mathbb{Z}_m.$$

Equivalently, we may define the embedding degree of the curve to be the smallest k such that $E[m] \subseteq E(F_{q^k})$, which is the extension field required by the Weil pairing in Definition A.11 in order to have all m-torsion points.

Similarly, Definition A.12 for the Tate pairing requires that the elliptic curve be defined over an extension field containing all m^{th} roots of unity. We define the *embedding degree of the field F_q* with respect to q and m to be the smallest integer $k > 0$ such that $\mu_m \subseteq F_{q^k}$. Equivalently, the embedding degree of the field is the smallest k such that $m \mid q^k - 1$.

Remark A.11 *We stress that both the embedding degree of the curve and the field may be viewed as a function $k(q, m)$ of q and m.*

We give an overview of the relationship between the embedding degree of the curve and the embedding degree of the field. The following theorem shows that in most cases, these values are equal.

Theorem 7 *Let E be an elliptic curve over a finite field F_q of characteristic p and let m be an integer coprime to p. Let k_F denote the embedding degree of the field and k_C denote the embedding degree of the curve. Then we have one of the following cases:*

1. $k_F = k_C = 1$
2. $k_F = 1$ *and* $k_C = m$
3. $k_F = k_C > 1$.

Proof: The proof is given in [HPS08, Prop. 5.45], with the first case following naturally from Proposition A.5, which in terms of embedding degrees, says that if $k_C = 1$ then $k_F = 1$. ∎

Remark A.12 *Since $k_C = k_F = k$ for all but the second case above, the general term* embedding degree *is often used to refer to the embedding degree of the field, in cryptographic literature. For clarity, we will always specify which embedding degree is used. In cryptographic literature, the embedding degree of the field is sometimes referred to as the* security multiplier *or as the* Tate embedding degree, *indicating its requirement in the definition of the Tate pairing. The embedding degree of the curve may often be referred to as the* MOV degree, *indicating the relation to the MOV reduction of [MOV93], or as the* Weil embedding degree, *indicating its requirement in the definition of the Weil pairing.*

From the definition, we see that we can determine the embedding degree of the field by simply finding the smallest integer k that satisfies $m \mid q^k - 1$. Equivalently, we may simply determine the order of q modulo m. As a side note, since q and m

are relatively prime, Euler's Theorem states that $q^{\varphi(m)} \equiv 1 \bmod m$, where φ is the Euler-phi function. Consequently, we have that $k \mid \varphi(m)$ and in the case of m prime, $k \mid m-1$, as mentioned in [BSe05, §IX.5]. Furthermore, if we have a Sophie Germain prime $m = 2t+1$ for a prime t, then $k \mid 2t$ implies that $k \in \{1, 2, t, 2t\}$ and thus we have very few values to check to determine the field embedding degree k.

Althougth we have no concrete way of directly calculating the embedding degree of the curve, we can deduce the value in most cases. Clearly, a necessary condition for $k_C = 1$ is that $m^2 \mid \#E(F_q)$. Thus if $m^2 \nmid \#E(F_q)$, we cannot have the first case of Theorem 7 and can distinguish between the second and third cases by computing k_F. Similarly, if $m > \sqrt{q}+1$, we know that $m^2 \nmid \#E(F_q)$ since Hasse's Theorem ensures that $\#E(F_q) \leq \left(\sqrt{q}+1\right)^2$. Thus our only concern is to distinguish between the first and second cases when $m^2 \mid \#E(F_q)$.

Fortunately, for most cryptographic applications, we choose the embedding degree of the field to be larger than 1. In [MOV93], Menezes, Okamoto and Vanstone proved that, for supersingular curves, the embedding degree of the curve (and thus the embedding degree of the field) is bounded by $k \leq 6$. This restraint poses a security threat to elliptic curve cryptosystems, by allowing the ECDLP on the supersingular curve $E(F_q)$ to be embedded into the DLP on the extension field F_{q^k}, which can be easily computed for small values of k. The so-called *MOV reduction* is achieved using the Weil pairing and represents the first time this important pairing was used for cryptographic purposes. A similar attack using the Tate pairing was proposed by Frey and Rück in [FR94]. Thus in most cryptographic applications we do not encounter the first and second cases of Theorem 7. However, we provide an example of the first case below.

Example A.7 *Let E be the elliptic curve over F_{31} defined by*

$$y^2 = x^3 + 11,$$

which was shown to have order $\#E(F_{31}) = 25$ in Example A.3. Choosing $m = 5$ coprime to 31, we see that $5 \mid 30$ and so the embedding degree of the field is $k_F = 1$. Indeed, we see that all 5 roots of $x^5 - 1 = 0$, specifically $\mu_5 = \{1, 2, 4, 8, 16\}$, lie in F_{31}. Since $m^2 = 25 \mid \#E(F_q)$, we cannot distinguish between the first and second

cases of Theorem 7. However, from Example A.3, we know that $E(F_{31}) \cong E(F_{31})[5]$ *and thus* $k_C = 1$.

A.4.2 Distortion Maps

In Section A.3, we described some of the nice cryptographic properties of the Weil and Tate pairings. Unfortunately, the alternating property of the Weil pairing actually works against us when using the pairing for cryptography. We discuss the issue in the context of the Weil pairing, although it also applies to the Tate pairing in some situations, as discussed in Section A.4.4.

Recall that a cryptographic pairing is defined over a cyclic group $\mathbb{G}_1$ with large prime order m. Since m is chosen to be prime, we may view $E[m]$ as a 2-dimensional vector space over $\mathbb{Z}_m$. We say that points P and Q are *linearly independent* if they form a basis for $E[m]$. Suppose that we choose $\mathbb{G}_1 \subseteq E[m]$ to be the cyclic subgroup generated by the point $P \in E(F_q)$ of order m. For all $Q, R \in \mathbb{G}_1$, there exists $a, b \in \mathbb{Z}_m$ such that $Q = aP$ and $R = bP$. From the bilinearity and alternating properties of the Weil pairing, we have

$$w_m(Q, R) = w_m(aP, bP) = w_m(P, P)^{ab} = 1,$$

and thus $w_m(Q, R) = 1$ regardless of our choice of Q and R. Under these circumstances, the Weil pairing does not satisfy the non-degeneracy nor the strong non-degeneracy property (discussed in Section 3.1) and thus cannot be used for cryptographic applications. In order to obtain a non-trivial value, we must compute the pairing of two linearly independent points. This idea is summarized in the following proposition from [HPS08, Prop 5.45].

Proposition A.8 *Let* E *be an elliptic curve over a finite field* F_q *of characteristic* p *and let* $m \geq 3$ *be a prime different from* p *dividing* $\#E$*, so that we may view* $E[m]$ *as a 2-dimensional vector space over* $\mathbb{Z}_m$*. Given* $P, Q \in E[m]$*, the following statements are equivalent:*

1. P and Q are linearly independent,
2. $w_m(P, Q)$ is a primitive m^{th} root of unity,

3. $w_m(P,Q) \neq 1$.

For the case where $E[m] \subseteq E(F_q)$ (i.e. $k_F = k_C = 1$) we have $E(F_q)[m] \cong \mathbb{Z}_m \times \mathbb{Z}_m$ and can easily find linearly independent points $P, Q \in E(F_q)$ of prime order m. However, when $E[m] \subsetneq E(F_q)$, we have $E(F_q)[m] \cong \mathbb{Z}_m$ and thus all points in $E(F_q)[m]$ are linearly dependent. In order to find linearly independent points, we must look to the extension field $E(F_{q^k})$ with $E[m] \subseteq E(F_{q^k})$.

In [Ver01], Verheul proposed the idea of applying an endomorphism ϕ (defined over F_{q^k}) on the elliptic curve E with the property that $\phi(E(F_q)) \subsetneq E(F_q)$, known as a *distortion map* or a *non-rational endomorphism*, to find these linearly independent points. Distortion maps have taken several definitions in cryptographic literature. We give a more general definition than those given in [BSe05, HPS08], so that it may be applied to both the Weil and the Tate pairings.

Definition A.13 *Let E be an elliptic curve over a finite field F_q of characteristic p, $m \neq p$ be a prime such that $m \mid \#E$ and $k > 1$ be the field embedding degree with respect to q and m. An* m-distortion map $\phi_m : E(F_q) \rightarrow E(F_{q^k})$ *is a non-rational endomorphism such that, for all $P \in E(F_q)[m]$, if $\phi_m(P) \notin E(F_q)$, then P and $\phi_m(P)$ form a basis for the vector space $E(F_{q^k})[m]$.*

The general idea is that, by mapping the point $P \in E(F_q)[m]$ to a linearly independent point $\phi_m(P) \notin E(F_q)$, we can modify our pairings to satisfy the strong non-degeneracy property required for cryptographic applications. The question that remains, is whether or not these distortion maps are available and over what type of curves. As it turns out, for cases relevant to cryptography (i.e. cases in which the embedding degree of the field is $k > 1$), there is a direct connection between supersingular curves and distortion maps.

In [Ver01, Theorem 11] and [Ver04, Theorem 6], Verheul shows that distortion maps (as given in Definition A.13) do not exist for ordinary (or non-supersingular) elliptic curves. Suppose we have an ordinary elliptic curve E over F_q of characteristic p and an m-distortion map ϕ_m such that, for some point $P \in E(F_q)[m]$, we have $\phi_m(P) \notin E(F_q)$. From the properties of supersingular curves given in Definition A.5, we see that since E is a non-supersingular curve, the endomorphism ring $End(E)$ is commutative. As a result, the q-power Frobenius map $\Phi \in End(E)$, which maps

$\Phi(P) = P$ for all $P \in E(F_q)$, should commute with the distortion map $\phi_m \in End(E)$. We have

$$\phi_m(P) = \phi_m(\Phi(P)) = \Phi(\phi_m(P)),$$

and thus $\phi_m(P) \in E(F_q)$, which contradicts the definition of our distortion map. Thus distortion maps do not exist for ordinary curves. Furthermore, in [Ver04, Theorem 5] Verheul proves that we can always find a distortion map for a supersingular curve E. We summarize these results in the following theorem.

Theorem 8 *Let E be an elliptic curve defined over a field F_q of characteristic p, $m \neq p$ be a prime such that $m \mid \#E$ and $k > 1$ be the field embedding degree with respect to q and m. There exists a distortion map ϕ_m (defined over F_{q^k}) for the curve E if and only if E is supersingular.*

Remark A.13 *In* [Ver01], *Verheul states that distortion maps can only exist if $E[m]$ is non-cyclic. This can only occur if the prime $m \neq \mathrm{char}(F_q)$, which explains the additional requirement that $m \neq p$. In [Ver04], Verheul shows that when the field embedding degree is $k = 1$, distortion maps may exist for ordinary curves, which explains the requirement that $k > 1$.*

So there are always distortion maps available, provided that we choose supersingular curves. In [BSe05, Table IX.1], Galbraith gives examples of distortion maps for some of the more popular choices of supersingular curves.

A.4.3 The (Modified) Weil Pairing over Finite Fields

Applying the m-distortion map from the previous section, we can construct a modified version of the Weil pairing that is suitable for cryptographic applications.

Definition A.14 *Let E be an elliptic curve over a finite field F_q of characteristic p, $m \neq p$ be a prime such that $m \mid \#E$ and $k > 1$ be the field embedding degree with respect to q and m. For points $P, Q \in E(F_q)[m]$, the* (modified) Weil pairing of P and Q *is a map*

$$\widehat{w}_m : E[m] \times E[m] \rightarrow \mu_m \subseteq F_{q^k}$$

given by

$$\widehat{w}_m(P,Q) = w_m(P, \phi_m(Q)).$$

Returning to our previous discussion with $\mathbb{G}_1 = \langle P \rangle$, the m-distortion map ensures that for all points $Q = aP$ and $R = bP$, the points Q and $\phi(R)$ are linearly independent and thus

$$\widehat{w}_m(Q,R) = w_m(Q, \phi(R)) = w_m(aP, \phi(bP)) = w_m(P, \phi(P))^{ab} \neq 1$$

unless $m \mid ab$. In the latter case, we have either $m \mid a$ and thus $Q = \mathcal{O}$ or $m \mid b$ and $R = \mathcal{O}$. Thus for all $Q, R \neq \mathcal{O} \in \mathbb{G}_1$, the modified Weil pairing is strongly non-degenerate, satisfying the definition for a cryptographic pairing. This idea is summarized in the following proposition from [HPS08].

Proposition A.9 *Let E be an elliptic curve over F_q, $P \in E(F_q)[m]$ be a point of prime order $m \geq 3$ and ϕ_m be an m-distortion map as defined above. For points Q, R that are multiplies of P,*

$$w_m(Q, \phi_m(R)) = 1 \text{ if and only if } Q = \mathcal{O} \text{ or } R = \mathcal{O}.$$

A.4.4 The (Modified) Tate Pairing over Finite Fields

Earlier we had mentioned that the Tate pairing is generally favoured by cryptographers. Unfortunately, our original definition of the Tate pairing requires several modifications to suit the requirements of our cryptographic pairing. Suppose that the embedding degree of the field is k, so that we are working with an elliptic curve over the extension field F_{q^k}.

First and foremost, we recall that the value of $t_m(P,Q)$ corresponds to an equivalence class in $F_{q^k}^{\times}/(F_{q^k}^{\times})^m$. Since pairing-based applications generally require an input and output of unique elements, this poses a problem for cryptographers. However, this problem is easily resolved by raising $t_m(P,Q)$ to the power $\frac{q^k-1}{m}$. Indeed, given $t_m(P,Q) = g(F_{q^k}^{\times})^m$, we have

$$t_m(P,Q)^{\frac{q^k-1}{m}} = \left(g(F_{q^k}^{\times})^m\right)^{\frac{q^k-1}{m}}$$

$$= g^{\frac{q^k-1}{m}}(F_{q^k}^{\times})^{q^k-1}$$
$$= g^{\frac{q^k-1}{m}},$$

obtaining a distinct element $h = g^{\frac{q^k-1}{m}}$ that corresponds to an m^{th} root of unity. This yields the *(reduced) Tate pairing,* $t_m(P,Q)^{\frac{q^k-1}{m}}$.

Recall that our cryptographic pairing from Section 3.1 maps two elements from the group $\mathbb{G}_1$ to an element of $\mathbb{G}_2$, while the Tate pairing is defined over two separate groups $E(F_{q^k})[m]$ and $E(F_{q^k})/mE(F_{q^k})$. In order to use the Tate pairing for our cryptographic applications, we must modify it to map a pair of elements from the same group. Although the groups $E(F_{q^k})[m]$ and $E(F_{q^k})/mE(F_{q^k})$ have the same number of elements, we can not necessarily use the elements of $E(F_{q^k})[m]$ as representatives for the equivalence classes in $E(F_{q^k})/mE(F_{q^k})$, as discussed in [BSe05, §IX.3]. More specifically, we must be sure that $E(F_{q^k})[m] \cap mE(F_{q^k}) = \{\mathcal{O}\}$. A simple argument shows that a necessary and sufficient condition for $E(F_{q^k})[m] \cap mE(F_{q^k}) = \{\mathcal{O}\}$ is that $E(F_{q^k})$ contains no points of order m^2. Indeed, given a non-trivial point $P \in E(F_{q^k})$ of order m^2, let $Q = mP$. Then $Q \in E(F_{q^k})[m]$ and $Q \in mE(F_{p^k})$ and thus the intersection is non-trivial. Conversely, suppose there exists a non-trivial point $Q \in E(F_{q^k})[m] \cap mE(F_{q^k})$. Since $Q \in mE(F_{q^k})$, there exists some non-trivial point $P \in E(F_{q^k})$ such that $Q = mP$. Since $Q \in E(F_{q^k})[m]$, we have $mQ = m(mP) = \mathcal{O}$ and thus P is a point of order m^2. It can be shown that if there exists a point $P \in E(F_{q^k})$ of order m^2, then we must have $m^3 \mid \#E(F_{q^k})$. So to use elements of $E(F_{q^k})[m]$ as representatives for classes in $E(F_{q^k})/mE(F_{q^k})$, we require that $m^2 \parallel \#E(F_{q^k})$.

Finally, we address the issue of the alternating property of the Tate pairing. While the Tate pairing does not satisfy this property in general, the reduced Tate pairing is alternating in the following situations:

1. If $P \in E(F_{q^k})[m] \cap mE(F_{q^k})$, then $t_m(P,P)^{\frac{q^k-1}{m}} = 1$, as shown in [BSe05, Lemma IX.13]. As discussed in the previous paragraph, we avoid this situation by requiring that $m^2 \parallel \#E(F_{q^k})$.

2. Suppose that the embedding degree of the field is $k = 1$. Recall from Theorem 7 that the embedding degree of the curve may be either 1 or m. We have the

following two subcases:

i) If the embedding degree of the curve is m, then there is a unique subgroup of order m in $E(F_q)$ and the reduced Tate pairing is strongly non-degenerate (i.e. $t_m(P,P)^{\frac{q^k-1}{m}} \neq 1$ for all points $P \in E(F_{q^k})[m]$), as shown in [BSe05, §IX.7.2] and [Was08, Lemma 5.4].

ii) If the embedding degree of the curve is 1, so that there are m^2 points of order m in $E(F_q)$, then the reduced Tate pairing may or may not be strongly non-degenerate, as illustrated in [BSe05, §IX.7.2].

3. If the embedding degree of the field (and thus the embedding degree of the curve) is $k > 1$, then the reduced Tate pairing is alternating for all points $P \in E(F_q)[m]$, as shown in [BSe05, Lemma IX.13].

In the case that the embedding degree of the field is $k = 1$, the condition $m^2 \nmid \#E(F_q)$ ensures that the embedding degree of the curve is m, and thus the reduced Tate pairing is strongly non-degenerate. However, as we often choose the embedding degree of the field to be $k > 1$, we must apply the distortion map to obtain strong non-degeneracy. As shown by Galbraith in [BSe05, Lemma IX.14], if $m^2 \parallel \#E(F_{q^k})$ and we have an m-distortion map such that $\phi_m(P) \notin E(F_q)$, then $t_m(P, \phi_m(P))^{\frac{q^k-1}{m}} \neq 1$ for any point $P \in E(F_q)[m]$ and thus the modified Tate pairing is strongly non-degenerate. Using this distortion map, we formally define the modified Tate pairing for cryptographic applications.

Definition A.15 *Let E be an elliptic curve over a finite field F_q of characteristic p, $m \neq p$ be a prime such that $m \mid \#E(F_q)$ and $k > 1$ be the field embedding degree with respect to q and m. Suppose that $m^2 \parallel \#E(F_{q^k})$. For points $P, Q \in E(F_{q^k})[m]$, the* (modified) Tate pairing of P and Q *is a map*

$$\widehat{t}_m : E(F_{q^k})[m] \times E(F_{q^k})[m] \to \mu_m$$

given by

$$\widehat{t}_m(P,Q) = t_m(P, \phi_m(Q))^{\frac{q^k-1}{m}}.$$

We have shown that the modified Weil and Tate pairings satisfy the bilinearity and strong non-degeneracy properties required for cryptographic pairings. It remains to show that they can be efficiently computed.

A.4.5 Miller's Algorithm

In 1985, Miller discovered an algorithm for computing pairings and wrote it up as a short note. Although it was never published, this note was possibly the most influential paper to the study of pairing-based cryptography. Miller's algorithm allowed for pairings to be implemented in practical applications, sparking an interest in pairings from the perspective of computer scientists. The first such implementation of the Weil pairing was in a paper by Kaliski [Kal87], where he developed a new pseudo-random bit generator using elliptic curves. In recent years, designing faster and more efficient algorithms for pairing calculation has been an active area of research.

The real challenge we face with pairing computation is finding the function f_P such that $\text{div}(f_P) = m(P) - m(\mathcal{O})$ and evaluating it for a given divisor. Miller's idea was to build this function inductively, similar to the Double-and-Add algorithm from Section A.1.2.

Given points $P, Q \in E$, we define $L_{P,Q}$ to be the equation of the line passing through the points P and Q. We handle the special cases as we did with point addition on elliptic curves in Section A.1.2. If $Q = P$, then $L_{P,P}$ is the equation of the tangent line to the curve at P. If $Q = \mathcal{O}$, then $L_{P,\mathcal{O}} = L_{P,-P}$ is the equation of the vertical line through the points P and $-P$. Finally, if both $P = Q = \mathcal{O}$, we set $L_{\mathcal{O},\mathcal{O}} = 1$.

The following lemma follows naturally from our definition of a divisor of a function.

Lemma A.1 *If $P \in E$, then*

$$\text{div}(L_{P,Q}) = (P) + (Q) + (-(P+Q)) - 3(\mathcal{O}).$$

We use the equations of these lines to construct the following function.

Definition A.16 *Let $P, Q \in E$. We define*

$$g_{P,Q} = \frac{L_{P,Q}}{L_{P+Q,\mathcal{O}}}.$$

Recall that we choose $\mathbb{G}_1 = E(F_{q^k})[m]$. In the case that P is a generator for $\mathbb{G}_1$, all points can be representated as $Q = aP$ for some $a \in \mathbb{Z}$. To make the algorithm more comprehensible, we follow the algorithm as given in [BSe05, §IX.8] and use the notation $[m]P$ for the multiplication-by-m map. We now consider the function above with respect to points $[i]P$ and $[j]P$ for $i, j \in \mathbb{Z}$, defined as

$$g_{[i]P,[j]P} = \frac{L_{[i]P,[j]P}}{L_{[i+j]P,\mathcal{O}}}.$$

Lemma A.2 *For a point $P \in E$, we have*

$$\mathrm{div}(g_{[i]P,[j]P}) = ([i]P) + ([j]P) - ([i+j]P) - (\mathcal{O}).$$

Proof: From Lemma A.1, we have

$$\begin{aligned} \mathrm{div}(L_{[i]P,[j]P}) &= ([i]P) + ([j]P) + (-[i+j]P) - 3(\mathcal{O}), \\ \mathrm{div}(L_{[i+j]P,\mathcal{O}}) &= ([i+j]P) + (\mathcal{O}) + (-[i+j]P) - 3(\mathcal{O}). \end{aligned}$$

The rest of the proof follows from the definition of $g_{[i]P,[j]P}$.

$$\begin{aligned} \mathrm{div}(g_{[i]P,[j]P}) &= \mathrm{div}(L_{[i]P,[j]P}) - \mathrm{div}(L_{[i+j]P,\mathcal{O}}) \\ &= \{([i]P) + ([j]P) + (-[i+j]P) - 3(\mathcal{O})\} \\ &\quad -\{([i+j]P) + (\mathcal{O}) + (-[i+j]P) - 3(\mathcal{O})\} \\ &= ([i]P) + ([j]P) - ([i+j]P) - (\mathcal{O}). \end{aligned}$$

Therefore the divisor of a function $g_{[i]P,[j]P}$ is given by $\mathrm{div}(g_{[i]P,[j]P}) = ([i]P) + ([j]P) - ([i+j]P) - (\mathcal{O})$. ∎

The results of the above lemma may be familiar, as we obtained similar results in Example A.5. We are now prepared to define the intermediate functions that we

will use in the algorithm to build the function f.

Definition A.17 *Let $P \in E$. We define f_n to be a function such that*

$$\mathrm{div}(f_n) = n(P) - ([n]P) - (n-1)(\mathcal{O}).$$

Note that our goal is to find a function f_P with divisor $m(P) - m(\mathcal{O})$, for some point P of order m. The above definition is unique up to constant multiplication, and so we may take $f_m = cf_P$ for some scalar $c \in \overline{K}^{\times}$. We can build the function f_m inductively using the following theorem.

Theorem 9 (Miller's formula) *Let $P \in E$. For all $i, j \in \mathbb{Z}$, we have*

$$f_{i+j} = f_i \cdot f_j \cdot g_{[i]P,[j]P}.$$

Proof: We are given divisors of functions f_n and $g_{P,Q}$ that satisfy

$$\begin{aligned} \mathrm{div}(f_n) &= n(P) - ([n]P) - (n-1)(\mathcal{O}), \\ \mathrm{div}(g_{[i]P,[j]P}) &= ([i]P) + ([j]P) - ([i+j]P) - (\mathcal{O}). \end{aligned}$$

From our definition of the function f_n, we have

$$\begin{aligned} \mathrm{div}(f_{i+j}) &= (i+j)(P) - ([i+j]P) - (i+j-1)(\mathcal{O}) \\ &= i(P) - ([i]P) - (i-1)(\mathcal{O}) \\ &\quad + j(P) - ([j]P) - (j-1)(\mathcal{O}) \\ &\quad + ([i]P) + ([j]P) - ([i+j]P) - (\mathcal{O}) \\ &= \mathrm{div}(f_i) + \mathrm{div}(f_j) + \mathrm{div}(g_{[i]P,[j]P}). \end{aligned}$$

Thus we have $f_{i+j} = f_i \cdot f_j \cdot g_{[i]P,[j]P}$, as required. ∎

Let $m = (m_s m_{s-1} \ldots m_1 m_0)_2$ for $m_i \in \{0, 1\}$ and $m_s = 1$ be the binary representation of m. Note that $s = \lfloor \log_2 m \rfloor$. We can use Miller's formula given above to build the function f_m inductively, as in the following algorithm.

Algorithm A.2 (Miller's Algorithm)

```
set f ⟵ 1, T ⟵ P and s = ⌊log_2 m⌋
for i = s − 1, s − 2, ..., 1, 0 do
{
  set f ⟵ f^2(D) · g_{T,T}(D) and T ⟵ [2]T  // Point Doubling
  if m_i = 1, then
  {
    set f ⟵ f(D) · g_{T,P}(D) and T ⟵ T + P  // Point Addition
  }
}
return f;
```

Remark A.14 *We note that we would like to evaluate the intermediate functions f_i at the divisor D. We have that $f_{i+j}(D) = f_i(D) \cdot f_j(D) \cdot g_{[i]P,[j]P}(D)$. We choose our divisor $D_P \sim (P) - (\mathcal{O})$ to be $D = (P + R) - (R)$ for some point R chosen at random from $E(K)$, such that the intermediate functions are defined and non-zero at the divisor. For example, we would not choose $R = -P$, since we would have to evaluate the functions at the point-at-infinity.*

We illustrate Miller's algorithm with the following example.

Example A.8 *Let E be the elliptic curve over F_{31} defined by*

$$y^2 = x^3 + 11.$$

From Example A.3, we know that

$$E(F_{31}) = E(F_{31})[5] \cong \mathbb{Z}_5 \oplus \mathbb{Z}_5$$

and the embedding degrees of both the field and the curve are equal to 1. As described in Section A.4.2, we do not require distortion maps to compute the Weil pairing $w_5(P, Q)$. We need only choose linearly independent points P and Q. Given linearly independent points $P = (11, 3)$ and $Q = (2, 9)$, we have divisors $D_P \sim (P) - (\mathcal{O})$ and $D_Q \sim (Q) - (\mathcal{O})$. Given a random point $R = (10, 22) \in E(F_{31})[5]$, we choose our divisors to be $D_P = (P - R) - (-R)$ and $D_Q = (Q + R) - (R)$ and compute the Weil

pairing as

$$\begin{aligned} w_5(P,Q) &= \frac{f_P(D_Q)}{f_Q(D_P)} \\ &= \frac{\frac{f_P(Q+R)}{f_P(R)}}{\frac{f_Q(P-R)}{f_Q(-R)}} \\ &= \frac{f_P(Q+R)}{f_P(R)} \cdot \frac{f_Q(-R)}{f_Q(P-R)}. \end{aligned}$$

We need functions $f_P, f_Q \in F_{31}[x]$ *with divisors* $\mathrm{div}(f_P) = 5(P) - 5(\mathcal{O})$ *and* $\mathrm{div}(f_Q) = 5(Q) - 5(\mathcal{O})$, *respectively. Recall from Example A.6 that*

$$f_P = 21x^2 + (10-y)x + 7(1-y).$$

Given $Q + R = (26, 14)$, *we compute*

$$\begin{aligned} \frac{f_P(Q+R)}{f_P(R)} &= \left(\frac{(21x^2 + (10-y)x + 7(1-y)) \mid_{(26,14)}}{(21x^2 + (10-y)x + 7(1-y)) \mid_{(10,22)}} \right) \\ &= \left(\frac{20}{4} \right) \bmod 31 \\ &= 5. \end{aligned}$$

We can compute f_Q *using Miller's algorithm. We see that* $m = 5 = (101)_2$ *and so* $s = 2$.

1. *We have* $i = 1,\ m_i = 0,\ f = 1$ *and* $T = Q$.

   (a) *Compute*

   $$\begin{aligned} f &= f^2 \cdot g_{Q,Q} \\ &= (1)^2 \cdot \left(\frac{11x - y + 18}{x - 24} \right) \\ &= \frac{11x - y + 18}{x - 24}. \end{aligned}$$

   (b) *Set* $T = 2Q$ *and* $i = 0$.

2. *We have* $i = 0,\ m_i = 1,\ f = \frac{11x-y+18}{x-24}$ *and* $T = 2Q$.

   (a) *Compute*

   $$f = f^2 \cdot g_{2Q,2Q}$$

$$= \left(\frac{11x - y + 18}{x - 24}\right)^2 \cdot \left(\frac{22x - y + 27}{x - 2}\right).$$

(b) Set $T = 4Q$. *Since* $m_0 = 1$, *we compute*

$$\begin{aligned} f &= f \cdot g_{4Q,Q} \\ &= \left(\left(\frac{11x - y + 18}{x - 24}\right)^2 \cdot \left(\frac{22x - y + 27}{x - 2}\right)\right) \cdot \frac{x - 2}{1} \\ &= \frac{(11x - y + 18)^2 (22x - y + 27)}{(x - 24)^2}. \\ &= f_Q. \end{aligned}$$

We first compute $P - R = (15, 21)$ *and* $-R = (10, 9)$. *Then*

$$\begin{aligned} \frac{f_Q(-R)}{f_Q(P - R)} &= \left(\frac{\frac{(11x-y+18)^2(22x-y+27)}{(x-24)^2}\,|_{(10,9)}}{\frac{(11x-y+18)^2(22x-y+27)}{(x-24)^2}\,|_{(15,21)}}\right) \\ &= \left(\frac{6}{23}\right) \bmod 31 \\ &= 7. \end{aligned}$$

Thus we have

$$\begin{aligned} w_5(P, Q) &= \frac{f_P(Q + R)}{f_P(R)} \cdot \frac{f_Q(-R)}{f_Q(P - R)} \\ &= 5 \cdot 7 \bmod 31 \\ &= 4. \end{aligned}$$

As a final step, we check that our solution $w_5(P, Q) = 4$ *is indeed a* 5^{th} *root of unity, by verifying that*

$$w_5(P, Q)^5 = 4^5 = 1.$$

A.4.6 Weil Pairing vs. Tate Pairing

In summary, we compare the cryptographic Weil and Tate pairings. We immediately notice the difference in the definition of the two pairings. While the Weil pairing is symmetric (i.e. $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$), the Tate pairing is asymmetric (i.e. $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$). For use with the cryptographic applications described throughout this thesis, we must impose some additional constraints to make the Tate pairing symmetric. In

addition, the Tate pairing is slightly more complicated to understand, as groups $\mathbb{G}_2$ and $\mathbb{G}_T$ refer to groups of cosets in $E(F_{q^k})/mE(F_{q^k})$ and $F_{q^k}^{\times}/(F_{q^k}^{\times})^m$, respectively. However, once we have a firm understanding of the Tate pairing, we see that it has numerous cryptographic advantages over the Weil pairing.

Since the Weil pairing maps a pair of linearly independent m-torsion points, we must work over the extension field $E(F_{q^k})$ such that $E[m] \subseteq E(F_{q^k})$, which requires the curve embedding degree k. For the Tate pairing, we work over the extension field $E(F_{q^k})$ such that $\mu_m \subseteq E(F_{q^k})$, which requires the field embedding degree k. Recall from Section A.4.1, that in most cases, the field and curve embedding degrees are equivalent. However in the case that the field embedding degree is $k_F = 1$ and the curve embedding degree is $k_C = m$, the Tate pairing has several advantages over the Weil pairing. The Tate pairing may be computed over the base field F_q, while the Weil pairing requires the much larger field F_{q^m}. In addition, the Tate pairing is strongly non-degenerate, as shown in Section A.4.2, while the Weil pairing is alternating and thus requires the use of a distortion map.

The most notable advantage of the Tate pairing over the Weil pairing is from a computational point of view. Although the Tate pairing requires an extra exponentiation to obtain an m^{th} root of unity, the user is only required to find a single function f_P, inducing a single execution of Miller's algorithm. In contrast, the Weil pairing requires two separate executions of Miller's algorithm to find both f_P and f_Q. For cryptographic applications, pairings are considered to be quite computationally expensive in comparison to elliptic curve multiplication or field exponentiations. Consequently, any computational benefits are considered to be very advantageous. Although some minor adjustments can be made to speed-up computations in each case, as shown in [GHS02], the Tate pairing can generally be more efficiently implemented and is thus favoured for practical cryptographic applications.

Bibliography

[ADR02] Jee Hea An, Yevgeniy Dodis, and Tal Rabin, *On the Security of Joint Signature and Encryption*, Advances in Cryptology - Proceedings of EUROCRYPT '02, LNCS 2332 (London, UK), Springer-Verlag, 2002, pp. 83–107.

[ARP02] Sattam S. Al-Riyami and Kenneth G. Paterson, *Tripartite Authenticated Key Agreement Protocols from Pairings*, Cryptology ePrint Archive, Report 2002/035, 2002.

[AST98] Giuseppe Ateniese, Michael Steiner, and Gene Tsudik, *Authenticated Group Key Agreement and Friends*, CCS '98: Proceedings of the 5th ACM conference on Computer and Communications Security (New York, NY, USA), ACM Press, 1998, pp. 17–26.

[BAA+07] Raghan Bhaskar, Daniel Augot, Cédric Adjih, Paul Mühlethaler, and Saadi Boudjit, *AGDH (Asymmetric Group Diffie Hellman) An Efficient and Dynamic Group Key Agreement Protocol for Ad hoc Networks*, Tech. report, Microsoft Research India, 2007.

[BB04] Dan Boneh and Xavier Boyen, *Secure Identity Based Encryption Without Random Oracles*, Advances in Cryptology - Proceedings of CRYPTO '04, LNCS 3152, Springer, 2004, pp. 443–459.

[BCEP03] Emmanuel Bresson, Olivier Chevassut, Abdelilah Essiari, and David Pointcheval, *Mutual Authentication and Group Key Agreement for Low-Power Mobile Devices*, In proceedings of the 5th IFIP-TC6/IEEE , MWCN 2003, 2003, pp. 59–62.

[BCP02] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval, *Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions*, Advances in Cryptology - Proceedings of EUROCRYPT '02, LNCS 2332 (London, UK), Springer-Verlag, 2002, pp. 321–336.

[BCPQ01] Emmanuel Bresson, Olivier Chevassut, David Pointcheval, and Jean-Jacques Quisquater, *Provably Authenticated Group Diffie-Hellman Key*

Exchange, CCS'01: Proceedings of the 8th ACM Conference on Computer and Communications Security, ACM New York, NY, USA, 2001, pp. 255–264.

[BD94] Mike Burmester and Yvo Desmedt, *A Secure and Efficient Conference Key Distribution System*, Advances in Cryptology - Proceedings of EUROCRYPT'94, LNCS 950, Springer-Verlag, 1994, pp. 275–286.

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway, *Relations Among Notions of Security for Public-Key Encryption Schemes*, Advances in Cryptology - Proceedings of CRYPTO 1998, LNCS 1462, Springer-Verlag, 1998, pp. 26–45.

[BDS03] Rana Barua, Ratna Dutta, and Palash Sarkar, *Extending Joux's Protocol to Multi Party Key Agreement*, Cryptology ePrint Archive, Report 2003/062, 2003, http://eprint.iacr.org.

[BF01] Dan Boneh and Matthew Franklin, *Identity-based Encryption from the Weil Pairing*, Advances in Cryptology - Proceedings of CRYPTO 2001, LNCS 2139, Springer-Verlag, 2001, pp. 219–229.

[BF03] ———, *Identity-based Encryption from the Weil Pairing*, SIAM Journal of Computing **32** (2003), no. 3, 586–615.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham, *Aggregate and Verifiably Encrypted Signatures from Bilinear Maps*, Advances in Cryptology - Proceedings of EUROCRYPT '03, LNCS 2656, 2003, pp. 416–432.

[BGR98] Mihir Bellare, Juan A. Garay, and Tal Rabin, *Fast Batch Verification for Modular Exponentiation and Digital Signatures*, Advances in Cryptology - Proceedings of EUROCRYPT '98, LNCS 1403, Springer-Verlag, 1998, pp. 236–250.

[BLMQ05] Paulo S. L. M. Barreto, Benoît Libert, Noel McCullagh, and Jean-Jacques Quisquater, *Efficient and Provably-Secure Identity-Based Signatures and Signcryption from Bilinear Maps*, Advances in Cryptology - Proceedings of ASIACRYPT '05, LNCS 3788, Springer-Verlag, 2005, pp. 515–532.

[BN03] Colin Boyd and Juan Manuel González Nieto, *Round-Optimal Contributory Conference Key Agreement*, Public Key Cryptography - In Proceedings of PKC 2003, LNCS 2567, Springer-Verlag, 2003, pp. 161–174.

[Boy97] Colin Boyd, *On Key Agreement and Conference Key Agreement*, Information Security and Privacy, Second Australasian Conference, LNCS 1270, Springer-Verlag, 1997.

[Boy03] Xavier Boyen, *Multipurpose Identity-Based Signcryption - A Swiss Army Knife for Identity-Based Cryptography*, Advances in Cryptology - Proceedings of CRYPTO 2003, LNCS 2729, Springer-Verlag Berlin, 2003, pp. 383–399.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway, *Authenticated Key Exchange Secure against Dictionary Attacks*, Advances in Cryptology - Proceedings of EUROCRYPT 2000, LNCS 1807, Springer, 2000, pp. 139–155.

[BR93] Mihir Bellare and Phillip Rogaway, *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*, CCS '93: Proceedings of the 1st ACM conference on Computer and Communications Security (New York, NY, USA), ACM, 1993, pp. 62–73.

[BR95] ———, *Provably Secure Session Key Distribution: The Three Party Case*, STOC '95: Proceedings of the 27th annual ACM symposium on Theory of computing (New York, NY, USA), ACM, 1995, pp. 57–66.

[BSe05] Ian F. Blake, Gadiel Seroussi, and Nigel Smart (eds.), *Advances in Elliptic Curve Cryptography*, London Mathematical Society Lecture Note Series 317, Cambridge University Press, New York, NY, USA, 2005.

[BW98] Klaus Becker and Uta Wille, *Communication Complexity of Group Key Distribution*, CCS '98: Proceedings of the 5th ACM conference on Computer and Communications Security (New York, NY, USA), ACM Press, 1998, pp. 1–6.

[BZ04] Joonsang Baek and Yuliang Zheng, *Identity-Based Threshold Signature Scheme from the Bilinear Pairings*, ITCC '04: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2 (Washington, DC, USA), IEEE Computer Society, 2004, p. 124.

[CC90] Leonard S. Charlap and Raymond Coley, *An Elementary Introduction to Elliptic Curves II*, Tech. Report CCR Expository Report No. 34, Institute for Defense Analysis, 1990.

[CC03] Jae Choon Cha and Jung Hee Cheon, *An Identity-Based Signature from Gap Diffie-Hellman Groups*, Public Key Cryptography - PKC 2003, LNCS 2139, Springer-Verlag, 2003, pp. 18–30.

[CC05] Liqun Chen and Zhaohui Cheng, *Security Proof of Sakai-Kasahara's Identity-Based Encryption Scheme*, Cryptology ePrint Archive, Report 2005/226, 2005.

[CCD+05] Dario Catalano, Ronald Cramer, Ivan Damgard, Giovanni Di Crescenzio, David Pointcheval, and Tsuyoshi Takagi, *Contemporary Cryptology*, Advanced Courses in Mathematics-CRM Barcelona, Birkhauser-Verlag, Germany, 2005.

[CCS07] L. Chen, Z. Cheng, and N.P. Smart, *Identity-based key agreement protocols from pairings*, International Journal of Information Security **6** (2007), no. 4, 213–241.

[CHL04] Kyu Young Choi, Jung Yeon Hwang, and Dong Hoon Lee, *Efficient ID-based Group Key Agreement with Bilinear Maps*, Public Key Cryptography - Proceedings of PKC 2004, LNCS 2947, Springer-Verlag, 2004, pp. 130–144.

[CHL08] Kyu Young Choi, Jung Yeon Hwang, and Dong Hoon Lee, *ID-Based Authenticated Group Key Agreement Secure Against Insider Attacks*, IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences **91** (2008), no. 7, 1828–1830.

[Cho06] Kim-Kwang Raymond Choo, *Key Establishment : Proofs and Refutations*, PhD, Queensland University of Technology, Brisbane, Australia, 2006.

[CHP07] Jan Camenisch, Susan Hohenberger, and Michael Østergaard Pedersen Pedersen, *Batch Verification of Short Signatures*, Advances in Cryptology - Proceedings of EUROCRYPT '07, LNCS 4515 (Berlin, Heidelberg), Springer-Verlag, 2007, pp. 246–263.

[CK01] Ran Canetti and Hugo Krawczyk, *Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels*, Advances in Cryptology - Proceedings of EUROCRYPT 2001, LNCS 2045 (London, UK), Springer-Verlag, 2001, pp. 453–474.

[CK02] Liqun Chen and Caroline Kudla, *Identity Based Authenticated Key Agreement Protocols from Pairings*, Cryptology ePrint Archive, Report 2002/184, 2002.

[CKY04] Jung Hee Cheon, Yongdae Kim, and Hyo Jin Yoon, *A New ID-based Signature with Batch Verification*, Cryptology ePrint Archive, Report 2004/131, 2004.

[CLLC06] Huiyan Chen, Shuwang Lu, Zhenhua Liu, and Qing Chen, *An Identity-Based Signcryption Scheme with Short Ciphertext from Pairings*, Emerging Directions in Embedded and Ubiquitous Computing, LNCS **4097** (2006), 342–351.

[CLX06] Tianjie Cao, Dongdai Lin, and Rui Xue, *Security Analysis of Some Batch Verifying Signatures from Pairings*, International Journal of Network Security **3** (2006), no. 2, 138–143.

[CML05] Liqun Chen and John Malone-Lee, *Improved Identity-Based Signcryption*, Public Key Cryptography - PKC 2005, LNCS 3386, Springer-Verlag, 2005, pp. 362–279.

[CS05] Y. Challal and H. Seba, *Group Key Management Protocols: A Novel Taxonomy*, International Journal of Information Technology **2** (2005), no. 1, 105–118.

[CSCW07] Seokhyang Cho, Kiwon Song, Dongsub Cho, and Dongho Won, *Secure Mobile Content Delivery Using Dynamic Group Key Agreement with Batch Verification*, Computational Science and Its Applications - Proceedings of ICCSA 2007, LNCS 4706, Springer-Verlag, 2007, pp. 996–1007.

[CYHC04] Sherman S. M. Chow, Siu-Ming Yiu, Lucas Chi Kwong Hui, and K. P. Chow, *Efficient Forward and Provably Secure ID-Based Signcryption Scheme with Public Verifiability and Public Ciphertext Authenticity*, Information Security and Cryptology - Proceedings of ICISC 2003, LNCS 2971, Springer-Verlag, 2004, pp. 352–369.

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor, *Nonmalleable Cryptography*, SIAM Journal on Computing **30** (2000), no. 2, 391–437.

[Déc05] Isabelle Déchène, *Generalized Jacobians in Cryptography*, PhD in Mathematics, McGill University, Montreal, Canada, 2005.

[DFJW04] Yevgeniy Dodis, Michael J. Freedman, Stanislaw Jarecki, and Shabsi Walfish, *Optimal Signcryption from Any Trapdoor Permutation*, Cryptology ePrint Archive, Report 2004/020, 2004.

[DH76] Whitfield Diffie and Martin E. Hellman, *New Directions in Cryptography*, IEEE Transactions on information Theory **22** (1976), no. 6, 644–654.

[DWGW03a] Xinjun Du, Ying Wang, Jianhua Ge, and Yumin Wang, *ID-based Authenticated Two Round Multi-Party Key Agreement*, Cryptology ePrint Archive, Report 2003/247, 2003.

[DWGW03b] ———, *An Improved ID-based Authenticated Group Key Agreement Scheme*, Cryptology ePrint Archive, Report 2003/260, 2003.

[FGHP08] Anna Lisa Ferrara, Matthew Green, Susan Hohenberger, and Michael Østergaard Pedersen, *Practical Short Signature Batch Verification*, Cryptology ePrint Archive, Report 2008/015, 2008.

[Fia89] Amos Fiat, *Batch RSA*, Advances in Cryptology - Proceedings of CRYPTO '89, LNCS 435 (London, UK), Springer-Verlag, 1989, pp. 175–185.

[FIP93] Federal Information Processing Standards FIPS, *(DSS) - Digital Signature Standard*, February 1993.

[FMR99] Gerhard Frey, Michael Müller, and Hans-Georg Rück, *The Tate Pairing and the Discrete Logarithm Applied to Elliptic Curve Cryptosystems*, IEEE Transactions on Information Theory **45** (1999), no. 5, 1717–1719.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto, *Secure Integration of Asymmetric and Symmetric Encryption Schemes*, Advances in Cryptology - Proceedings of CRYPTO '99, LNCS 1666 (London, UK), Springer-Verlag, 1999, pp. 537–554.

[FR94] Gerhard Frey and Hans-Georg Rück, *A Remark Concerning m-Divisibility and the Discrete Logarithm in the Divisor Class Group of Curves*, Mathematics of Computation **62** (1994), no. 206, 865–874.

[GHS02] Steven D. Galbraith, Keith Harrison, and David J. Soldera, *Implementing the Tate pairing*, Algorithmic Number Theory - Proceedings of the 5th International Symposium, ANTS-V, LNCS 2369, Springer, 2002, pp. 324–337.

[GM84] Shafi Goldwasser and Silvio Micali, *Probabilistic Encryption*, Journal of Computer and System Sciences **28** (1984), no. 2, 270–299.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest, *A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks*, SIAM Journal on Computing **17** (1988), 281–308.

[Hes02] Florian Hess, *Exponent Group Signature Schemes and Efficient Identity Based Signature Schemes Based on Pairings*, Cryptology ePrint Archive, Report 2002/012, 2002.

[Hes03] ———, *Efficient Identity Based Signature Schemes Based on Pairings*, SAC '02: Revised Papers from the 9th Annual International Workshop on Selected Areas in Cryptography, LNCS 2595 (London, UK), Springer-Verlag, 2003, pp. 310–324.

[Hes04] ———, *A Note on the Tate Pairing of Curves over Finite Fields*, Arch. Math **82** (2004), 28–32.

[HLH07] Yong-Zhong He, Xiao-Yong Li, and Zhen Han, *An Insider Attack Resistant Group Key Agreement Protocol from Pairings*, ICICIC '07: Proceedings of the Second International Conference on Innovative Computing, Information and Control, IEEE Computer Society, 2007, pp. 281–284.

[HLL07] Chengyu Hu, Pengtao Liu, and Daxing Li, *An Efficient Group Key Agreement Protocol from Bilinear Pairings*, CISW '07: Proceedings of the 2007 International Conference on Computational Intelligence and Security Workshops (Washington, DC, USA), IEEE Computer Society, 2007, pp. 737–740.

[How96] Everett W. Howe, *The Weil pairing and the Hilbert symbol*, Mathematische Annalen **305** (1996), no. 1, 387–392.

[HPS08] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman, *An Introduction to Mathematical Cryptography*, Springer Publishing Company, Incorporated, 2008.

[ITW82] Ingemar Ingemarsson, Donald T. Tang, and C. K. Wong, *A conference key distribution system*, IEEE Transactions on Information Theory **28** (1982), no. 5, 714–720.

[JN03] Antoine Joux and Kim Nguyen, *Separating Decision Diffie-Hellman from Computational Diffie-Hellman in Cryptographic Groups*, Journal of Cryptology **16** (2003), no. 4, 239–247.

[Jou00] Antoine Joux, *A One Round Protocol for Tripartite Diffie-Hellman*, ANTS-IV: Proceedings of the 4th International Symposium on Algorithmic Number Theory, LNCS 1838 (London, UK), Springer-Verlag, 2000, pp. 385–394.

[Jou04] ———, *A One Round Protocol for Tripartite Diffie-Hellman*, Journal of Cryptology **17** (2004), no. 4, 263–276.

[Kal87] Burton S. Kaliski, Jr., *A pseudo-random bit generator based on elliptic logarithms*, Advances in Cryptology - Proceedings of CRYPTO '86, LNCS 263 (London, UK), Springer-Verlag, 1987, pp. 84–103.

[Kal01] ———, *An Unknown Key-Share Attack on the MQV Key Agreement Protocol*, ACM Transactions on Information and System Security **4** (2001), no. 3, 275–288.

[KKC+06] Jeeyeon Kim, Seungjoo Kim, Kilsoo Chun, Jaeil Lee, and Dongho Won, *Group Key Agreement Protocol Among Mobile Devices in Different Cells*, ISPA 2006, LNCS 4431, Springer-Verlag, 2006, pp. 1090–1097.

[KLL04] Hyun-Jeong Kim, Su-Mi Lee, and Dong Hoon Lee, *Constant-Round Authenticated Group Key Exchange for Dynamic Groups*, Advances in Cryptology - Proceedings of ASIACRYPT 2004, LNCS 3329, Springer, 2004, pp. 245–259.

[KNKW05] Hyunjue Kim, Junghyun Nam, Seungjoo Kim, and Dongho Won, *Secure and Efficient ID-Based Group Key Agreement Fitted for Pay-TV*, Advances in Multimedia Information Processing - Proceedings of PCM 2005, LNCS 3768, Springer-Verlag, 2005, pp. 117–128.

[KPT00] Yongdae Kim, Adrian Perrig, and Gene Tsudik, *Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups*, CCS '00: Proceedings of the 7th ACM conference on Computer and Communications Security (New York, NY, USA), ACM, 2000, pp. 235–244.

[KPT01] Yongdae Kim, Adrian Perrig, and Gene Tsudik, *Communication-Efficient Group Key Agreement*, SEC '01: Proceedings of the 16th International Conference on Information Security: Trusted Information, 2001, pp. 229–244.

[KPT04] ———, *Tree-Based Group Key Agreement*, ACM Transactions on Information and System Security **7** (2004), no. 1, 60–96.

[KY03] Jonathon Katz and Moti Yung, *Scalable Protocols for Authenticated Group Key Exchange*, Advances in Cryptology - Proceedings of CRYPTO 2003, LNCS 2729, Springer-Verlag, 2003, pp. 110–125.

[LKKR04] Sangwon Lee, Yongdae Kim, Kwangjo Kim, and Dae-Hyun Ryu, *An Efficient Tree-Based Group Key Agreement Using Bilinear Map*, ACNS 2003, LNCS 2846, Springer-Verlag, 2004, pp. 357–371.

[LL94] Chae Hoon Lim and Pil Joong Lee, *On the Security of Interactive DSA Batch Verification*, Electronics Letters **30** (1994), no. 19, 1592–1593.

[LM07] Laurie Law and Brian J. Matt, *Finding Invalid Signatures in Pairing-Based Batches*, Cryptography and Coding, LNCS 4887, Springer-Verlag, 2007, pp. 34–53.

[LMQ+98] Laurie Law, Alfred Menezes, Minghua Qu, Jerry Solinas, and Scott Vanstone, *An efficient protocol for authenticated key agreement*, Tech. Report CORR 98-05, University of Waterloo, 1998.

[LMQ+03] ———, *An Efficient Protocol for Authenticated Key Agreement*, Designs, Codes and Cryptography **28** (2003), no. 2, 119–134.

[LQ03] Benoît Libert and Jean-Jacques Quisquater, *New Identity Based Signcryption Schemes from Pairings*, Cryptology ePrint Archive, Report 2003/023, 2003.

[LQ04a] ———, *The Exact Security of an Identity Based Signature and its Applications*, Cryptology ePrint Archive, Report 2004/102, 2004.

[LQ04b] ———, *Improved Signcryption from q-Diffie-Hellman Problems*, Security in Network Communications, LNCS 3352, Springer-Verlag, 2004, pp. 220–234.

[Mat09] Brian J. Matt, *Identification of Multiple Invalid Signatures in Pairing-Based Batched Signatures*, Public Key Cryptography - Proceedings of PKC 2009, LNCS 5443, Springer-Verlag, 2009, pp. 337–356.

[ML02] John Malone-Lee, *Identity-Based Signcryption*, Cryptology ePrint Archive, Report 2002/098, 2002.

[ML05] ———, *Signcryption with Non-interactive Non-repudiation*, Designs, Codes and Cryptography **37** (2005), no. 1, 81–109.

[MOV93] Alfred Menezes, Tatsuaki Okamoto, and Scott A. Vanstone, *Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field*, IEEE Transactions on Information Theory **39** (1993), no. 5, 1639–1646.

[NIS07] National Institute of Standards and Technology NIST, *Cryptographic Hash Algorithm Competition*, http://csrc.nist.gov/, 2007.

[NLKW05] Junghyun Nam, Jinwoo Lee, Seungjoo Kim, and Dongho Won, *DDH-based Group Key Agreement for Mobile Computing*, Journal of Systems and Software **17** (2005), no. 1, 73–83.

[NMVR94] David Naccache, David M'Raïhi, Serge Vaudenay, and Dan Raphaeli, *Can D.S.A. be Improved? Complexity Trade-Offs with the Digital Signature Standard*, Advances in Cryptology - Proceedings of EUROCRYPT '94, LNCS 950, 1994, pp. 77–85.

[NR03] Divya Nalla and K. C. Reddy, *Signcryption Scheme for Identity-based Cryptosystems*, Cryptology ePrint Archive, Report 2003/066, 2003.

[PAK09] Hyewon Park, Tomoyuki Asano, and Kwangjo Kim, *Improved ID-based Authenticated Group Key Agreement Secure Against Impersonation Attack by Insider*, 2009 Symposium on Cryptography and Information Security (SCIS 2009), preprint, 2009.

[Pat02] Kenneth G. Paterson, *ID-based Signatures from Pairings on Elliptic Curves*, Electronics Letters **38** (2002), no. 18, 1025–1026.

[PHYK08] Hyewon Park, Kyusuk Han, Cna Yeob Yeun, and Kwangjo Kim, *Improving Choi et al.'s ID-based Authenticated Group Key Agreement Scheme at PKC2004*, 2008 Symposium on Cryptography and Information Security (SCIS 2008), preprint, 2008.

[Pom97] Carl Pomerance (ed.), *Cryptology and Computational Number Theory*, American Mathematical Society, Boston, MA, USA, 1997.

[RH03] S. Rafaeli and D. Hutchison, *A Survey of Key Management for Secure Group Communication*, ACM Computing Surveys **35** (2003), no. 3, 309–329.

[RN02] K.C. Reddy and Divya Nalla, *Identity Based Authenticated Group Key Agreement Protocol*, INDOCRYPT 2002, LNCS 2551 (2002), 215–233.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM **21** (1978), no. 2, 120–126.

[Sha79] Adi Shamir, *How to Share a Secret*, Communications of the ACM **22** (1979), no. 11, 612–613.

[Sha85] ———, *Identity-based cryptosystems and signature schemes*, Advances in Cryptology - Proceedings of CRYPTO '84, LNCS 196, Springer-Verlag, 1985, pp. 47–53.

[Shi03] Kyung-Ah Shim, *Cryptanalysis of Al-Riyami-Paterson's Authenticated Three Party Key Agreement Protocols*, Cryptology ePrint Archive, Report 2003/122, 2003.

[Shi07] ———, *Further Analysis of ID-Based Authenticated Group Key Agreement Protocol from Bilinear Maps*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science **E90-A** (2007), no. 1, 295–298.

[SHL05] Sheng-Hua Shiau, Ren-Junn Hwang, and Ming-Fuu Lin, *Key Agreement Protocol Based on Weil Pairing*, AINA '05: Proceedings of the 19th International Conference on Advanced Information Networking and Applications (Washington, DC, USA), IEEE Computer Society, 2005, pp. 597–602.

[Sho99] Victor Shoup, *On Formal Models for Secure Key Exchange*, Tech. Report RZ3120, IBM Zurich Research Lab, November 1999.

[Sil86] Joseph H. Silverman, *The Arithmetic of Elliptic Curves*, Graduate Texts in Mathematics, vol. 106, Springer-Verlag, New York, 1986.

[SLS03] Jun-Bum Shin, Kwangsu Lee, and Kyungah Shim, *New DSA-Verifiable Signcryption Schemes*, Information, Security and Cryptology - Proceedings of ICISC 2002, LNCS 2587, Springer-Verlag, 2003, pp. 35–47.

[Sma01] Nigel Smart, *An Identity based Authenticated Key Agreement protocol Based on the Weil Pairing*, Cryptology ePrint Archive, Report 2001/111, 2001.

[Smi05] Clayton D. Smith, *Digital Signcryption*, MSc in Mathematics, University of Waterloo, Waterloo, Ontario, Canada, 2005.

[SOK00] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara, *Cryptosystems based on pairing*, SCIS 2000 (Okinawa, Japan), January 2000.

[SSDW90] D. G. Steer, L. Strawczynski, Whitfield Diffie, and Michael J. Wiener, *A Secure Audio Teleconference System*, Advances in Cryptology - CRYPTO '88, LNCS 403 (London, UK), Springer-Verlag, 1990, pp. 520–528.

[STW96] Michael Steiner, Gene Tsudik, and Michael Waidner, *Diffie-Hellman Key Distribution Extended to Group Communication*, CCS '96: Proceedings of the 3rd ACM Conference on Computer and Communications Security (New York, NY, USA), ACM, 1996, pp. 31–37.

[SVG+08a] S. Sharmila Deva Selvi, S. Sree Vivek, Ragavendran Gopalakrishnan, Naga Naresh Karuturi, and Pandu Rangan Chandrasekaran, *Cryptanalysis of ID-Based Signcryption Scheme for Multiple Receivers*, Cryptology ePrint Archive, Report 2008/238, 2008.

[SVG+08b] S.Sharmila Deva Selvi, S.Sree Vivek, Ragavendran Gopalakrishnan, Naga Naresh Karuturi, and Pandu Rangan Chandrasekaran, *On the Provable Security of Multi-Receiver Signcryption Schemes*, Cryptology ePrint Archive, Report 2008/238, 2008.

[TM04] Qiang Tang and Chris J. Mitchell, *Rethinking the Security of some Authenticated Group Key Agreement Schemes*, Cryptology ePrint Archive, Report 2004/363, 2004.

[TM05] ———, *Security Properties of Two Authenticated Conference Key Agreement Protocols*, ICICS 2005 - Proceedings of the 7th International Conference on Information and Communications Security, LNCS 3783, Springer, 2005, pp. 304–314.

[Tse06] Yuh-Min Tseng, *On the Security of Two Group Key Agreement Protocols for Mobile Devices*, MDM '06: Proceedings of the 7th International Conference on Mobile Data Management (Washington, DC, USA), IEEE Computer Society, 2006, pp. 97–100.

[Tse07] ———, *A secure authenticated group key agreement protocol for resource-limited mobile devices*, The Computer Journal **50** (2007), no. 1, 41–52.

[Ver01] Eric R. Verheul, *Evidence that XTR is More Secure than Supersingular Elliptic Curve Cryptosystems*, Advances in Cryptology - Proceedings of EUROCRYPT 2001, LNCS 2045, Springer-Verlag, 2001, pp. 195–210.

[Ver04] Eric R. Verheul, *Evidence that XTR is More Secure than Supersingular Elliptic Curve Cryptosystems*, Journal of Cryptology **17** (2004), no. 4, 277–296.

[Was08] Lawrence C. Washington, *Elliptic Curves: Number Theory and Cryptography, Second Edition*, Discrete Mathematics and Its Applications, Chapman & Hall/CRC, 2008.

[Wat05] Brent Waters, *Efficient Identity-Based Encryption Without Random Oracles*, Advances in Cryptology - Proceedings of EUROCRYPT '05, LNCS 3494, Springer, 2005, pp. 114–127.

[YCK04] HyoJin Yoon, Jung Hee Cheon, and Yongdae Kim, *Batch Verifications with ID-Based Signatures*, ICISC 2004, LNCS 3506, Springer-Verlag, 2004, pp. 233–248.

[YHVK08] Chan Yeob Yeun, Kyusuk Han, Duc Liem Vo, and Kwangjo Kim, *Secure Authenticated Group Key Agreement Protocol in the MANET Environment*, Information Security Technical Report **13** (2008), no. 3, 158–164.

[YKH+06] Peng Yang, Takashi Kitagawa, Goichiro Hanaoka, Rui Zhang, Kanta Matsuura, and Hideki Imai, *Applying Fujisaki-Okamoto to Identity-Based Encryption*, Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 16th International Symposium (AAECC16), LNCS 3857, Springer, 2006, pp. 183–192.

[YYHZ07] Yong Yu, Bo Yang, Xinyi Huang, and Mingwu Zhang, *Efficient Identity-Based Signcryption Scheme for Multiple Receivers*, ATC 2007, LNCS 4610, Springer-Verlag, 2007, pp. 13–21.

[ZC03] Fangguo Zheng and Xiaofeng Chen, *Attacks on Two ID-based Authenticated Group Key Agreement Schemes*, Cryptology ePrint Archive, Report 2003/259, 2003.

[Zhe97] Yuliang Zheng, *Digital Signcryption or How to Achieve Cost(Signature & Encryption) << Cost(Signature) + Cost(Encryption)*, Advances in Cryptology - Proceedings of CRYPTO '97, LNCS 1294 (London, UK), Springer-Verlag, 1997, pp. 165–179.

[ZI98] Yuliang Zheng and Hideki Imai, *Compact and Unforgeable Key Establishment over an ATM Network*, In Proceedings of IEEE INFOCOM '98, vol. 2, IEEE, Inc., 1998, pp. 411–418.

[ZK03] Fangguo Zhang and Kwangjo Kim, *Efficient ID-Based Blind Signature and Proxy Signature from Bilinear Pairings*, Proceedings of ACISP'03, LNCS 2727, Springer-Verlag, 2003, pp. 312–323.

[ZLK02] Fangguo Zhang, Shengli Liu, and Kwangjo Kim, *ID-Based One Round Authenticated Tripartite Key Agreement Protocol with Pairings*, Cryptology ePrint Archive, Report 2002/122, 2002.

[ZSM06] Lan Zhou, Willy Susilo, and Yi Mu, *Efficient ID-Based Authenticated Group Key Agreement from Bilinear Pairings*, Mobile Ad-hoc and Sensor Networks, LNCS 4325, Springer-Verlag, 2006, pp. 521–532.

[ZSNS04] Fangguo Zhang, Reihaneh Safavi-Naini, and Willy Susilo, *An Efficient Signature Scheme from Bilinear Pairings and Its Applications*, Public Key Cryptography - Proceedings of PKC 2004, LNCS 2947, Springer, 2004, pp. 277–290.