

UNIVERSITY OF CALIFORNIA  
Los Angeles

**New Constructions in Pairing-Based Cryptography**

A dissertation submitted in partial satisfaction  
of the requirements for the degree  
Doctor of Philosophy in Mathematics

by

**Steve Naichia Lu**

2009

UMI Number: 3351737

Copyright 2009 by  
Lu, Steve Naichia

All rights reserved.

### INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

**UMI**<sup>®</sup>

---

UMI Microform 3351737

Copyright 2009 by ProQuest LLC.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

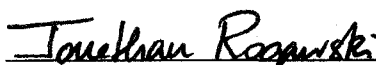
ProQuest LLC  
789 E. Eisenhower Parkway  
PO Box 1346  
Ann Arbor, MI 48106-1346

© Copyright by  
Steve Naichia Lu  
2009

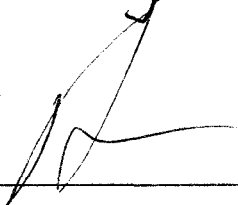
The dissertation of Steve Naichia Lu is approved.



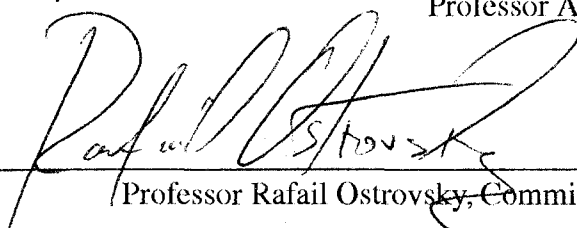
Professor Haruzo Hida



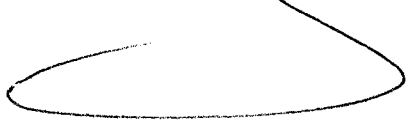
Professor Jonathan Rogawski



Professor Amit Sahai



Professor Rafail Ostrovsky, Committee Chair



University of California, Los Angeles

2009

To my family.

## TABLE OF CONTENTS

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Sequential Aggregate Signatures and Variants	3
1.1.1	Related Work	3
1.1.2	Our Contribution	4
1.2	Non-Interactive Shuffle with Pairing-Based Verifiability	6
1.2.1	Related Work	7
1.2.2	Our Contribution	8
1.3	Accountable Authority Identity-Based Encryption	9
1.3.1	Related Work	11
1.3.2	Our Contribution	12
<b>2</b>	<b>Preliminaries and Notation</b>	<b>16</b>
2.1	Groups with Efficiently Computable Bilinear Maps	16
2.2	Asymptotic and Concrete Security	17
2.3	Computational Assumptions	18
2.3.1	Computational Diffie-Hellman	18
2.3.2	Decisional Linear Assumption	19
2.3.3	Decisional Bilinear Diffie-Hellman (DBDH) Assumption	19
<b>3</b>	<b>Sequential Aggregate Signatures and Variants</b>	<b>21</b>
3.1	Background	21
3.1.1	Existentially Unforgeable Signatures	22

3.1.2	The Waters Signature Scheme . . . . .	23
3.2	Sequential Aggregate Signatures . . . . .	24
3.2.1	Definitions . . . . .	26
3.2.2	The WSA Sequential Aggregate Signature Scheme . . . . .	28
3.2.3	Proof of Security . . . . .	31
3.2.4	A More Efficient Variant in the Random Oracle Model . . . . .	34
3.3	Multisignatures . . . . .	34
3.3.1	The WM Multisignature Scheme . . . . .	35
3.3.2	Proof of Security . . . . .	36
3.4	Verifiably Encrypted Signatures . . . . .	38
3.4.1	Our Scheme . . . . .	39
3.4.2	VES from General Assumptions . . . . .	41
3.5	WVES Proofs of Security . . . . .	45
3.5.1	Unforgeability . . . . .	45
3.5.2	Opacity . . . . .	47
3.6	Comparison to Previous Work . . . . .	50
3.7	Conclusions and Open Problems . . . . .	52
<b>4</b>	<b>Non-Interactive Shuffle with Pairing-Based Verifiability . . . . .</b>	<b>54</b>
4.1	Background . . . . .	54
4.1.1	BBS Encryption . . . . .	54
4.1.2	Shuffling BBS Ciphertexts . . . . .	55
4.1.3	Non-interactive Zero-Knowledge Arguments . . . . .	56

4.1.4	Non-interactive WI Proofs for Bilinear Groups . . . . .	59
4.2	Cryptographic Assumptions . . . . .	61
4.2.1	Permutation Pairing Assumption . . . . .	61
4.2.2	Simultaneous Pairing Assumption . . . . .	62
4.2.3	Our Assumptions in the Generic Group Model . . . . .	63
4.3	NIZK Argument for Correctness of a Shuffle . . . . .	67
4.4	Remark on Shuffling BGN Ciphertexts . . . . .	74
4.5	Conclusions and Open Problems . . . . .	75
<b>5</b>	<b>Accountable Authority Identity-Based Encryption . . . . .</b>	<b>76</b>
5.1	Background . . . . .	76
5.1.1	Identity-Based Encryption . . . . .	77
5.1.2	Fully Simulatable k-out-of-n Oblivious Transfer . . . . .	78
5.1.3	Attribute-Based Encryption . . . . .	78
5.2	Definitions and the Model . . . . .	80
5.3	Generic construction of A-IBE . . . . .	85
5.3.1	Discussion on Requirements for Building Blocks . . . . .	86
5.3.2	The Construction . . . . .	89
5.4	Security Proofs . . . . .	94
5.4.1	Dishonest PKG Game . . . . .	95
5.4.2	Dishonest User Game . . . . .	107
5.5	Concrete Construction Based on the DBDH Assumption . . . . .	116
5.5.1	The Construction . . . . .	117



5.5.2 Security Proofs . . . . .	121
5.6 Conclusion and Open Problems . . . . .	123
<b>References . . . . .</b>	<b>125</b>

## LIST OF TABLES

3.1	Comparison of aggregate signature schemes. . . . .	50
3.2	Comparison of multisignature schemes. . . . .	52
3.3	Comparison of verifiably encrypted signature schemes. . . . .	52

## ACKNOWLEDGMENTS

First, I thank my advisor Rafail Ostrovsky for introducing me to the fascinating field of cryptography. His spirited enthusiasm for the subject has inspired and motivated my own passion for research. He seems to have boundless energy, and even while keeping a busy schedule, he was always able to dedicate time and energy to advise me in my studies, and I am deeply grateful for that. His dutiful mentoring has played a major role in shaping me into the researcher I am today.

In the time that I spent with Rafi, I have found that he has the admirable ability to apply his wealth of knowledge to contribute great ideas and support for the various topics which I have found interesting. I feel quite fortunate in that regard, and it was because of his guidance that I was able to discover and research the problems which I enjoyed the most. He has also been a fantastic mentor and collaborator, and I hope for many more opportunities to work with him in the future.

Early in my journey into cryptography, I also had the privilege of learning from and working with Amit Sahai. I thank him for his guidance, collaboration, and support over the past few years. He has on numerous occasions taken his time to offer me his well-informed perspective on many different subjects. I noticed that every time I had a long chat with Amit for his supervision and comments, I was able to gain a valuable piece of wisdom and understanding. His sensible and honest attitude in his advice has helped me to reflect upon my own craft and the type of research I aim to do in the future. The advice that I have received from Amit has also had a great impact on the kind of researcher I am today. I am immensely thankful for his genuine concern and sound advice toward my research, career, and life in general. I also look forward to future opportunities to have the pleasure of collaborating with him.

Before delving into cryptography, I studied number theory with other students in the number theory group. I wish to acknowledge Don Blasius for his commitment to the number theory students and the department. My most vivid memories of my experience with him are in his participation in our seminars into the late hours of the evening. He has, among many other things, helped organize inter-disciplinary seminars and courses, which has been of great assistance to my transition into cryptography. But for me what stands out most is that even among these wonderful orchestrations that he has done in general, he also had a personal commitment for my success as an individual student. I am indebted to him for serving as my committee chair when I was advancing to candidacy. I thank him for his support in the journey of my life as a graduate student.

I thank the members of my Ph.D. committee — Haruzo Hida, Jonathan Rogawski, Amit Sahai, and Rafail Ostrovsky (Chair) — for overseeing my passage from my advancement to candidacy to the submission of my thesis.

I thank the Institute for Pure and Applied Mathematics at UCLA for inviting me to their three-month program “Securing Cyberspace: Applications and Foundations of Cryptography and Computer Security” and for providing financial support.

I thank my collaborators Vipul Goyal, Jens Groth, Daniel Manchala, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters.

I thank all my friends for their companionship and the entertaining times we shared.

I thank Michel Abdalla for allowing me to use his extensive bibliography.

The results in Chapter 3 partially contain joint work with Ostrovsky, Sahai, Shacham, and Waters [LOS06], the extended abstract of which appeared in the proceedings of EUROCRYPT 2006. My collaborators and I thank Dan Boneh, Andrew Bortz, Xavier Boyen, and Ilya Mironov for their helpful comments.

The results in Chapter 4 partially contain joint work with Jens Groth [GL07a], the extended abstract of which appeared in the proceedings of ASIACRYPT 2007.

The results in Chapter 5 partially contain joint work with Goyal, Sahai, and Waters [GLS08], the extended abstract of which appeared in the proceedings of the ACM Conference on Computer and Communications Security 2008.

## VITA

- 1985            Born, Torrance, California, USA.
- 1996–2000     B.S. Mathematics and B.S. Computer Science, CSU Dominguez Hills.
- 2001–2002     M.S. Computer Science, Stanford University.
- 2002–present   Ph.D. student, Department of Mathematics, UCLA.
- 2005–2008     Teaching Assistant, Department of Mathematics, UCLA. For 7 academic quarters during this time period.
- Summer 2005   Research Assistant, Department of Mathematics, UCLA.
- Summer 2006   Research Assistant, Department of Mathematics, UCLA.
- Fall 2006       Research Fellow. Institute for Pure and Applied Mathematics, UCLA.
- Summer 2007   Research Assistant, Department of Mathematics, UCLA.

## PUBLICATIONS

Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. “Sequential aggregate signatures and multisignatures without random oracles.” In *EUROCRYPT 2006*, pages 465–485. Springer, July 2006.

Jens Groth and Steve Lu. “Verifiable shuffle of large size ciphertexts.” In *PKC 2007*, pages 377–392. Springer, June 2007.

Jens Groth and Steve Lu. “A non-interactive shuffle with pairing based verifiability.” In *ASIACRYPT 2007*, pages 51–67. Springer, November 2007.

Steve Lu, Daniel Manchala, and Rafail Ostrovsky. “Visual cryptography on graphs.” In *COCOON 2008*, pages 225–234. Springer, June 2008. Best Paper Award.

Vipul Goyal, Steve Lu, Amit Sahai, and Brent Waters. “Black-box accountable authority identity-based encryption.” In *CCS '08*, pages 427–436. ACM Press, 2008.

ABSTRACT OF THE DISSERTATION

**New Constructions in Pairing-Based Cryptography**

by

**Steve Naichia Lu**

Doctor of Philosophy in Mathematics

University of California, Los Angeles, 2009

Professor Professor Rafail Ostrovsky, Chair

In the past decade, pairing-based cryptography has emerged as an active area of research that gave rise to new algorithms, protocols, and primitives. These new techniques allowed researchers to achieve cryptographic schemes which had no known (or less efficient) counterparts in groups without bilinear pairings. In this dissertation, we introduce several schemes in which pairings play a central role in their construction. The results that we present in this dissertation stem from three papers which are respectively joint work with Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters, with Jens Groth, and with Vipul Goyal, Amit Sahai, and Brent Waters.

In the dissertation, we present the first sequential aggregate signature, the first multisignature, and the first verifiably encrypted signature provably secure without random oracles. Our constructions derive from a novel application of a signature scheme due to Waters. We review the definition of these signature variants and consider applications to secure routing and proxy signatures. We show how these are constructed using pairing-based cryptography.

Another scheme we present is a non-interactive verifiable shuffle. A shuffle is a permutation and re-encryption of a set of ciphertexts. Shuffles are used, for instance, in mix-nets for anonymous broadcast and voting. One way to make a shuffle verifiable



is to give a zero-knowledge proof of correctness. All currently known practical zero-knowledge proofs for correctness of a shuffle rely on interaction. We give the first efficient non-interactive zero-knowledge proof for correctness of a shuffle based on pairings.

Finally, we consider the problem of accountability for PKGs in identity-based encryption. A well-known concern in the setting of identity-based encryption is that the PKG is all powerful and must be completely trusted. To mitigate this problem, the notion of Accountable Authority Identity-Based Encryption (A-IBE) was recently introduced by Goyal, who provided constructions to realize the notion of A-IBE only in the white-box and weak black-box models. In this dissertation, we present a resolution to the main open question left in Goyal's work by providing a construction of a fully black-box A-IBE system. We show how such a scheme can be securely realized from generic underlying primitives, then give a concrete realization of the scheme in any bilinear group where the Decisional Bilinear Diffie-Hellman assumption holds.

# CHAPTER 1

## Introduction

In this dissertation, we investigate the construction of various cryptographic schemes by making use of groups with bilinear pairings.

The roots of pairing-based cryptography lie in the use of bilinear pairings on elliptic curves. Research on elliptic curve cryptography began when, in 1985, Koblitz [Kob87] and Miller [Mil86b] independently suggested the use of elliptic curves for cryptography (elliptic curves have also found important uses in cryptanalysis, e.g. Lenstra's elliptic curve factorization [Len87]). While the initial outlook was promising, in 1991 Menezes, Okamoto, and Vanstone [MVO91] cast some doubt on the subject by showing that due to the structure of certain supersingular curves, the underlying assumption — the elliptic curve discrete log problem (ECDLP) — was less secure. Their method was a novel use of the Weil pairing (introduced by Weil [Wei40] in his proof of the Riemann hypothesis for function fields) which reduced the hardness of ECDLP to the hardness of the discrete logarithm problem in a finite field. This was one of the first results that required the efficient computation of a bilinear pairing. The actual computation of the Weil pairing relies on Miller's algorithm [Mil86a], discovered a few years earlier, in 1986. In 1994, Frey and Rück [FR94] produced a similar attack by using the Tate pairing on elliptic curves.

The turn of the century brought several cryptographic results using bilinear pairings, including the works of Mitsunari-Sakai-Kasahara [MSK02], Sakai-Oghishi-Kasahara [SOK00], Joux [Jou04], and Boneh-Franklin [BF01, BF03].

Boneh and Franklin’s result was one of the two first efficient solutions to the problem of creating an identity-based encryption scheme, first posed by Shamir in 1984 [Sha85] (the other solution, by Cocks [Coc01], worked in groups that relied on the hardness of the Quadratic Residuosity assumption). Bilinear pairings have since proven to be very useful in the construction of cryptographic schemes. Indeed, in the past decade, there have been many new results in pairing-based cryptography (see, for a partial list, Barreto’s website [Bar06]).

Research in optimizing the computation and construction of these bilinear pairings has also emerged. These improvements are important for the efficient implementation of pairing-based schemes. Clever optimizations of the Tate pairing has lead to the introduction of new pairings such as the eta pairing [BGh07] and the ate pairing [HSV06], whose computation are still based upon Miller’s algorithm. In addition, several works have proposed classes of elliptic curves which are “pairing-friendly”, namely those that satisfy certain structural properties (see, e.g., the taxonomy of Freeman, Scott, and Teske [FST06]). For this dissertation, we circumnavigate the details of these constructions and work instead with abstract groups which admit bilinear pairings (we define these in Chapter 2).

This dissertation presents three results in pairing-based cryptography. The extended abstracts of these results appear in [LOS06, GL07a, GLS08]. We organize the chapters as follows. First, in Chapter 2, we review preliminary information, give the necessary background, and introduce the notation to be used throughout the dissertation. In Chapter 3<sup>1</sup> we present a pairing-based sequential aggregate signature scheme, a multisignature scheme, and a verifiably encrypted signature scheme, which are provably secure without random oracles. Then, in Chapter 4<sup>2</sup>, we present a non-interactive

---

<sup>1</sup>Partially contains work by Lu, Ostrovsky, Sahai, Shacham, Waters [LOS06]

<sup>2</sup>Partially contains work by Groth, Lu [GL07a]

shuffle with pairing-based verifiability. Finally, Chapter 5<sup>3</sup> introduces the notion of a black-box accountable authority identity-based encryption scheme and presents the construction of such a scheme.

We now give a brief introduction to the results contained in this dissertation.

## 1.1 Sequential Aggregate Signatures and Variants

Chapter 3 partially contains a version of the work by Lu, Ostrovsky, Sahai, Shacham, Waters [LOS06], the extended abstract of which appeared in the proceedings of EUROCRYPT 2006. We present an aggregate signature scheme, a multisignature scheme, and a verifiably encrypted signature scheme. Unlike previous such schemes, our constructions are provably secure without random oracles. Random-oracle-free schemes have become more attractive since a series of papers beginning with the uninstantiability result of Canetti, Goldreich, and Halevi [CGH98] has cast some doubt on the soundness of the random oracle methodology. Moreover, our proposed schemes are quite practical, and in some cases outperform the most efficient random-oracle-based schemes.

### 1.1.1 Related Work

An aggregate signature scheme allows a collection of signatures to be able to be compressed into one short signature. Aggregate signatures are useful for applications such as secure route attestation and certificate chains where the space requirements for a sequence of signatures can impact practical application performance.

Boneh et al. [BGL03] presented the first aggregate signature scheme, which was based on the BLS signature [BLS04] in bilinear groups. Lysyanskaya et al. [LMR04]

---

<sup>3</sup>Partially contains work by Goyal, Lu, Sahai, Waters[GLS08]

presented a sequential RSA-based scheme that, while more limited, could be instantiated using more general assumptions. In a sequential aggregate signature scheme, the aggregate signature must be constructed sequentially, with each signer modifying the aggregate signature in turn. However, in most known applications, the signatures are sequentially constructed anyway. One drawback of both schemes is that they are provably secure only in the random oracle model and thus there is only a heuristic argument for their security.

In a multisignature scheme, a single short object — the multisignature — can take the place of  $n$  signatures by  $n$  signers, all on the *same* message. (Aggregate signatures can be thought of as a multisignature without this restriction.) Boldyreva [Bol03] gave the first multisignature scheme in which multisignature generation does not require signer interaction, based on BLS signatures.

A verifiably encrypted signature is an object that anyone can confirm contains the encryption of a signature on some message, but from which only the party under whose key it was encrypted can recover the signature. Such a primitive is useful in contract signing. Boneh et al. [BGL03] gave the first verifiably encrypted signature scheme based on BLS signatures.

### 1.1.2 Our Contribution

All of our constructions derive from adaptations of the signature scheme of Waters [Wat05],  $\mathcal{W}$ , which follows from his identity-based encryption scheme.

We present the first aggregate signature scheme,  $\mathcal{WSA}$ , that is provably secure without random oracles. Our signatures are sequentially constructed, but unlike the scheme of Lysyanskaya et al. a verifier need not know the order in which the aggregate signature was created. Our signatures are also shorter than those of Lysyanskaya et al. and can be verified more efficiently than those of Boneh et al. We prove the security

of our scheme without random oracles under the registered key model and summarize this as the following theorem (details are found in Chapter 3, Section 3.2):

**Theorem 1.1.1.** *The sequential aggregate signature scheme  $\mathcal{WSA}$  is  $(t, q_C, q_S, n, \epsilon)$ -unforgeable if the underlying Waters signature scheme  $\mathcal{W}$  is  $(t', q', \epsilon')$ -unforgeable on  $\mathbb{G}$ , where*

$$t' = t + O(q_C + nq_S + n) \quad \text{and} \quad q' = q_S \quad \text{and} \quad \epsilon' = \epsilon .$$

In addition, in Chapter 3, Section 3.3, we present the first multisignature scheme,  $\mathcal{WM}$ , that is provably secure without random oracles. We state this as the following theorem, which we prove in the chapter:

**Theorem 1.1.2.** *The  $\mathcal{WM}$  multisignature scheme is  $(t, q, \epsilon)$ -unforgeable if the  $\mathcal{W}$  signature scheme is  $(t', q', \epsilon')$ -unforgeable, where*

$$t' = t + O(q) \quad \text{and} \quad q' = q \quad \text{and} \quad \epsilon' = \epsilon .$$

Finally, in Chapter 3, Section 3.4, we present two verifiably encrypted signature schemes that are provably secure without random oracles. The first,  $\mathcal{GVES}$ , is a generic construction from any existentially unforgeable signature scheme, CCA2-secure encryption scheme, and unbounded adaptive NIZK. We also give a concrete construction,  $\mathcal{WVES}$ , based on the Waters signature scheme. The security of these schemes requires not only that the signatures are unforgeable, but also that they must be opaque (these concepts are discussed in the chapter). We prove the following theorems in the chapter:

**Theorem 1.1.3.** *The  $\mathcal{GVES}$  verifiably encrypted signature scheme is unforgeable if the underlying signature scheme is unforgeable and the underlying NIZK scheme is adaptively unbounded.*

**Theorem 1.1.4.** *The  $\mathcal{GVES}$  verifiably encrypted signature scheme is opaque if the underlying signature scheme is unforgeable, the underlying encryption scheme is CCA2-secure, and the underlying NIZK scheme is adaptively unbounded.*

**Theorem 1.1.5.** *The  $\mathcal{WVES}$  verifiably encrypted signature scheme is  $(t, q_S, q_A, \epsilon)$ -unforgeable if the  $W$  signature scheme is  $(t', q', \epsilon')$ -unforgeable, where*

$$t' = t + O(q_S + q_A) \quad \text{and} \quad q' = q_S \quad \text{and} \quad \epsilon' = \epsilon .$$

**Theorem 1.1.6.** *The  $\mathcal{WVES}$  scheme is  $(t, q_S, q_A, \epsilon)$ -opaque if aggregate extraction is  $(t', \epsilon')$ -hard on  $\mathbb{G}$ , where*

$$t' = t + O(q_S + q_A) \quad \text{and} \quad q' = q_S \quad \text{and} \quad \epsilon' = 4kq_A\epsilon .$$

## 1.2 Non-Interactive Shuffle with Pairing-Based Verifiability

Chapter 4 partially contains a version of the work by Groth, Lu[GL07a], the extended abstract of which appeared in the proceedings of ASIACRYPT 2007.

In Chapter 4, we present an efficient non-interactive verifiable shuffle by using bilinear pairings. A shuffle is a permutation and re-encryption of a set of ciphertexts. Shuffles are used for instance in mix-nets [Cha81], which in turn are used in protocols for anonymous broadcast and electronic voting. In a typical construction of a mix-net, the users encrypt messages that they want to publish anonymously. They send the encrypted messages to a set of mix-net servers that will anonymize the messages. The first server permutes and re-encrypts the incoming set of messages, i.e. it carries out a shuffle. The next server takes the output from the first server and shuffles these ciphertexts. The protocol continues like this until all servers have permuted and re-encrypted the ciphertexts. After the mixing is complete, the mix-servers may now perform a threshold decryption operation to get out the permuted set of messages. The idea is

that if just one mix-server is honest, the messages will be randomly permuted and because of the re-encryption step nobody will know the permutation. The messages therefore appear in random order and cannot be traced back to the senders.

The mix-net protocol we just described is not secure if one of the mix-servers is dishonest. A dishonest mix-server could for instance discard some of the ciphertexts and inject new ciphertexts of its own choosing. It is therefore desirable to make the shuffle verifiable. An obvious way to make the mix-net verifiable is to ask each mix-server to provide a zero-knowledge proof of its shuffle being correct. The zero-knowledge proof guarantees that the shuffle is correct, yet reveals nothing about the permutation or the re-encryption and therefore preserves the privacy of the mix-net.

### **1.2.1 Related Work**

Much research has already been done on making shuffles verifiable by providing interactive proofs of correctness [SK95, Abe99, AH01, Nef01, FS01, Gro03, NSK04, NSK05, Fur05, Wik05, GL07b]. The proofs in these papers are all interactive and rely on the verifier choosing random challenges. Using the Fiat-Shamir heuristic, where the verifier's challenges are computed through the use of a cryptographic hash-function, it is possible to make these proofs non-interactive. As a heuristic argument for the security of these non-interactive proofs, one can prove them secure in the random oracle model [BR93], where the cryptographic hash-function is viewed as a random oracle that outputs a random string. However, Goldwasser and Kalai [GK03] demonstrate that the Fiat-Shamir heuristic sometimes yields insecure non-interactive proofs. Other works casting doubt on the Fiat-Shamir heuristic are [CGH98, Nie02, BBP04, CGH04].

It was an open problem to construct efficient non-interactive zero-knowledge (NIZK) proofs or arguments for the correctness of a shuffle that do not rely on the ran-



dom oracle model in the security proof. Such NIZK arguments can be used to reduce the round-complexity of protocols relying on verifiable shuffles. Moreover, interactive zero-knowledge proofs are usually deniable [Pas03]; a transcript of an interactive proof can only convince somebody who knows that the challenges were chosen correctly. NIZK arguments on the other hand are transferable. They consist of a single message that can be distributed and convince anybody that the shuffle is correct.

Obviously, one can apply general NIZK proof techniques to demonstrate the correctness of a shuffle. However, reducing the shuffle proof to a general NP statement and applying a general NIZK to it is very inefficient. Using NIZK techniques developed by Groth, Ostrovsky and Sahai [GOS06b, GOS06a, Gro06, GS08] one can get better performance. Some existing interactive zero-knowledge arguments for correctness of a shuffle naturally fit this framework. For example, it is possible to achieve non-interactive shuffle proofs of size  $O(n \log n)$  group elements for a shuffle of  $n$  ciphertexts by using Abe and Hoshino's scheme [AH01]. This kind of efficiency still falls short of what can be achieved using interactive techniques and the interactive proofs or arguments that grow linearly in the size of the shuffle do not seem easy to make non-interactive using the techniques of Groth, Ostrovsky and Sahai.

### 1.2.2 Our Contribution

In Chapter 4, we offer the first (efficient) non-interactive zero-knowledge argument for correctness of a shuffle. The NIZK argument is in the common reference string model and has perfect zero-knowledge. The security proof of our scheme does not rely on the random oracle model. Instead we make use of recently developed techniques for making non-interactive witness-indistinguishable proofs for bilinear groups by Groth and Sahai [GS08], which draws on earlier work by Groth, Ostrovsky and Sahai [GOS06b, GOS06a, Gro06].

The NIZK argument we suggest is for the correctness of a shuffle of BBS ciphertexts. This cryptosystem, suggested by Boneh, Boyen and Shacham [BBS04], has ciphertexts that consist of 3 group elements for each group element that they encrypt. We consider statements consisting of  $n$  input ciphertexts and  $n$  output ciphertexts and the claim that the output ciphertexts are a shuffle of the input ciphertexts. Our NIZK arguments,  $GL\text{-}SHUF$ , consist of  $15n$  group elements, which is reasonable in comparison with the statement size, which is  $6n$  group elements. In Chapter 4, we prove the following theorem:

**Theorem 1.2.1.** *The protocol  $GL\text{-}SHUF$  is a non-interactive perfectly complete, computationally  $R_{\text{co}}$ -sound, perfect zero-knowledge argument of a correct shuffle of BBS ciphertexts under the Decisional Linear Assumption, Permutation Pairing Assumption, and Simultaneous Pairing Assumption.*

### 1.3 Accountable Authority Identity-Based Encryption

Chapter 5 partially contains a version of the work by Goyal, Lu, Sahai, and Waters[GLS08]. The extended abstract of this work appeared in the proceedings of the ACM Conference on Computer and Communications Security 2008.

In Chapter 5, we define and give two constructions of an accountable authority identity-based encryption scheme. Shamir [Sha85] introduced the concept of identity-based encryption (IBE) as an approach to simplify public key and certificate management in a public key infrastructure (PKI). One of the first practical and fully functional IBE schemes was proposed by Boneh and Franklin [BF01, BF03] in the random oracle model (another efficient solution was independently proposed by Cocks [Coc01]). Following that work, a rapid development of identity-based PKI has taken place (see [CHK03, BB04a, BB04b, BBG05, Wat05, Gen06] and the references therein).

In an IBE system, the public key of a user may be an arbitrary string, such as an e-mail address or other identifier. Of course, users are not capable of generating a private key for an identity themselves. For this reason, there is a trusted party called the private key generator (PKG) who does the system setup. To obtain a private key for his identity, a user would go to the PKG and prove his identity. The PKG would then generate the appropriate private key and pass it on to the user.

Such a setting, however, leads to the following problem. Since the PKG is able to compute the private key corresponding to any identity, it must be completely trusted. The PKG is free to engage in malicious activities without any risk of being confronted in a court of law. The malicious activities could include decrypting and reading messages meant for any user, or worse still, generating and distributing private keys for any identity. This, in fact, has been cited as a reason for the slow adoption of IBE despite its nice properties in terms of usability. It has been argued that due to the inherent key escrow problem, the use of IBE is restricted to small and closed groups where a central trusted authority is available [AP03, LBD04, Gen03].

**Accountable Authority Identity-Based Encryption.** Goyal [Goy07] introduced the notion of Accountable Authority Identity-Based Encryption (A-IBE) as a new approach to mitigate the above problem of trust. The proposal was to have an algorithm that, provided with two keys for the same user, could determine if they came from the same “family” of keys. This algorithm would be used to implicate a malicious party, whether it was the user or the PKG. Informally speaking, the simplified view of the approach is as follows:

1. In the IBE scheme, there will be an exponential (or super-polynomial) number of possible decryption keys corresponding to every identity ID.
2. A user gets the decryption key corresponding to his identity from the PKG using

a secure *key generation protocol*. The protocol allows the user to obtain a single decryption key  $d_{\text{ID}}$  for his identity *without letting the PKG know which key he obtained*.

3. Now if the PKG generates a decryption key  $d'_{\text{ID}}$  for malicious usage, with all but negligible probability, it will be different from the key  $d_{\text{ID}}$  which the user obtained. Hence the key pair  $(d_{\text{ID}}, d'_{\text{ID}})$  is a cryptographic proof of malicious behavior by the PKG (since in normal circumstances, only one key per identity should be in circulation).
4. Given one decryption key for an identity, it is intractable (for the user) to find any other. This is so that a dishonest user cannot frame the PKG.

Thus, this approach severely restricts the PKG as far as malicious distribution of the private keys is concerned. The knowledge of the key  $d'_{\text{ID}}$  enables an entity  $E$  to go to the honest user  $U$  (with identity  $\text{ID}$  and having key  $d_{\text{ID}}$ ) and together with him, sue the PKG by presenting the pair  $(d'_{\text{ID}}, d_{\text{ID}})$  as a proof of fraud.

### 1.3.1 Related Work

As mentioned above, the idea of an accountable authority IBE was introduced by Goyal [Goy07] as a mitigation to the problem of trust in the PKG. Au et al. [AHL08] extended this work by introducing a retrieval algorithm which causes the PKG's master secret key to be revealed if more than one key per identity is released. The motivation is to penalize the PKG without the users having to go to the court. However, this work is orthogonal to ours since their security proofs are in the white box model of security (as opposed to black-box or even weakly black-box) and require the PKG to release a well formed decryption key. To our knowledge, these are the only known mitigation approaches without using multiple PKGs.

On the multiple PKGs side, Boneh and Franklin [BF01, BF03] proposed an efficient approach to make the PKG distributed in their scheme using techniques from threshold cryptography. Lee *et al* [LBD04] proposed a variant of this approach using multiple key privacy agents (KPAs).

### 1.3.2 Our Contribution

**The Right Model for A-IBE.** Goyal [Goy07] presented two constructions towards achieving the notion of A-IBE. However, his security proofs could only provide a very limited guarantee: that the PKG cannot maliciously distribute a *well-formed* decryption key. As noted by Goyal, while this is a starting point, these kind of “white box” guarantees are completely insufficient in practice. The PKG could, for example, release an obfuscated program (or simply a decryption box) that successfully decrypts the ciphertexts and yet does not contain the decryption key in any canonical form. Furthermore, trivial constructions can satisfy the “white box” security guarantee and clearly be insecure in practice: For instance, if we take any IBE scheme and force the user to also obtain a blind signature from the PKG on a random message (which is checked by the decryption algorithm), this would already satisfy the “white box” security definition. Obviously this scheme would be completely broken in practice since the PKG could release a box that decrypts for an identity but does not contain a signature (and therefore is not well-formed).

Goyal also showed how to extend his constructions to achieve security guarantees according to a *weak black-box model*, in which a malicious PKG has to output a decryption box just after running the key generation protocol with the honest user. However, this security model is also insufficient. It is conceivable that the PKG (or a party colluding with the PKG) could trick the user into decrypting a maliciously prepared ciphertext and see the result (in an attempt to learn more information about the

decryption key which the user selected during the key generation protocol). Indeed, if such decryption queries are allowed, the weak black-box scheme of [Goy07] can be completely broken with only a small number of queries.

In what we call the *full black-box model*, the PKG is given access to decryption queries and no assumptions are made regarding how the decryption box works. In particular, just by observing the input/output behavior of the given decryption box, a judge should be able to decide if the box was created by the actual user or by a dishonest PKG. The construction of an A-IBE scheme in the full black-box model — the model which we believe provides the “right” real world security guarantees — was left as an important open problem in [Goy07].

In this work, we resolve the above open question and provide constructions of (fully black-box) A-IBE schemes. Our main result is a general construction of an A-IBE scheme from *black-box* use of any IBE scheme, oblivious transfer protocol, and “sufficiently expressive”<sup>4</sup> key-policy attribute-based encryption scheme. We then give a concrete application of our general construction, which results in an efficient realization of an A-IBE scheme that is secure under the Decisional Bilinear Diffie-Hellman (DBDH) assumption. Under the DBDH assumption, we also present a more efficient concrete construction of an A-IBE scheme that does not make use of the general construction.

The main technical difficulty is resolving the tension between the information leaked as part of the decryption queries and the success of the exoneration procedure. That is, on one hand we require that during regular operation, the outcome of the decryption of a ciphertext should not leak information about which decryption key the user selected. On the other hand, during exoneration, a judge should be able to extract enough information about the user key selection from the decryption box in order to

---

<sup>4</sup>We state the exact requirements in Chapter 5, Section 5.3.1.

determine that the user could not have generated the box (and therefore the PKG must be at fault).

The tension is resolved by a novel combinatorial construction. The key idea in all our constructions is to design a scheme having *imperfect completeness*. That is, for every possible decryption key, there exist a negligible fraction of (valid) ciphertexts which cannot be decrypted by this key. On one hand, this property is helpful in tracing: a judge (given the decryption box and the decryption key of the user) can probe the box exactly on those ciphertexts which the user key should not be able to decrypt. On the other hand, this does not seem to create a problem for decryption queries since the chance that a malicious PKG will hit such a ciphertext (with a polynomial number of queries) is negligible.

We construct such a scheme using ideas from key-policy attribute-based encryption (KP-ABE) [SW05, GPS06, OSW07]. Very roughly, we label each ciphertext as well as a decryption key with a list of dummy attributes. There exists a policy which decides whether or not a ciphertext will be decrypted by a particular private key. To achieve statistical completeness, for every decryption key, all but a negligible fraction of ciphertexts will satisfy this policy.

We summarize the main results in Chapter 5. Given an IBE scheme  $\mathcal{IBE}$ , a  $k$ -out-of- $n$  oblivious transfer protocol  $\mathcal{OT}$ , and a sufficiently expressive KP-ABE scheme  $\mathcal{ABE}$ , we will show how to construct an accountable authority IBE scheme  $\mathcal{AIBE}$  only using  $\mathcal{IBE}$ ,  $\mathcal{OT}$ ,  $\mathcal{ABE}$  as black-boxes. We refer to this generic compilation as the  $\mathcal{AIBE}\text{-}\mathcal{GEN}$  scheme. In Chapter 5, Section 5.4, we prove the following theorems:

**Theorem 1.3.1.** *The advantage of an adversary in the IND-ID-CPA game is negligible for  $\mathcal{AIBE}\text{-}\mathcal{GEN}$  assuming the underlying IBE scheme is IND-ID-CPA secure.*

**Theorem 1.3.2.** *Assuming that the underlying OT is fully simulatable (secure as per the ideal/real world security definition [Can00]), the advantage of any adversary in*

*the DishonestPKG game is negligible for  $\mathcal{AI}\mathcal{BE}\text{-}\mathcal{GEN}$ .*

**Theorem 1.3.3.** *Assuming that the underlying OT is fully simulatable (secure as per the ideal/real world security definition [Can00]) and the underlying ABE scheme is Selective-Set secure, the advantage of any adversary in the Selective-ID DishonestUser game is negligible for  $\mathcal{AI}\mathcal{BE}\text{-}\mathcal{GEN}$ .*

In Chapter 5, Section 5.5, we give a concrete instantiation of the generic scheme using the IBE scheme due to Waters [Wat05], the OT protocol due to Lindell [Lin08], and the KP-ABE scheme due to Goyal et al. [GPS06]. This gives rise to our  $\mathcal{AI}\mathcal{BE}\text{-}1$  scheme. The security of this scheme is inherited from the security of the generic A-IBE scheme.



## CHAPTER 2

### Preliminaries and Notation

In this chapter, we review preliminary concepts and set the notation for the remaining chapters.

#### 2.1 Groups with Efficiently Computable Bilinear Maps

We briefly review the necessary facts about bilinear maps and groups associated with these maps.

Consider the following setting:

**Large prime:** Let  $p$  be a prime (with  $\log p$  polynomially related to the security parameter). Let  $\mathbb{G}$  and  $\mathbb{G}_T$  be groups of order  $p$ .

**Source Group:**  $\mathbb{G}$  is called the *source group* and is written using additive notation.

**Target Group:**  $\mathbb{G}_T$  is called the *target group* and is written using multiplicative notation.

**Generator:** Let  $P$  be a generator of  $\mathbb{G}$ .

**Bilinear Map:** Let  $e: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$  be a map with the following properties:

- **Bilinear:** For all  $u, v \in \mathbb{G}$  and  $a, b \in \mathbb{Z}/p\mathbb{Z}$ ,  $e(a \cdot u, b \cdot v) = e(u, v)^{ab}$ ;
- **Non-trivial:**  $e(P, P)$  is a generator of  $\mathbb{G}_T$ .

**Efficiency:** Group operations in  $\mathbb{G}, \mathbb{G}_T$  and the bilinear map can be computed efficiently and group membership is efficiently decidable.

We say that  $\mathbb{G}$  (or the tuple  $(p, \mathbb{G}, \mathbb{G}_T, e, P)$ ) is a bilinear group if it satisfies these requirements. For more detail regarding the construction of these groups, see e.g. [Gal05, Pat05].

## 2.2 Asymptotic and Concrete Security

The security of modern cryptographic schemes is demonstrated via a reductionist approach. A reduction relates the hardness of a complicated scheme to that of a simpler underlying assumption. Examples of such assumptions are the existence of cryptographic primitives or the hardness of concrete mathematical problems. Traditionally, in complexity-based cryptography, these reductions are asymptotic relative to a *security parameter*  $k$ . Reductions of this nature usually take the form of “If no (non-uniform) PPT algorithm can break some underlying assumption  $\mathcal{P}$  with non-negligible probability (in  $k$ ), then there are no (non-uniform) PPT algorithms that can break our security notion  $\mathcal{S}$  with non-negligible probability”.

Taking another approach, a more accurate measure is desirable in real-world implementations, so we can also speak of concrete security. In such a setting, we spell out the security reduction in terms of certain metrics (e.g. running time, memory, oracle queries). When working in the context of concrete security, we speak in explicit terms – if no PPT algorithm that takes at most  $t$  steps can break some underlying assumption with probability more than  $\epsilon$  then no PPT algorithm that takes at most  $t'$  steps can break our security notion  $\mathcal{S}$  with probability more than  $\epsilon'$ , where  $t'$  and  $\epsilon'$  are written in terms of  $t$  and  $\epsilon$ .

Several results in the dissertation are stated using concrete security. However, due

to the fact that some of the schemes are not optimized for concrete security, we will also state the security of our schemes using asymptotic security when it is convenient.

## 2.3 Computational Assumptions

In this section, we review a few computational assumptions that will be used in subsequent chapters. These assumptions are made in the context of bilinear groups. For concrete security, we fix the tuple  $(p, \mathbb{G}, \mathbb{G}_T, e, P)$  and let it be globally known (as GK). On the other hand, for asymptotic security, the tuple is generated by  $\text{Setup}(1^k)$ , which is a (randomized) setup algorithm that takes a security parameter as its input.

### 2.3.1 Computational Diffie-Hellman

Define the advantage of an algorithm  $\mathcal{A}$  in solving the Computational Diffie-Hellman problem as

$$\mathbf{Adv}_{\mathcal{A}}^{\text{cdh}} := \Pr \left[ a, b \xleftarrow{R} \mathbb{Z}/p\mathbb{Z} : \mathcal{A}(g, aP, bP) = abP \right] .$$

The probability is over the uniform random choice of  $a$  from  $\mathbb{Z}/p\mathbb{Z}$ , and the coin tosses of  $\mathcal{A}$ . In terms of concrete security, we say that an algorithm  $\mathcal{A}$   $(t, \epsilon)$ -breaks Computational Diffie-Hellman if  $\mathcal{A}$  runs in time at most  $t$ , and  $\mathbf{Adv}_{\mathcal{A}}^{\text{cdh}}$  is at least  $\epsilon$ .

In terms of asymptotic security, the probability is also taken over the coin tosses of  $\text{Setup}$ . We say that a (non-uniform) algorithm  $\mathcal{A}$  breaks Computational Diffie-Hellman if its running time  $t(k)$  is polynomial and its success probability  $\mathbf{Adv}_{\mathcal{A}}^{\text{cdh}}(k)$  is non-negligible in the security parameter  $k$ .

**Definition 2.3.1.** We say that the  $((t, \epsilon)$ -Computational Diffie-Hellman assumption holds if no adversary can  $((t, \epsilon)$ -break the Computational Diffie-Hellman problem.

### 2.3.2 Decisional Linear Assumption

We recap the Decisional Linear Problem introduced by Boneh, Boyen and Shacham [BBS04]. It can be loosely stated as: Given  $A, B \xleftarrow{R} \mathbb{G}$  and given  $sA, tB, zP \in G$ , decide if  $z = s + t$ .

Define the advantage of an algorithm  $\mathcal{A}$  in solving the Decisional Linear problem as

$$\mathbf{Adv}_{\mathcal{A}}^{\text{dlin}} := \left| \Pr \left[ A, B \xleftarrow{R} \mathbb{G}; s, t \xleftarrow{R} \mathbb{Z}/p\mathbb{Z} : \mathcal{A}(A, B, sA, tB, (s+t)P) = 1 \right] - \Pr \left[ A, B \xleftarrow{R} \mathbb{G}; s, t, z \xleftarrow{R} \mathbb{Z}/p\mathbb{Z} : \mathcal{A}(A, B, sA, tB, zP) = 1 \right] \right|.$$

The probabilities are taken over the uniform random choice of  $A$  and  $B$  from  $\mathbb{G}$ , of  $s, t, z$  from  $\mathbb{Z}/p\mathbb{Z}$ , and the coin tosses of  $\mathcal{A}$ . In terms of concrete security, we say that an algorithm  $\mathcal{A}$   $(t, \epsilon)$ -breaks the Decisional Linear Assumption if  $\mathcal{A}$  runs in time at most  $t$ , and  $\mathbf{Adv}_{\mathcal{A}}^{\text{dlin}}$  is at least  $\epsilon$ .

In terms of asymptotic security, the probability is also taken over the coin tosses of **Setup**. We say that a (non-uniform) algorithm  $\mathcal{A}$  breaks the Decisional Linear Assumption if its running time  $t(k)$  is polynomial and its success probability  $\mathbf{Adv}_{\mathcal{A}}^{\text{dlin}}(k)$  is non-negligible in the security parameter  $k$ .

**Definition 2.3.2.** We say that the  $((t, \epsilon)$ -)Decisional Linear Assumption holds if no adversary can  $((t, \epsilon)$ -)break the Decisional Linear problem.

### 2.3.3 Decisional Bilinear Diffie-Hellman (DBDH) Assumption

Define the advantage of an algorithm  $\mathcal{A}$  in solving the Decisional Bilinear Diffie-Hellman [BB04a] problem as

$$\mathbf{Adv}_{\mathcal{A}}^{\text{dbdh}} := \left| \Pr \left[ a, b, c \xleftarrow{R} \mathbb{Z}/p\mathbb{Z} : \mathcal{A}(aP, bP, cP, e(P, P)^{abc}) = 1 \right] - \Pr \left[ a, b, c, z \xleftarrow{R} \mathbb{Z}/p\mathbb{Z} : \mathcal{A}(aP, bP, cP, e(P, P)^z) = 1 \right] \right|.$$

The probabilities are taken over the uniform random choice of  $a, b, c, z$  from  $\mathbb{Z}/p\mathbb{Z}$ , and the coin tosses of  $\mathcal{A}$ . In terms of concrete security, we say that an algorithm  $\mathcal{A}$   $(t, \epsilon)$ -breaks Decisional Bilinear Diffie-Hellman if  $\mathcal{A}$  runs in time at most  $t$ , and  $\text{Adv}_{\mathcal{A}}^{\text{dbdh}}$  is at least  $\epsilon$ .

In terms of asymptotic security, the probability is also taken over the coin tosses of  $\text{Setup}$ . We say that a (non-uniform) algorithm  $\mathcal{A}$  breaks Decisional Bilinear Diffie-Hellman if its running time  $t(k)$  is polynomial and its success probability  $\text{Adv}_{\mathcal{A}}^{\text{dbdh}}(k)$  is non-negligible in the security parameter  $k$ .

**Definition 2.3.3.** We say that the  $((t, \epsilon)$ -)Decisional Bilinear Diffie-Hellman assumption holds if no adversary can  $((t, \epsilon)$ -)break the Decisional Bilinear Diffie-Hellman problem.

## CHAPTER 3

### Sequential Aggregate Signatures and Variants

We briefly describe the organization of this chapter (see the introductory discussion in Chapter 1, Section 1.1).

In this chapter, we present an aggregate signature scheme, a multisignature scheme, and a verifiably encrypted signature scheme. Unlike previous such schemes, our constructions are provably secure without random oracles. We begin in Section 3.1 by giving background information about signatures and review the Waters [Wat05] signature scheme. In Section 3.2, we define what a secure sequential aggregate signature scheme is and present the construction of our  $WSA$  scheme. In Section 3.3, we give the definition of a secure multisignature scheme and present the construction of our  $WM$  scheme. In Section 3.4, we give the definition of a secure verifiably encrypted signature scheme and present two constructions: a generic construction  $GVES$  (built from any secure signature scheme, NIZK, and encryption scheme), and our efficient pairing-based construction  $WVES$ . Due to the subtlety of the security proofs, we dedicate Section 3.5 to the proof of security for  $WVES$ . We compare our proposed schemes to prior work in Section 3.6, and conclude in Section 3.7.

#### 3.1 Background

In this section, we introduce the background necessary for the remainder of the chapter. We give the definition of existentially unforgeable signatures and review the Wa-

ters [Wat05] signature scheme. Further in the chapter, we will make use of the Computational Diffie-Hellman assumption, which we discussed in Chapter 2, Section 2.3.1. We begin with an optimization that may be applied to the schemes presented in this chapter.

**Asymmetric Pairings and Short Representations.** It is a simple (though tedious) matter to rewrite our schemes to employ an asymmetric pairing  $e: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ . Signatures will then include elements of  $\mathbb{G}_1$ , while public keys will include elements of  $\mathbb{G}_2$  and  $\mathbb{G}_T$ . This setting allows us to take advantage of curves due to Barreto and Naehrig [BN05]. With these curves, elements of  $\mathbb{G}_1$  have a 160-bit representation at the 1024-bit security level.<sup>1</sup> In this case, security follows from a different assumption known as the Computational co-Diffie-Hellman problem [BLS04].

### 3.1.1 Existentially Unforgeable Signatures

A signature scheme is a three-tuple of (randomized) algorithms  $\mathcal{S} = (\text{KeyGen}, \text{Sign}, \text{Verify})$ . These algorithms behave as follows.

**KeyGen.** This algorithm generates a private key  $\text{SK}$  for signing and a public key  $\text{PK}$  for verification of signatures.

**Sign( $\text{SK}, M$ ).** This algorithm signs the message  $M$  using the user's private key  $\text{SK}$  and outputs the signature  $\sigma$ .

**Verify( $\text{PK}, M, \sigma$ ).** This algorithm takes a message  $M$  and a purported signature  $\sigma$  under the public key  $\text{PK}$  and outputs `valid` or `invalid`.

---

<sup>1</sup>By “1024-bit security,” we mean parameters such that the conjectured complexity of computing discrete logarithms is roughly comparable to the complexity of factoring 1024-bit numbers. For a more refined analysis see Koblitz and Menezes [KM05].

We say that the scheme is perfectly correct if for all messages  $M$ , it satisfies

$$\Pr[(PK, SK) \leftarrow \text{KeyGen}; \sigma \leftarrow \text{Sign}(SK, M) : \text{Verify}(PK, M, \sigma) = \text{valid}] = 1 .$$

The probability is taken over the randomness of the key generation, signing, and verification algorithms.

We review the standard notion of signature security—existential unforgeability under chosen-message attack—due to Goldwasser, Micali, and Rivest [GMR88]. We define this as game where an algorithm  $\mathcal{A}$  attempts to forge a signature for the scheme  $\mathcal{S}$ . The algorithm wins if the challenger outputs 1 in the following game:

**Setup.** The challenger obtains  $(PK, SK) \leftarrow \text{KeyGen}$  and runs  $\mathcal{A}$  on input  $PK$ .

**Signing Queries.** Algorithm  $\mathcal{A}$  requests a signature on a message  $M$ . The challenger obtains a signature  $\sigma = \text{Sign}(SK, M)$  and provides it to  $\mathcal{A}$ .

**Output.** Algorithm  $\mathcal{A}$  outputs a message  $M^*$  and a signature  $\sigma^*$ . If  $M^*$  has not been queried before and  $\text{Verify}(PK, M^*, \sigma^*) = \text{valid}$  then the challenger outputs 1, otherwise 0.

**Definition 3.1.1.** We say that a signature scheme is  $(t, q_s, \epsilon)$  secure if no  $t$ -time adversary making  $q_s$  signing queries can win the above game with advantage more than  $\epsilon$ .

### 3.1.2 The Waters Signature Scheme

We describe the Waters signature scheme [Wat05]. In our description the messages will be signatures on bitstrings of the form  $\{0, 1\}^k$  for some fixed  $k$ . However, in practice one could apply a collision-resistant hash function  $H_k: \{0, 1\}^* \rightarrow \{0, 1\}^k$  to sign messages of arbitrary length.



The scheme requires, besides the random generator  $P \in \mathbb{G}$ ,  $k+1$  additional random generators  $u', u_1, \dots, u_k \in \mathbb{G}$ . In the basic scheme, these can be generated at random as part of system setup and shared by all users. In some of the variants below, each user has generators  $(u', u_1, \dots, u_k)$  of her own, which must be included in her public key. We will draw attention to this in introducing the individual schemes.

The Waters signature scheme is a three-tuple of algorithms  $\mathcal{W} = (\text{W-KeyGen}, \text{W-Sign}, \text{W-Verify})$ . These are constructed as follows.

**W-KeyGen.** Pick random  $\alpha \xleftarrow{R} \mathbb{Z}/p\mathbb{Z}$  and set  $A \leftarrow e(P, P)^\alpha$ . The public key PK is  $A \in \mathbb{G}_T$ . The private key SK is  $\alpha$ .

**W-Sign(SK, M).** Parse the user's private key SK as  $\alpha \in \mathbb{Z}/p\mathbb{Z}$  and the message  $M$  as a bitstring  $(m_1, \dots, m_k) \in \{0, 1\}^k$ . Pick a random  $r \xleftarrow{R} \mathbb{Z}/p\mathbb{Z}$  and compute

$$S_1 \leftarrow \alpha P + r \left( u' + \sum_{i=1}^k m_i u_i \right) \quad \text{and} \quad S_2 \leftarrow r P . \quad (3.1)$$

The signature is  $\sigma = (S_1, S_2) \in \mathbb{G}^2$ .

**W-Verify(PK, M,  $\sigma$ ).** Parse the user's public key PK as  $A \in \mathbb{G}_T$ , the message  $M$  as a bitstring  $(m_1, \dots, m_k) \in \{0, 1\}^k$ , and the signature  $\sigma$  as  $(S_1, S_2) \in \mathbb{G}^2$ . Verify that

$$e(S_1, P) \cdot e \left( S_2, u' + \sum_{i=1}^k m_i u_i \right)^{-1} \stackrel{?}{=} A \quad (3.2)$$

holds; if so, output valid; if not, output invalid.

## 3.2 Sequential Aggregate Signatures

In a sequential aggregate signature, as in an ordinary aggregate signature, a single short object — called the aggregate — takes the place of  $n$  signatures by  $n$  signers on

$n$  messages. Thus aggregate signatures are a generalization of multisignatures. Sequential aggregates differ from ordinary aggregates in that the aggregation operation is performed by each signer in turn, rather than by an unrelated party after the fact.

Aggregate signatures have many applications, as noted by Boneh et al. [BGL03] and Lysyanskaya et al. [LMR04]. Below, we consider two: Secure BGP route attestation and proxy signatures.

In BGP, routers generate and forward route attestations to other routers to advertise the routes which should be used to reach their networks. Secure BGP solves the problem of attestation forgery by having each router add its signature to a valid attestation before forwarding it to its neighbors. Because the size of route attestations is limited, aggregate signatures are useful in reducing the overhead of multiple signatures along a path. Nicol, Smith, and Zhao [NSZ04] gave a detailed analysis of the application of aggregate signatures to the Secure BGP routing protocol [KLS00]. Our sequential aggregate signature scheme is well suited for improving SBGP. Since all of the incoming route attestations need to be verified anyway, the fact that our signing algorithm requires a verification adds no overhead. Additionally, our signature scheme can have signatures that are smaller than those of Lysyanskaya et al. and verification will be faster than that of the Boneh et al. scheme.

A proxy signature scheme allows a user, called the *designator*, to delegate signing authority to another user, the *proxy signer*. This signature primitive, introduced by Mambo, Usada, and Okamoto [MUO96], has been discussed and used in several practical applications. Boldyreva, Palacio, and Warinschi [BPW03] show how to construct a secure proxy signature scheme from any aggregate (or sequential aggregate) signature scheme. Instantiating the Boldyreva-Palacio-Warinschi construction with our scheme, we obtain a practical proxy signature secure without random oracles.

### 3.2.1 Definitions

A sequential aggregate signature scheme includes three algorithms. The first, **KeyGen**, is used to generate public-private keypairs. The second, **AggSign**, takes not only a private key and a message to sign, as does an ordinary signing algorithm, but also an aggregate-so-far by a set of  $l$  signers on  $l$  corresponding messages; it folds the new signature into the aggregate, yielding a new aggregate signature by  $l+1$  signers on  $l+1$  messages. The third algorithm, **AVerify**, takes a purported aggregate signature, along with  $l$  public keys and  $l$  corresponding messages, and decides whether the aggregate is valid.

**The Sequential Aggregate Certified-Key Model.** Because our aggregate signature behaves like a sequential aggregate signature from the signers' viewpoint, but like a standard aggregate signature from the verifiers' viewpoint, we describe a security model for it that is a hybrid of the sequential aggregate chosen key model of Lysyanskaya et al. [LMR04] and the aggregate chosen key model of Boneh et al. [BGL03]. In both models, the adversary is given a single challenge key, along with an appropriate signing oracle for that key. His goal is to generate a sequential aggregate that frames the challenge user. The adversary is allowed to choose all the keys in that forged aggregate but the challenge key.

We prove our scheme in a more restricted model that requires that the adversary certify that the public keys it includes in signing oracle queries and in its forgery were properly generated. This we handle by having the adversary hand over the private keys before using the public keys. We could also extract the keys by rewinding or, if this is impossible, using the NIZKs proposed by Groth, Ostrovsky, and Sahai [GOS06b].

Formally, the advantage of a forger  $\mathcal{A}$  in our model is the probability that the challenger outputs 1 in the following game:

**Setup.** Initialize the list of certified public keys  $C = \emptyset$ . Then generate  $(\mathbf{PK}, \mathbf{SK}) \leftarrow \text{KeyGen}$ . Run algorithm  $\mathcal{A}$  with  $\mathbf{PK}$  as input.

**Certification Queries.** Algorithm  $\mathcal{A}$  provides a keypair  $(\mathbf{PK}', \mathbf{SK}')$  in order to certify  $\mathbf{PK}'$ . Add  $\mathbf{PK}'$  to  $C$  if  $\mathbf{SK}'$  is its matching private key.

**Signing Queries.** Algorithm  $\mathcal{A}$  requests a sequential aggregate signature, under the challenge key  $\mathbf{PK}$ , on a message  $M$ . In addition, it supplies an aggregate-so-far  $\sigma'$  on messages  $M$  under keys  $\mathbf{PK}$ . Check that the signature  $\sigma'$  verifies; that each key in  $\mathbf{PK}$  is in  $C$ ; that  $\mathbf{PK}$  does not appear in  $\mathbf{PK}$ ; and that  $|\mathbf{PK}| < n$ . If any of these fails to hold, answer invalid. Otherwise respond with  $\sigma = \text{AggSign}(\mathbf{SK}, M, \sigma', M, \mathbf{PK})$ .

**Output.** Eventually,  $\mathcal{A}$  halts, outputting a forgery  $\sigma^*$  on messages  $M$  under keys  $\mathbf{PK}$ . This forgery must verify as valid under  $\text{AVerify}$ ; each key in  $\mathbf{PK}$  (except the challenge key) must be in  $C$ ; and  $|\mathbf{PK}| \leq n$  must hold. In addition, the forgery must be nontrivial: the challenge key  $\mathbf{PK}^*$  must appear in  $\mathbf{PK}$ , WLOG at index 1 (since signature verification in our scheme has no inherent order), and the corresponding message  $M[1]$  must not have been queried by  $\mathcal{A}$  of its sequential aggregate signing oracle. Output 1 if all these conditions hold, 0 otherwise.

We say that an aggregate signature scheme is  $(t, q_C, q_S, n, \epsilon)$  secure if no  $t$ -time adversary making  $q_C$  certification queries and  $q_S$  signing queries can win the above game with advantage more than  $\epsilon$ , where  $n$  is an upper bound on the length of the sequential aggregates involved.

### 3.2.2 The WSA Sequential Aggregate Signature Scheme

We start by giving some intuition for our scheme. Each signer in our scheme will have a unique public key from the Waters signature scheme

$$u', \mathbf{u} = (u_1, \dots, u_k), A \leftarrow e(P, P)^\alpha.$$

While in the original signature scheme the private key consists only of  $\alpha$ , in our aggregate signature scheme it is important that the private key holder will additionally choose and remember the discrete logs of  $u', \mathbf{u} = (u_1, \dots, u_k)$ . In the Waters signature scheme, signatures are made of two group elements  $S_1$  and  $S_2$ . At a high level, we can view  $S_2$  as some randomness for the signature and  $S_1$  as the signature on a message relative to that randomness.

An aggregate signature in our scheme also consists of group elements  $S'_1, S'_2$ . The second element  $S'_2$  again consists of some “shared” randomness for the signature. When a signer wishes to add his signature on a message to an aggregate  $(S'_1, S'_2)$ , he simply figures out what his  $S_1$  component would be in the underlying signature scheme given  $S'_2$  as the randomness. In order to perform this computation the signer must know the discrete log values of all of his public generators. He then multiplies this value into  $S'_1$  and finally re-randomizes the signature.

We now formally describe the sequential aggregate obtained from the Waters signature.

Our sequential aggregate scheme is a three-tuple of algorithms  $\mathcal{WSA} = (\text{KeyGen}, \text{AggSign}, \text{AVerify})$ . These behave as follows.

**WSA-KeyGen.** Pick random  $\alpha, y' \xleftarrow{R} \mathbb{Z}/p\mathbb{Z}$  and a random vector  $\mathbf{y} = (y_1, \dots, y_k) \xleftarrow{R} (\mathbb{Z}/p\mathbb{Z})^k$ . Compute

$$u' \leftarrow y'P \quad \text{and} \quad \mathbf{u} = (u_1, \dots, u_k) \leftarrow (y_1P, \dots, y_kP) \quad \text{and} \quad A \leftarrow e(P, P)^\alpha.$$

The user's private key is  $\text{SK} = (\alpha, y', y) \in (\mathbb{Z}/p\mathbb{Z})^{k+2}$ . The public key is  $\text{PK} = (A, u', u) \in \mathbb{G}_T \times \mathbb{G}^{k+1}$ ; it must be certified to ensure knowledge of the corresponding private key.

**WSA-AggSign**( $\text{SK}, M, \sigma', \text{M}, \text{PK}$ ). The input is a private key  $\text{SK}$ , to be parsed as  $(\alpha, y', y_1, \dots, y_k) \in (\mathbb{Z}/p\mathbb{Z})^{k+2}$ ; a message  $M$  to sign, parsed as  $(m_1, \dots, m_k) \in \{0, 1\}^k$ ; and an aggregate-so-far  $\sigma'$  on messages  $\text{M}$  under public keys  $\text{PK}$ . Verify that  $\sigma'$  is valid by calling **AVerify**( $\sigma', \text{M}, \text{PK}$ ); if not, output `fail` and halt. Check that the public key corresponding to  $\text{SK}$  does not already appear in  $\text{PK}$ ; if it does, output `fail` and halt. (We revisit the issue of having one signer sign multiple messages below.)

Otherwise, parse  $\sigma'$  as  $(S'_1, S'_2) \in \mathbb{G}^2$ . Set  $l \leftarrow |\text{PK}|$ . Now, for each  $i$ ,  $1 \leq i \leq l$ , parse  $\text{M}[i]$  as  $(m_{i,1}, \dots, m_{i,k}) \in \{0, 1\}^k$ , and parse  $\text{PK}[i]$  as  $(A_i, u'_i, u_{i,1}, \dots, u_{i,k}) \in \mathbb{G}_T \times \mathbb{G}^{k+1}$ . Compute

$$w_1 \leftarrow S'_1 + \alpha P + (y' + \sum_{j=1}^k y_j m_j) S'_2 \quad \text{and} \quad w_2 \leftarrow S'_2. \quad (3.3)$$

The values  $(w_1, w_2)$  form a valid signature on  $\text{M} \| M$  under keys  $\text{PK} \| \text{PK}$ , but this signature needs to be re-randomized: otherwise whoever created  $\sigma'$  could learn the user's private key  $\alpha P$ . Choose a random  $\tilde{r} \in \mathbb{Z}/p\mathbb{Z}$ , and compute

$$S_1 \leftarrow w_1 + \tilde{r} (u' + \sum_{j=1}^k m_j u_j) + \sum_{i=1}^l \tilde{r} (u'_i + \prod_{j=1}^k m_{i,j} u_{i,j}) \quad \text{and} \quad S_2 \leftarrow w_2 + \tilde{r} P. \quad (3.4)$$

It is easy to see that  $\sigma = (S_1, S_2)$  is also a valid sequential aggregate signature on  $\text{M} \| M$  under keys  $\text{PK} \| \text{PK}$ , with randomness  $r + \tilde{r}$ , where  $w_2 = rP$ ; output  $\sigma$  and halt.

**WSA-AVerify**( $\sigma, \text{M}, \text{PK}$ ). The input is a purported sequential aggregate  $\sigma$  on messages  $\text{M}$  under public keys  $\text{PK}$ . Parse  $\sigma$  as  $(S_1, S_2) \in \mathbb{G}$ . If any key appears

twice in  $\mathbf{PK}$ , if any key in  $\mathbf{PK}$  has not been certified, or if  $|\mathbf{PK}| \neq |\mathbf{M}|$ , output `invalid` and `halt`.

Otherwise, set  $l \leftarrow |\mathbf{PK}|$ . If  $l = 0$  and  $S_1 = S_2 = 1$ , then output `valid`, and `invalid` otherwise.

Now, for each  $i$ ,  $1 \leq i \leq l$ , parse  $\mathbf{M}[i]$  as  $(m_{i,1}, \dots, m_{i,k}) \in \{0, 1\}^k$ , and parse  $\mathbf{PK}[i]$  as  $(A_i, u'_i, u_{i,1}, \dots, u_{i,k}) \in \mathbb{G}_T \times \mathbb{G}^{k+1}$ . Finally, verify that

$$e(S_1, P) \cdot e\left(S_2, \sum_{i=1}^l \left(u'_i + \sum_{j=1}^k m_{i,j} u_{i,j}\right)\right)^{-1} \stackrel{?}{=} \prod_{i=1}^l A_i \quad (3.5)$$

holds; if so, output `valid`; if not, output `invalid`.

**Signature Form.** Consider a sequential aggregate signature on  $l$  messages  $\mathbf{M}$  under  $l$  public keys  $\mathbf{PK}$ . For each  $i$  let  $\mathbf{M}[i]$  be  $(m_{i,1}, \dots, m_{i,k})$  and let  $\mathbf{PK}[i]$  be  $(A_i, u'_i, u_{i,1}, \dots, u_{i,k})$  with corresponding private key  $(\alpha_i, y'_i, y_{i,1}, \dots, y_{i,k})$ . A well-formed sequential aggregate signature  $\sigma = (S_1, S_2)$  in this case has the form

$$S_1 = \sum_{i=1}^l \alpha_i P + r \sum_{i=1}^l \left(u'_i + \sum_{j=1}^k m_{i,j} u_{i,j}\right) \quad \text{and} \quad S_2 = rP .$$

Additionally, we consider  $\sigma = (1, 1)$  to be a valid signature on an empty set of signers. Notice that  $(S_1, S_2)$  is the product of Waters signatures all sharing the same randomness  $r$ .

Even though in our description we did not allow a signer to sign twice in an aggregate signature, a simple trick allows for this. Suppose a signer wishes to add his signature on message  $M$  to a sequential aggregate signature that already contains his signature on another message  $M'$ . He need simply first remove his signature on  $M'$  from the aggregate, essentially by dividing it out of  $S_1$ , and multiply in a signature on  $M' : M$ , which is a message that attests to both  $M'$  and  $M$ .

**Performance.** Verification in our signatures is fast, taking approximately  $k/2$  multiplications per signer in the aggregate, and only two pairings regardless of how many signers are included. In contrast, the aggregate signatures of Boneh et al. [BGL03] take  $l + 1$  pairings to verify when the aggregate includes  $l$  signers.

### 3.2.3 Proof of Security

**Theorem 3.2.1.** *The sequential aggregate signature scheme  $\mathcal{WSA}$  is  $(t, q_C, q_S, n, \epsilon)$ -unforgeable if the underlying Waters signature scheme  $\mathcal{W}$  is  $(t', q', \epsilon')$ -unforgeable on  $\mathbb{G}$ , where*

$$t' = t + O(q_C + nq_S + n) \quad \text{and} \quad q' = q_S \quad \text{and} \quad \epsilon' = \epsilon .$$

*Proof.* Suppose that there exists an adversary  $\mathcal{A}$  that succeeds with advantage  $\epsilon$ . We build a simulator  $\mathcal{B}$  to play the forgeability game against the scheme  $\mathcal{W}$ . In the end, our simulator will attempt to output a forged Waters signature (W signature). Given the challenge  $\mathcal{W}$  public key  $\text{PK}^* = (A, u', u_1, \dots, u_k)$ , simulator  $\mathcal{B}$  interacts with  $\mathcal{A}$  as follows.

**Setup.** Algorithm  $\mathcal{B}$  runs  $\mathcal{A}$  supplying it with the challenge key  $\text{PK}^*$ .

**Certification Queries.** Algorithm  $\mathcal{A}$  wishes to certify some public key  $\text{PK} = (A, u', u_1, \dots, u_k)$ , providing also its corresponding private key  $\text{SK} = (\alpha, y', y_1, \dots, y_k)$ . Algorithm  $\mathcal{B}$  checks that the private key is indeed the correct one and if so registers  $(\text{PK}, \text{SK})$  in its list of certified keypairs.

**Aggregate Signature Queries.** Algorithm  $\mathcal{A}$  requests a sequential aggregate signature, under the challenge key, on a message  $M$ . In addition, it supplies an aggregate-so-far  $\sigma'$  on messages  $M$  under keys  $\text{PK}$ . The simulator first checks that the signature  $\sigma'$  verifies; that each key in  $\text{PK}$  has been certified; that the



challenge key does not appear in  $\mathbf{PK}$ ; and that  $|\mathbf{PK}| < n$ . If any of these conditions does not hold,  $\mathcal{B}$  returns `fail`.

Otherwise,  $\mathcal{B}$  queries its own signing oracle for key  $\mathbf{PK}^*$ , obtaining a signature  $\sigma$  on message  $M$ , which we view as a sequential aggregate on messages  $(M)$  under keys  $(\mathbf{PK}^*)$ . The simulator now constructs the rest of the required aggregate by adding to  $\sigma$ , for each signer  $\mathbf{PK}[i]$ , the appropriate signature on message  $\mathbf{M}[i]$  using algorithm `AggSign`. It can do this because it knows—by means of the certification procedure—the private key corresponding to each public key in  $\mathbf{PK}$ . The result is an aggregate signature  $\sigma'$  on messages  $\mathbf{M}||M$  under keys  $\mathbf{PK}||\mathbf{PK}^*$ . This reconstruction method works because signatures are re-randomized after each aggregate signing operation and because our signatures have no inherent verification order.

**Output.** Eventually,  $\mathcal{A}$  halts, outputting a forgery,  $\sigma^* = (S_1^*, S_2^*)$  on messages  $\mathbf{M}$  under keys  $\mathbf{PK}$ . This forgery must verify as valid under `AVerify`; each key in  $\mathbf{PK}$  (except the challenge key) must have been certified; and  $|\mathbf{PK}| \leq n$  must hold. In addition, the forgery must be nontrivial: the challenge key  $\mathbf{PK}^*$  must appear in  $\mathbf{PK}$ , `WLOG` at index 1 (since signature verification in our scheme has no inherent order), and the corresponding message  $\mathbf{M}[1]$  must not have been queried by  $\mathcal{A}$  of its sequential aggregate signing oracle. If the adversary was not successful we can quit and disregard the attempt.

Now, for each  $i$ ,  $1 \leq i \leq l = |\mathbf{PK}| = |\mathbf{M}|$ , parse  $\mathbf{PK}[i]$  as  $(A_i, u'_i, u_{i,1}, \dots, u_{i,k})$  and  $\mathbf{M}[i]$  as  $(m_{i,1}, \dots, m_{i,k}) \in \{0, 1\}^k$ . Note that we have  $\mathbf{PK}^* = (A_1, u'_1, u_{1,1}, \dots, u_{1,k})$ . Furthermore, for each  $i$ ,  $2 \leq i \leq l$ , let  $(\alpha_i, y'_i, y_{i,1}, \dots, y_{i,k})$  be the private key corresponding to  $\mathbf{PK}[i]$ . Algorithm  $\mathcal{B}$  computes

$$S_1 \leftarrow S_1^* - \sum_{i=2}^l \left( \alpha_i P + (y'_i + \sum_{j=1}^k y_{i,j} m_{i,j}) S_2^* \right) \quad \text{and} \quad S_2 \leftarrow S_2^* .$$

We now have

$$\begin{aligned}
& e(S_1, P) \cdot e\left(S_2, u'_1 + \sum_{j=1}^k m_{1,j} u_{1,j}\right)^{-1} \\
&= e(S_1^*, P) \cdot e\left(S_2^*, u'_1 + \sum_{j=1}^k m_{1,j} u_{1,j}\right)^{-1} \\
&\quad \cdot \prod_{i=2}^l e(\alpha_i P, P)^{-1} \cdot \prod_{i=2}^l e\left(y'_i + \sum_{j=1}^k y_{i,j} m_{i,j}\right) S_2^*, P)^{-1} \\
&= e(S_1^*, P) \cdot e\left(S_2^*, u'_1 + \sum_{j=1}^k m_{1,j} u_{1,j}\right)^{-1} \\
&\quad \cdot \prod_{i=2}^l A_i^{-1} \cdot \prod_{i=2}^l e\left(S_2^*, u'_i + \sum_{j=1}^k m_{i,j} u_{i,j}\right)^{-1} \\
&= e(S_1^*, P) \cdot \prod_{i=1}^l e\left(S_2^*, u'_i + \sum_{j=1}^k m_{i,j} u_{i,j}\right)^{-1} \cdot \prod_{i=2}^l A_i^{-1} \\
&= \prod_{i=1}^l A_i \cdot \prod_{i=2}^l A_i^{-1} = A_1 = A .
\end{aligned}$$

The last line follows from the sequential aggregate verification equation. So  $(S_1, S_2)$  is a valid signature on  $M^* = \mathbf{M}[1] = (m_{1,1}, \dots, m_{1,k})$  under key  $\mathbf{PK}[1] = \mathbf{PK}^*$ . Moreover, since  $\mathcal{A}$  did not make an aggregate signing query at  $M^*$ ,  $\mathcal{B}$  did not make a signing query at  $M^*$ , so  $\sigma = (S_1, S_2)$  is a nontrivial  $\mathbf{W}$  signature forgery. Algorithm  $\mathcal{B}$  returns  $\sigma$  and halts.

Algorithm  $\mathcal{B}$  is successful whenever  $\mathcal{A}$  is. Algorithm  $\mathcal{B}$  makes as many signing queries as  $\mathcal{A}$  makes sequential aggregate signing queries. Algorithm  $\mathcal{B}$ 's running time is that of  $\mathcal{A}$ , plus the overhead in handling  $\mathcal{A}$ 's queries, and computing the final result. Each certification query can be handled in  $O(1)$  time; each aggregate signing query can be handled in  $O(n)$  time; and the final result can also be computed from  $\mathcal{A}$ 's forgery in  $O(n)$  time.  $\square$

### 3.2.4 A More Efficient Variant in the Random Oracle Model

Our scheme as described in Section 3.2.2 implicitly uses the hash  $H(m_1, \dots, m_k) = u' + \sum_{i=1}^k m_i u_i$ . It is also possible to instantiate it with the Boneh-Boyen hash  $H(M) = u' + H_0(M)u$ , where  $H_0$  maps  $\{0, 1\}^*$  to  $\mathbb{Z}/p\mathbb{Z}$  and is treated as a random oracle. (This derives from Boneh and Boyen’s suggested conversion, in the random oracle model, of their selective-ID IBE to a fully secure one [BB04a, Theorem 7.2], to which we then apply the Naor transform recorded by Boneh and Franklin [BF01, BF03] to obtain a signature.)

In this variant, each user picks  $x, y, \alpha \xleftarrow{R} \mathbb{Z}/p\mathbb{Z}$  and publishes  $u = xP, u' = yP$ , and  $A = e(P, P)^\alpha$ . Public key sizes are thus much smaller than in our Waters-based scheme.

Compared to the scheme of Boneh et al. [BGL03], whose proof of security is also in the random oracle model, our variant scheme is sequential, uses a weaker (certified-key) security model, and has somewhat longer public keys and signatures. On the other hand, verification in our variant scheme requires only a constant number of pairings rather than  $l + 1$  for an  $l$ -user aggregate as in BGLS.

## 3.3 Multisignatures

In a multisignature scheme, a single multisignature—the same size as one ordinary signature—stands for  $l$  signatures on a message  $M$ . Multisignatures were introduced by Itakura and Nakamura [IN83], and have been the subject of much research [Oka88, OO99, Bol03]. The first multisignatures in which signatures could be combined into a multisignature without interaction was proposed by Boldyreva [Bol03], based on BLS signatures [BLS04]. Below, we present another non-interactive multisignature scheme, based on the Waters signature, which is provably secure without random oracles.

**Security Model.** Micali, Ohta, and Reyzin [MOR01] gave the first formal treatment of multisignatures. We prove security in a variant of the Micali-Ohta-Reyzin model due to Boldyreva [Bol03]. In this model, the adversary is given a single challenge public key  $PK$ , and a signing oracle for that key. His goal is to output a forged multisignature  $\sigma^*$  on a message  $M^*$  under keys  $PK_1, \dots, PK_l$ . Of these keys,  $PK_1$  must be the challenge key  $PK$ . For the forgery to be nontrivial, the adversary must not have queried the signing oracle at  $M^*$ . The adversary is allowed to choose the remaining keys, but must prove knowledge of the private keys corresponding to them. For simplicity, Boldyreva handles this by having the adversary hand over the private keys; in a more complicated proof of knowledge, the keys could be extracted by rewinding, with the same result.

### 3.3.1 The WM Multisignature Scheme

We describe the multisignature obtained from the Waters signature. In this scheme, all users share the same random generators  $u', u_1, \dots, u_k$ , which are included in the system parameters. Our scheme is a five-tuple of algorithms  $\mathcal{WM} = (\text{KeyGen}, \text{Sign}, \text{Verify}, \text{Combine}, \text{MVerify})$ , which behave as follows.

**WM-KeyGen, WM-Sign, WM-Verify.** These are the same as **W-KeyGen, W-Sign,** and **W-Verify**, respectively.

**WM-Combine**( $\{PK_i, \sigma_i\}_{i=1}^l, M$ ). For each user in the multisignature the algorithm takes as input a public key  $PK_i$  and a signature  $\sigma_i$ . All of these signatures are on a single message  $M$ . For each  $i$ , parse user  $i$ 's public key  $PK_i$  as  $A_i \in \mathbb{G}_T$  and her signature  $\sigma_i$  as  $(S_1^{(i)}, S_2^{(i)}) \in \mathbb{G}^2$ ; parse the message  $M$  as a bitstring  $(m_1, \dots, m_k) \in \{0, 1\}^k$ . Verify each signature using **Verify**; if any is invalid,

output `fail` and halt. Otherwise, compute

$$S_1 \leftarrow \sum_{i=1}^l S_1^{(i)} \quad \text{and} \quad S_2 \leftarrow \sum_{i=1}^l S_2^{(i)} . \quad (3.6)$$

The multisignature is  $\sigma = (S_1, S_2)$ ; output it and halt.

**WM-MVerify**( $\{\mathbf{PK}_i\}_{i=1}^l, M, \sigma$ ). For each user in the multisignature, the algorithm takes a public key  $\mathbf{PK}_i$ . The algorithm also takes a purported multisignature  $\sigma$  on a message  $M$ . Parse user  $i$ 's public key  $\mathbf{PK}_i$  as  $A_i \in \mathbb{G}_T$ , the message  $M$  as a bitstring  $(m_1, \dots, m_k) \in \{0, 1\}^k$ , and the multisignature  $\sigma$  as  $(S_1, S_2) \in \mathbb{G}^2$ . Verify that

$$e(S_1, P) \cdot e(S_2, u' + \sum_{i=1}^k m_i u_i)^{-1} \stackrel{?}{=} \prod_{i=1}^l A_i^{(i)} \quad (3.7)$$

holds; if so, output `valid`; if not, output `invalid`.

It is clear that if all signatures verify individually, the multisignature formed by their product also verifies according to (3.7). Note that we have

$$(S_1, S_2) = \left( \left( \sum_{i=1}^l \alpha^{(i)} \right) P + \left( \sum_{i=1}^l r^{(i)} \right) \left( u' + \sum_{j=1}^k m_j u_j \right), \left( \sum_{i=1}^l r^{(i)} \right) P \right) ,$$

where  $r^{(i)}$  is the randomness used by User  $i$  to generate her signature.

### 3.3.2 Proof of Security

The  $\mathcal{WM}$  scheme is unforgeable if  $W$  signatures are unforgeable. We state this as the following theorem.

**Theorem 3.3.1.** *The  $\mathcal{WM}$  multisignature scheme is  $(t, q, \epsilon)$ -unforgeable if the  $\mathcal{W}$  signature scheme is  $(t', q', \epsilon')$ -unforgeable, where*

$$t' = t + O(q) \quad \text{and} \quad q' = q \quad \text{and} \quad \epsilon' = \epsilon .$$

*Proof.* Suppose  $\mathcal{A}$  is an adversary that can forge multisignatures, and  $(t, q, \epsilon)$ -breaks the WM scheme. We show how to construct an algorithm  $\mathcal{B}$  that  $(t', q, \epsilon)$ -breaks the W scheme. Algorithm  $\mathcal{B}$  is given a W public key  $A = e(P, P)^\alpha$ . It interacts with  $\mathcal{A}$  as follows.

**Setup.** Simulator  $\mathcal{B}$  invokes  $\mathcal{A}$ , providing to it the public key  $A$ .

**Signature queries.** Algorithm  $\mathcal{A}$  requests a signature on some message  $M$  under the challenge key  $A$ . Algorithm  $\mathcal{B}$  requests a signature on  $M$  in turn from its own signing oracle, and returns the result to the adversary.

**Output.** Finally,  $\mathcal{A}$  halts, having output a signature  $(S_1^*, S_2^*)$  on some message  $M^*$ , along with public keys  $A^{(1)}, \dots, A^{(l)}$  for some  $l$ , where  $A^{(1)}$  equals  $A$ , the challenge key. It must not previously have requested a signature on  $M^*$ . In addition, it outputs the private keys  $\alpha^{(2)}, \dots, \alpha^{(l)}$  for all keys except the challenge key. Algorithm  $\mathcal{B}$  sets  $S \leftarrow S_1^* - \sum_{i=2}^l \alpha^{(i)} P$ . Then we have

$$\begin{aligned} e(S, P) \cdot e(S_2, u' + \sum_{i=1}^k m_i u_i)^{-1} &= e(S_1, P) \cdot e(S_2, u' + \sum_{i=1}^k m_i u_i)^{-1} \\ &\quad \cdot \prod_{i=2}^l e(P, P)^{-\alpha^{(i)}} \\ &= \prod_{i=1}^l A^{(i)} \cdot \prod_{i=2}^l A^{-\alpha^{(i)}} = A^{(1)} = A, \end{aligned}$$

so  $(S, S_2)$  is a valid W signature on  $M^*$  under the challenge key  $A$ . Since  $\mathcal{A}$  did not make a signing query to the challenger at  $M^*$ , neither did  $\mathcal{B}$  make a signing query to its own signing oracle at  $M^*$ , and the forgery is thus nontrivial. Algorithm  $\mathcal{B}$  outputs  $(S, S_2)$  and halts.

Thus  $\mathcal{B}$  succeeds whenever  $\mathcal{A}$  does. Algorithm  $\mathcal{B}$  makes exactly as many signing queries as  $\mathcal{A}$  does. Its running time is the same as  $\mathcal{A}$ 's, plus the time required for setup

and output—both  $O(1)$ —and to handle  $\mathcal{A}$ 's signing queries— $O(1)$  for each of at most  $q$  queries.  $\square$

### 3.4 Verifiably Encrypted Signatures

A verifiably encrypted signature on some message attests to two facts:

- that the signer has produced an ordinary signature on that message; and
- that the ordinary signature can be recovered by the third party under whose key the signature is encrypted.

Such a primitive is useful for contract signing, in a protocol called optimistic fair exchange [ASW00, BDM98]. Suppose both Alice and Bob wish to sign some contract. Neither is willing to produce a signature without being sure that the other will. But Alice can send Bob a verifiably encrypted signature on the contract. Bob can now send Alice his signature, knowing that if Alice does not respond with hers he can take Alice's verifiably encrypted signature and the transcript of his interaction with Alice to the third party—called the adjudicator—who will reveal Alice's signature.

Boneh et al. [BGL03] introduced verifiably encrypted signatures, gave a security model for them, and constructed a scheme satisfying the definitions, based on the BLS short signature [BLS04].

We describe the verifiably encrypted signature scheme obtained from the Waters signature scheme. Unlike the scheme of Boneh et al., ours is secure without random oracles.

**Security Model.** Boneh et al. specify two properties (besides correctness) that a verifiably encrypted signature scheme must satisfy: unforgeability and opacity. Both

are defined in games. In each, the adversary is given a signer's public key PK and an adjudicator's public key APK. He is allowed to make verifiably encrypted signing queries of the form  $\text{EncSign}(\text{SK}, \text{APK}, \cdot)$  and adjudication queries of the form  $\text{Adj}(\text{ASK}, \text{PK}, \cdot, \cdot)$ . In the unforgeability game, his goal is to output  $(M^*, \eta^*)$  such that he did not query his signing oracle at  $M^*$ ; in the opacity game his goal is to output  $(M^*, \sigma^*)$  such that he did not query his *adjudication* oracle at  $M^*$ . An adversary can thus win the opacity game either by creating a forgery for the underlying signature scheme directly or by recovering the ordinary signature from an encrypted signature without the adjudicator's help.

### 3.4.1 Our Scheme

Our scheme is a seven-tuple of algorithms  $\mathcal{WVES} = (\text{KeyGen}, \text{Sign}, \text{Verify}, \text{AdjKg}, \text{EncSign}, \text{EncVerify}, \text{Adj})$  that behave as follows.

**WVES-KeyGen, WVES-Sign, WVES-Verify** These are the same as W-KeyGen, W-Sign, and W-Verify, respectively.

**WVES-AdjKg.** Pick  $\beta \xleftarrow{R} \mathbb{Z}/p\mathbb{Z}$ , and set  $v \leftarrow \beta P$ . The adjudicator's public key is  $\text{APK} = v$ ; the adjudicator's private key is  $\text{ASK} = \beta$ .

**WVES-EncSign(SK, APK, M)** Parse the user's private key SK as  $\alpha \in \mathbb{Z}/p\mathbb{Z}$  and the adjudicator's public key APK as  $v \in \mathbb{G}$ . To sign the message  $M = (m_1, \dots, m_k)$ , compute a signature  $(S_1, S_2) \leftarrow \text{Sign}(\text{SK}, M)$ . Pick a random  $s \xleftarrow{R} \mathbb{Z}/p\mathbb{Z}$ , and compute

$$K_1 \leftarrow S_1 + sv \quad \text{and} \quad K_2 \leftarrow S_2 \quad \text{and} \quad K_3 \leftarrow sP .$$

The verifiably encrypted signature  $\eta$  is the tuple  $(K_1, K_2, K_3)$ .

**WVES-EncVerify(PK, APK, M,  $\eta$ ).** Parse the user's public key PK as  $A \in \mathbb{G}_T$ , the



adjudicator's public key APK as  $v \in \mathbb{G}$ , and the verifiably encrypted signature  $\eta$  as  $(K_1, K_2, K_3) \in \mathbb{G}^3$ . Accept if the following equation holds:

$$e(K_1, P) \cdot e(K_2, u' + \sum_{i=1}^k m_i u_i)^{-1} \cdot e(K_3, v)^{-1} \stackrel{?}{=} A, \quad (3.8)$$

where  $M = (m_1, \dots, m_k)$ .

**WVES-Adj(ASK, PK, M,  $\eta$ ).** Parse the adjudicator's private key ASK as  $\beta \in \mathbb{Z}/p\mathbb{Z}$ .

Parse the user's public key PK as  $A \in \mathbb{G}_T$ , and check that it has been certified.

Parse the message  $M$  as  $(m_1, \dots, m_k) \in \{0, 1\}^k$ . Verify (using **EncVerify**) that the verifiably encrypted signature  $\eta$  is valid, and parse it as  $(K_1, K_2, K_3) \in \mathbb{G}^3$ .

Compute

$$S_1 \leftarrow K_1 - \beta K_3 \quad \text{and} \quad S_2 \leftarrow K_2;$$

re-randomize  $(S_1, S_2)$  by choosing  $s \xleftarrow{R} \mathbb{Z}/p\mathbb{Z}$  and computing

$$S'_1 \leftarrow S_1 + s \left( u' + \sum_{i=1}^k m_i u_i \right) \quad \text{and} \quad S'_2 \leftarrow S_2 + sP;$$

and output the signature  $(S'_1, S'_2)$ .

It is easy to see that this scheme is valid, since if all parties are honest we have, for a verifiably encrypted signature  $(K_1, K_2, K_3)$ ,

$$\begin{aligned} & e(K_1, P) \cdot e(K_2, u' + \sum_{i=1}^k m_i u_i)^{-1} \cdot e(K_3, v)^{-1} \\ &= (e(S_1, P) \cdot e(sv, P)) \cdot e(S_2, u' + \sum_{i=1}^k m_i u_i)^{-1} \cdot e(sP, v)^{-1} \\ &= e(S_1, P) \cdot e(S_2, u' + \sum_{i=1}^k m_i u_i)^{-1} = A, \end{aligned}$$

as required; and if  $(K_1, K_2, K_3)$  is a valid verifiably encrypted signature then

$$\begin{aligned}
& e(S_1, P) \cdot e(S_2, u' + \sum_{i=1}^k m_i u_i)^{-1} \\
&= (e(K_1, P) \cdot e(-\beta K_3, P)) \cdot e(K_2, u' + \sum_{i=1}^k m_i u_i)^{-1} \\
&= e(K_1, P) \cdot e(K_2, u' + \sum_{i=1}^k m_i u_i)^{-1} \cdot e(K_3, v)^{-1} = A,
\end{aligned}$$

so the adjudicated signature is indeed a valid one.

**Proofs of Security.** The WVES scheme is unforgeable if W signatures are unforgeable, and opaque if CDH is hard on  $\mathbb{G}$ . We give the proofs in Section 3.5.

### 3.4.2 VES from General Assumptions

Recent work has shown that group signatures [BMW03] and ring signatures [BKM06] can be built from general assumptions using Non-Interactive Zero Knowledge (NIZK) proofs. We note that verifiably encrypted signatures can also be realized from general assumptions. Roughly, the signer signs a message, encrypts the signature to the adjudicator and then attaches a NIZK proof that this was performed correctly.

Specifically, we show how to construct a VES scheme from secure generic signature schemes, encryption schemes, and adaptive unbounded NIZKs (in the common reference string model). Let these be denoted by the following tuples of algorithms —  $(\text{SKg}, \text{Sign}, \text{Verify})$ ,  $(\text{EKg}, \text{Enc}, \text{Dec})$ ,  $(\text{Pr}, \text{V}, \text{Sim})$ , respectively. From these we construct a verifiably encrypted signature scheme  $\mathcal{GVES} = (\text{KeyGen}, \text{Sign}, \text{Verify}, \text{AdjKg}, \text{EncSign}, \text{EncVerify}, \text{Adj})$  as follows.

**GVES-KeyGen, GVES-Sign, GVES-Verify, GVES-AdjKg** These algorithms are the same as SKg, Sign, and Verify, EKg respectively.

**GVES-EncSign**(SK, APK,  $M$ ) To sign the message  $M$ , we compute

$\sigma \leftarrow \text{Sign}(\text{SK}, M)$ . Then compute  $\gamma \leftarrow \text{Enc}(\text{APK}, \sigma)$  with randomness  $r$ , and provides a NIZK proof of the statement “There exists  $\sigma, r$  such that  $\gamma = \text{Enc}(\text{APK}, \sigma; r)$  and  $\text{Verify}(\text{PK}, M, \sigma) = 1$ ”. The verifiably encrypted signature  $\eta$  is the tuple  $(\gamma, \pi)$ .

**GVES-EncVerify**(PK, APK,  $M, \eta$ ). Parse the verifiably encrypted signature  $\eta$  as  $(\gamma, \pi)$ . Accept if  $\text{V}(M, \text{PK}, \text{APK}, \gamma, \pi) = 1$ .

**GVES-Adj**(ASK, PK,  $M, \eta$ ). Verify (using **EncVerify**) that the verifiably encrypted signature  $\eta$  is valid, and parse it as  $(\gamma, \pi)$  Compute  $\sigma \leftarrow \text{Dec}(\text{ASK}, \gamma)$  and output the signature  $\sigma$ .

**Theorem 3.4.1.** *The GVES verifiably encrypted signature scheme is unforgeable if the underlying signature scheme is unforgeable and the underlying NIZK scheme is adaptively unbounded.*

*Proof.* We show how to turn a verifiably-encrypted signature forger  $\mathcal{A}$  into a forger  $\mathcal{B}$  for the underlying signature scheme.

Algorithm  $\mathcal{B}$  is given a signature public key PK. It sets up its own encryption scheme and computes  $(\text{APK}, \text{ASK}) \leftarrow \text{AdjKg}$ , and provides the adversary  $\mathcal{A}$  with PK and APK.

When  $\mathcal{A}$  requests a verifiably encrypted signature on some message  $M$ , the challenger  $\mathcal{B}$  requests a signature on  $M$  from the signature scheme, obtaining a signature  $\sigma$ . It computes  $\eta = (\text{Enc}(\sigma), \pi)$  from its own encryption and NIZK oracle. Algorithm  $\mathcal{B}$  provides  $\mathcal{A}$  with it.

When algorithm  $\mathcal{A}$  requests adjudication of a verifiably encrypted signature  $(\gamma, \pi)$  on some message  $M$ ,  $\mathcal{B}$  verifies the NIZK and runs its decryption oracle on  $\gamma$  and replies with the decrypted value.

Finally,  $\mathcal{A}$  outputs a forged verifiably-encrypted signature  $\eta^* = (\gamma^*, \pi^*)$  on some message  $M^*$ . Algorithm  $\mathcal{A}$  must never have made a verifiably encrypted signing query at  $M^*$ . Furthermore, by the soundness of the NIZK,  $\gamma^*$  must be a valid encryption of some valid signature,  $\sigma^*$ . The challenger  $\mathcal{B}$  outputs  $\sigma^*$  and halts.

Algorithm  $\mathcal{B}$  thus succeeds whenever  $\mathcal{A}$  does and the soundness of the NIZK is preserved. Its running time overhead is  $O(1)$  for each of  $\mathcal{A}$ 's verifiably encrypted signing and adjudication queries, and for computing the final output.  $\square$

To prove that GVES is opaque, we create a series of hybrid experiments to show any adversary that can break GVES can break the security of at least one of the underlying schemes.

**Theorem 3.4.2.** *The GVES verifiably encrypted signature scheme is opaque if the underlying signature scheme is unforgeable, the underlying encryption scheme is CCA2-secure, and the underlying NIZK scheme is adaptively unbounded.*

*Proof.* We consider the following hybrid experiments:

**Experiment 0:** This will be the actual opacity experiment. Our goal is to show the adversary succeeds with negligible probability in this experiment.

**Experiment 1:** Experiment 1 succeeds if Experiment 0 succeeds and the adversary did not generate a proof of any false statement.

**Experiment 2:** Experiment 2 succeeds if Experiment 1 succeeds and the adversary queried for the verifiably encrypted signature of its output message  $M^*$ .

**Experiment 3:** In this experiment, we modify Experiment 2 by replacing all proofs provided to the adversary by simulated proofs.

**Experiment 4:** We change Experiment 3 as follows. For each verifiably encrypted signature query, with probability  $\lambda$  we provide the adversary with  $(\text{Enc}(\perp, r), \text{Sim}(\perp, r))$  as the verifiably encrypted signature. This experiment succeeds if the previous conditions hold and event  $\mathcal{E}$  occurs, where  $\mathcal{E}$  is the event that the adversary received real signatures on everything except  $M^*$  and a fake signature on  $M^*$ .

Let  $\mathcal{A}$  be an adversary that succeeds in the opacity game for verifiably encrypted signatures. We define the advantage  $\text{Adv}_{\mathcal{A}}^{\text{Expt}_i}$  as the probability it succeeds in experiment  $i$ . Define the differences in the advantages as  $\epsilon_i = |\text{Adv}_{\mathcal{A}}^{\text{Expt}_{i+1}} - \text{Adv}_{\mathcal{A}}^{\text{Expt}_i}|$ . We now show that each of these differences must be negligible.

$\epsilon_0$ : This difference between these two experiments is the probability the adversary can (when used as a subroutine) generate a proof of a false statement. This must be negligible by the soundness of the NIZK.

$\epsilon_1$ : This difference between these two experiments is the probability the adversary can (when used as a subroutine) forge a valid signature on  $M^*$ . This must be negligible by the existential unforgeability of the underlying signature scheme.

$\epsilon_2$ : This difference between these two experiments is the probability the adversary can (when used as a subroutine) distinguish between the actual prover and the simulator. This must be negligible by the adaptive unbounded zero-knowledge of the NIZK.

$\epsilon_3$ : First notice if  $q_s$  is the number of signing queries that  $\mathcal{A}$  makes, we may choose  $\lambda$  to be  $1 - 1/q_s$  to make  $\mathcal{E}$  occur with probability greater than  $1/((e)(q_s - 1))$ , where  $e$  is the base of the natural logarithm. Thus we can use this adversary as a subroutine to create an algorithm that plays the CCA2 game against the encryption scheme. We make the obvious construction by replacing our own

decryption algorithm with decryption queries to the challenger of the CCA2 game and choosing  $\sigma^*, \perp$  as our challenge messages when the time comes. This constructed algorithm will have a  $\epsilon_3 / ((e)(q_S - 1))$  advantage in the game. Thus  $\epsilon_3$  must be negligible.

Finally we notice that the output  $(M^*, \sigma^*)$  of Experiment 4 is actually an existential forgery on  $M^*$  as we have never queried the signing oracle on it! Thus  $\text{Adv}_{\mathcal{A}}^{\text{Expt}_4}$  is negligible, and hence so is  $\text{Adv}_{\mathcal{A}}^{\text{Expt}_0}$ .  $\square$

## 3.5 WVES Proofs of Security

### 3.5.1 Unforgeability

**Theorem 3.5.1.** *The WVES verifiably encrypted signature scheme is  $(t, q_S, q_A, \epsilon)$ -unforgeable if the W signature scheme is  $(t', q', \epsilon')$ -unforgeable, where*

$$t' = t + O(q_S + q_A) \quad \text{and} \quad q' = q_S \quad \text{and} \quad \epsilon' = \epsilon .$$

*Proof.* We show how to turn a verifiably-encrypted signature forger  $\mathcal{A}$  into a forger  $\mathcal{B}$  for the underlying Waters signature scheme.

Algorithm  $\mathcal{B}$  is given a Waters signature public key  $A = e(P, P)^\alpha$ . It picks  $\beta \xleftarrow{R} \mathbb{Z}/p\mathbb{Z}$ , sets  $v \leftarrow \beta P$ , and provides the adversary  $\mathcal{A}$  with  $A$  and  $v$ .

When  $\mathcal{A}$  requests a verifiably encrypted signature on some message  $M$ , the challenger  $\mathcal{B}$  requests a signature on  $M$  from its own signing oracle, obtaining a signature  $(S_1, S_2)$ . It picks  $s \xleftarrow{R} \mathbb{Z}/p\mathbb{Z}$  and computes

$$K_1 \leftarrow S_1 \cdot sv \quad \text{and} \quad K_2 \leftarrow S_2 \quad \text{and} \quad K_3 \leftarrow sP .$$

The tuple  $(K_1, K_2, K_3)$  is a valid verifiably encrypted signature on  $M$ . Algorithm  $\mathcal{B}$  provides  $\mathcal{A}$  with it. (Here  $\mathcal{B}$  is simply evaluating  $\text{EncSign}$ , except that it uses its

signing oracle instead of evaluating  $\text{Sign}$  directly.)

When algorithm  $\mathcal{A}$  requests adjudication of a verifiably encrypted signature  $(K_1, K_2, K_3)$  on message  $M$  under the challenge key  $A$ ,  $\mathcal{B}$  responds with  $\text{Adj}(\beta, A, M, (K_1, K_2, K_3))$ . Note that  $\mathcal{B}$  knows the adjudicator's private key  $\beta$ .

Finally,  $\mathcal{A}$  outputs a forged verifiably-encrypted signature  $(K_1^*, K_2^*, K_3^*)$  on some message  $M^* = (m_1^*, \dots, m_k^*)$ . Algorithm  $\mathcal{A}$  must never have made a verifiably encrypted signing query at  $M^*$ .

The challenger  $\mathcal{B}$  computes

$$S_1^* \leftarrow K_1^* - \beta K_3^* \quad \text{and} \quad S_2^* \leftarrow K_2^*.$$

Then we have

$$\begin{aligned} & e(S_1^*, P) \cdot e(S_2^*, u' + \sum_{i=1}^k m_i^* u_i)^{-1} \\ &= \left[ e(K_1^*, P) \cdot e(K_2^*, u' + \sum_{i=1}^k m_i^* u_i)^{-1} \right] \cdot e(-\beta K_3^*, P) \\ &= e(K_1^*, P) \cdot e(K_2^*, u' + \sum_{i=1}^k m_i^* u_i)^{-1} \cdot e(K_3^*, v)^{-1} = A, \end{aligned}$$

and  $(S_1^*, S_2^*)$  is therefore a valid Waters signature on  $M^*$ . The last equality follows from equation (3.8). Because  $\mathcal{A}$  did not make a verifiably encrypted signing query at  $M^*$ , neither did  $\mathcal{B}$  make a signing query at  $M^*$ , and the forgery is thus nontrivial. The challenger  $\mathcal{B}$  outputs  $(S_1^*, S_2^*)$  and halts.

Algorithm  $\mathcal{B}$  thus succeeds whenever  $\mathcal{A}$  does. Its running time overhead is  $O(1)$  for each of  $\mathcal{A}$ 's verifiably encrypted signing and adjudication queries, and for computing the final output.  $\square$

### 3.5.2 Opacity

For convenience, we prove opacity by reduction from the aggregate extraction assumption: given  $(\alpha P, \beta P, \gamma P, \delta P, (\alpha\gamma + \beta\delta)P)$ , computing  $\alpha\gamma P$  is hard. Coron and Naccache [CN03] showed that this assumption, introduced by Boneh et al. [BGL03], is equivalent to the Computational Diffie-Hellman problem.

**Theorem 3.5.2** (Coron–Naccache [CN03]). *The aggregate extraction and Computational Diffie-Hellman problems are Karp reducible to each other with  $O(1)$  computation.<sup>2</sup>*

**Theorem 3.5.3.** *The  $\mathcal{WVES}$  scheme is  $(t, q_S, q_A, \epsilon)$ -opaque if aggregate extraction is  $(t', \epsilon')$ -hard on  $\mathbb{G}$ , where*

$$t' = t + O(q_S + q_A) \quad \text{and} \quad q' = q_S \quad \text{and} \quad \epsilon' = 4kq_A\epsilon .$$

*Proof.* Given an algorithm  $\mathcal{A}$  that breaks the opacity of the scheme, we show how to construct an algorithm  $\mathcal{B}$  that breaks the aggregate extraction assumption.

The challenger  $\mathcal{B}$  is given values  $\alpha P, \beta P, \gamma P$ , and  $\delta P$ , along with  $(\alpha\gamma + \beta\delta)P$ ; its goal is to produce  $\alpha\gamma P$ . It sets  $v \leftarrow \beta P, P_1 \leftarrow \alpha P$ , and  $P_2 \leftarrow \gamma P$ . It computes  $A \leftarrow e(P_1, P_2) = e(P, P)^{\alpha\gamma}$ .

Let  $\lambda = 2q_A$ . Algorithm  $\mathcal{B}$  picks  $\kappa \xleftarrow{R} \{0, \dots, k\}, x', x_1, \dots, x_k \xleftarrow{R} \{0, \dots, \lambda - 1\}$  and  $y', y_1, \dots, y_k \xleftarrow{R} \mathbb{Z}/p\mathbb{Z}$  and sets

$$u' \leftarrow (x' - \kappa\lambda)P_2 + y'P \quad \text{and} \quad u_i \leftarrow x_i P_2 + y_i P \quad \text{for } i = 1, \dots, k .$$

It then interacts with  $\mathcal{A}$  as follows.

---

<sup>2</sup>Strictly speaking, the amount of work is poly-logarithmic in the security parameter since the group element representations grow. The number of algebraic operations is constant.



**Setup.** Algorithm  $\mathcal{B}$  gives to  $\mathcal{A}$  the system parameters  $(P, u', u_1, \dots, u_k)$ , the signer's public key  $A$ , and the adjudicator's public key  $v$ . Note that the private signing key is  $\alpha\gamma$ .

**Verifiably Encrypted Signing Queries.**  $\mathcal{A}$  requests a verifiably-encrypted signature on  $M = (m_1, \dots, m_k) \in \{0, 1\}^k$  under challenge key  $A$  and adjudicator key  $v$ . Define  $F = -\kappa\lambda + x' + \sum_{i=1}^k x_i m_i$  and  $J = y' + \sum_{i=1}^k y_i m_i$ . If  $F \neq 0 \pmod p$  algorithm  $\mathcal{B}$  proceeds as follows. It picks  $r \xleftarrow{R} \mathbb{Z}/p\mathbb{Z}$  and sets

$$S_1 \leftarrow (-J/F)P_1 + r\left(u' + \sum_{i=1}^k m_i u_i\right) \quad \text{and} \quad S_2 \leftarrow (-1/F)P_1 + rP .$$

This is a valid W signature with randomness  $\tilde{r} = r - \alpha/F$ : observing that  $u' + \sum_{i=1}^k m_i u_i = F \cdot P_2 + J \cdot P$ , we see that

$$\begin{aligned} S_1 &= (-J/F)P_1 + r\left(u' + \sum_{i=1}^k m_i u_i\right) \\ &= \alpha P_2 - (-\alpha/F)(F \cdot P_2 + J \cdot P) + r(F \cdot P_2 + J \cdot P) \\ &= \alpha\gamma P + \tilde{r}\left(u' + \sum_{i=1}^k m_i u_i\right) , \end{aligned}$$

where for the second equality we have multiplied and divided by  $\alpha P_2$ . Algorithm  $\mathcal{B}$  then encrypts  $(S_1, S_2)$  by choosing  $s \xleftarrow{R} \mathbb{Z}/p\mathbb{Z}$  and setting

$$K_1 \leftarrow S_1 + sv \quad \text{and} \quad K_2 \leftarrow S_2 \quad \text{and} \quad K_3 \leftarrow sP .$$

If  $F = 0$ , however,  $\mathcal{B}$  picks  $r, s \xleftarrow{R} \mathbb{Z}/p\mathbb{Z}$  and sets

$$\begin{aligned} K_1 &\leftarrow (\alpha\gamma + \gamma\delta)P + s\gamma P + r\left(u' + \sum_{i=1}^k m_i u_i\right) \quad \text{and} \\ K_2 &\leftarrow rP \quad \text{and} \quad K_3 \leftarrow (\delta P) + sP . \end{aligned}$$

This is a WVES signature with randomness  $r$ , encrypted with randomness  $\delta + s$ . In either case,  $\mathcal{B}$  returns to  $\mathcal{A}$  the verifiably encrypted signature  $(K_1, K_2, K_3)$ .

**Adjudication Queries.** Suppose  $\mathcal{A}$  requests adjudication on  $(K_1, K_2, K_3)$  for message  $M = (m_1, \dots, m_k)$ . Algorithm  $\mathcal{B}$  first verifies that  $(K_1, K_2, K_3)$  is valid and rejects it otherwise. Define  $F = -\kappa\lambda + x' + \sum_{i=1}^k x_i m_i$  and  $J = y' + \sum_{i=1}^k y_i m_i$  as before. If  $F = 0 \pmod p$ ,  $\mathcal{B}$  declares failure and halts. Otherwise, it picks  $r \xleftarrow{R} \mathbb{Z}/p\mathbb{Z}$  and computes

$$S_1 \leftarrow (-J/F)P_1 + r\left(u' + \sum_{i=1}^k m_i u_i\right) \quad \text{and} \quad S_2 \leftarrow (-1/F)P_1 + rP$$

as above, returning  $(S_1, S_2)$  to  $\mathcal{A}$ .

(Note that  $\mathcal{A}$  must previously have made a verifiably encrypted signing query at  $M$ , since otherwise we could use it to break the unforgeability of WVES.)

**Output.** Finally, algorithm  $\mathcal{A}$  outputs a signature  $(S_1^*, S_2^*)$  on a message  $M^* = (m_1^*, \dots, m_k^*)$ ; it must not have queried its adjudication oracle at  $M^*$ . Define  $F^* = -\kappa\lambda + x' + \sum_{i=1}^k x_i m_i^*$  and  $J^* = y' + \sum_{i=1}^k y_i m_i^*$ . If  $F^* \neq 0 \pmod p$ ,  $\mathcal{B}$  declares failure and exits. Otherwise, we have  $u' + \sum_{i=1}^k m_i^* u_i = J^* \cdot P$ , so that

$$\begin{aligned} e(P_1, P_2) &= A = e(S_1^*, P) \cdot e\left(S_2^*, u' + \sum_{i=1}^k m_i^* u_i\right)^{-1} \\ &= e(S_1^*, P) \cdot e(S_2^*, J^* \cdot P)^{-1} = e(S_1^* - (J^*)S_2^*, P) \end{aligned}$$

and  $S_1^* - (J^*)S_2^*$  equals  $\alpha\gamma P$ , which is the solution to the aggregate extraction challenge;  $\mathcal{B}$  outputs this solution and halts.

The probability that  $\mathcal{B}$  does not abort in any adjudication query is at least  $1 - 1/\lambda$ ; since there are at most  $q_A = \lambda/2$  such queries,  $\mathcal{B}$  manages to answer all queries without aborting with probability at least  $1/2$ . Having done so,  $\mathcal{B}$  then receives a forgery such that  $F^* = 0 \pmod p$  with probability at least  $1/(\kappa\lambda) \geq 1/(2kq_A)$ . Thus  $\mathcal{B}$  succeeds with probability at least  $\epsilon/(4kq_A)$ . (For more detailed probability analysis, see Waters' original proof [Wat05].) Algorithm  $\mathcal{B}$ 's run-time overhead is  $O(1)$  to answer each of  $\mathcal{A}$ 's queries and to compute the final output.  $\square$

Scheme	R.O.	Seq.	Keys	Size	Verification	Signing
BGLS	YES	NO	Chosen	160 bits	$l + 1$ pairings	1 exp.
LMRS-1	YES	YES	Chosen	1024 bits	$2l$ exp.	ver. + 1 exp.
LMRS-2	YES	YES	Reg.	1024 bits	$4l$ mult.	ver. + 1 exp.
Ours	NO	YES	Reg.	320 bits	2 pair., $lk/2$ mult.	ver. + 1 exp.

Table 3.1: Comparison of aggregate signature schemes.

Signatures are by  $l$  signers;  $k$  is the output length of a collision resistant hash function; “R.O.” denotes if the security proof uses random oracles. The key model indicates whether we are in the chosen-key model or the registered key model.

### 3.6 Comparison to Previous Work

In this section, we compare the schemes we have presented to previous schemes in the literature. For the comparison, we instantiate pairing-based schemes using Barreto-Naehrig curves [BN05] with 160-bit point representation. Note that BLS-based constructions must compute, for signing and verification, a hash function onto  $\mathbb{G}$ . This is an expensive operation [BLS04, Sect. 3.2].

**Sequential Aggregate Signatures.** We compare our sequential aggregate signature scheme to the aggregate scheme of Boneh et al. [BGL03] (BGLS) and to the sequential aggregate signature scheme of Lysyanskaya et al. [LMR04] (LMRS).

We instantiate the LMRS scheme using the RSA-based permutation family with common domain devised by Hayashi, Okamoto, and Tanaka [HOT04]. With this permutation family LMRS signatures do not grow by 1 bit with each signature, as is the case with the RSA-based instantiation given by Lysyanskaya et al. [LMR04]; but evaluating the permutation requires two applications of the underlying RSA function. Lysyanskaya et al. give two variants of their scheme. One places constraints on the for-

mat of the RSA keys, thereby avoiding key certification; we call this variant LMRS-1. The other uses ordinary RSA keys and can have public exponent  $e = 3$  for fast verification, but requires key certification, like our scheme; we call this variant LMRS-2.

We present the comparisons in Table 3.6. The size column gives signature length at the 1024-bit security level. The Verification and Signing columns give the computational costs of those operations;  $l$  is the number of signatures in an aggregate, and  $k$  is the output length of a collision-resistant hash function.

One drawback of our scheme is that a user's public key will be quite large. If we use a 160-bit collision resistant hash function, then keys will be approximately 160 group elements and take around 10KB to store. While it is desirable to achieve smaller public keys, this will be acceptable in many settings such as SBGP where achieving the signature size is a much more important consideration than the public key size. Additionally, Naccache [Nac07] and Chatterjee and Sarkar [CS05] independently proposed ways to achieve shorter public keys in the Waters signature scheme. Using these methods we can also achieve considerably shorter public keys.

**Multisignatures.** We compare our multisignature scheme to the Boldyreva multisignature [Bol03]. We present the comparisons in Table 3.6. The size column gives signature length at the 1024-bit security level. The Verification and Signing columns give the computational costs of those operations;  $l$  is the number of signatures in a multisignature, and  $k$  is the output length of a collision-resistant hash function.

**Verifiably Encrypted Signatures.** We compare our verifiably encrypted signature scheme to that of Boneh et al. [BGL03] (BGLS). We present the comparisons in Table 3.6. The size column gives signature length at the 1024-bit security level. The Verification and Generation columns give the computational costs of those operations;

Scheme	R.O.	Key Model	Size	Verification	Signing
Boldyreva	YES	Registered	160 bits	2 pairings	1 exp.
Ours	NO	Registered	320 bits	2 pairings, $k/2$ mult.	1 exp.

Table 3.2: Comparison of multisignature schemes.

Multisignatures are by  $l$  signers;  $k$  is the output length of a collision resistant hash function; “R.O.” denotes if the security proof uses random oracles.

Scheme	R.O.	Key Model	Size	Verification	Generation
BGLS	YES	Registered	320 bits	3 pairings	3 exp.
Ours	NO	Registered	480 bits	3 pairings, $k/2$ mult.	4 exp.

Table 3.3: Comparison of verifiably encrypted signature schemes.

We let  $k$  be the output length of a collision resistant hash function. “R.O.” specifies whether the security proof uses random oracles.

$k$  is the output length of a collision-resistant hash function.

### 3.7 Conclusions and Open Problems

In this chapter we gave the first aggregate signature scheme which is provably secure without random oracles; the first multisignature scheme which is provably secure without random oracles; and the first verifiably encrypted signature scheme which is provably secure without random oracles. All our constructions derive from the recent signature scheme due to Waters [Wat05]. All our constructions are quite practical.

Signatures in our aggregate signature scheme are sequentially constructed, but knowledge of the order in which messages are signed is not necessary for verification. Additionally, our scheme gives shorter signatures than in the LMRS sequential aggregate signature scheme [LMR04] and has a more efficient verification algorithm

than the BGLS aggregate signature scheme [BGL03]. This gives some interesting tradeoffs for practical applications such as secure routing and proxy signatures.

Some interesting problems remain open for random-oracle-free aggregate signatures:

1. To find a scheme which supports full aggregation, in which aggregate signatures do not need to be sequentially constructed. While many applications only require sequential aggregation, having a more general capability is desirable.
2. To find a sequential aggregate signature scheme provably secure in the chosen-key model.
3. To find a sequential aggregate signature scheme with shorter user keys. The size of public keys in our system reflects the size of keys in the underlying Waters signature scheme. Naccache [Nac07] and Chatterjee and Sarkar [CS05] have proposed ways to shorten the public keys of the Waters IBE/signature scheme by trading off parameter size with tightness in the security reduction. It would be better to have a solution in which the public key is just a few group elements.

The last two are particularly important for certificate chain compression, proposed by Boneh et al. [BGL03] as an application for aggregate signatures. If keys need to be registered with an authority then a chaining application is impractical, and having large public keys negates any benefit from reducing the signature size in a certificate chain, since the keys must be included in the certificates.

## CHAPTER 4

### Non-Interactive Shuffle with Pairing-Based Verifiability

We briefly describe the organization of this chapter (see the introductory discussion in Chapter 1, Section 1.2).

In this chapter, we present an efficient non-interactive verifiable shuffle by using bilinear pairings. We begin in Section 4.1 by giving background information pertinent to our results. In Section 4.2, we present two new cryptographic assumptions and provide heuristic evidence by showing that they hold in the generic group model. In Section 4.3, we present our scheme *GL-SHUF* and prove its security. In Section 4.4, we remark on shuffling ciphertexts of a different pairing-based cryptosystem under a different assumption. Finally, in Section 4.5 we conclude and discuss open problems.

#### 4.1 Background

##### 4.1.1 BBS Encryption

The BBS cryptosystem was introduced by Boneh, Boyen and Shacham [BBS04]. Following prior notation, we work in a bilinear group  $(p, \mathbb{G}, \mathbb{G}_T, e, P)$ . The public key is of the form  $(Q = xP, R = yP)$ . The secret key is  $(x, y) \in (\mathbb{Z}/p\mathbb{Z})^2$ . To encrypt  $m \in G$ , we choose random  $s, t \in \mathbb{Z}/p\mathbb{Z}$  and let the ciphertext be

$$(u, v, w) := (sQ, tR, (s + t)P + m).$$

To decrypt a ciphertext  $(u, v, w) \in G^3$ , we compute  $m = w - (1/x)u - (1/y)v$ . The BBS cryptosystem is semantically secure under chosen plaintext attack if the Decisional Linear Problem is hard in the bilinear group. We refer to Section 2.3.2 for a formal definition of this assumption.

#### 4.1.2 Shuffling BBS Ciphertexts

The BBS cryptosystem is homomorphic in the sense that entrywise addition<sup>1</sup> of two ciphertexts yields an encryption of the addition of the plaintexts. We have:

$$\begin{aligned} & (sQ, tR, (s+t)P + m) \\ + & (SQ, TR, (S+T)P + M) = ((s+S)Q, (t+T)R, \\ & (s+S+t+T)P + (m+M)). \end{aligned}$$

It is easy to make a random shuffle of BBS ciphertexts. Given  $n$  input ciphertexts, we permute them randomly and then re-encrypt them by multiplying them with random encryptions of the identity element of  $G$ . Multiplication with encryptions of 1 preserves the plaintexts by the homomorphic property, but the plaintexts now appear in permuted order. If the Decisional Linear Assumption holds, the BBS cryptosystem is semantically secure and thus the permutation is hidden. For notational purposes, we will let  $\{x_i\}$  denote  $\{x_i\}_{i=1}^n$ .

**Definition 4.1.1.** A *shuffle* of  $n$  BBS ciphertexts  $\{(u_i, v_i, w_i)\}$  is a list of output ciphertexts  $\{(U_i, V_i, W_i)\}$  such that there exists some permutation  $\pi \in S_n$  and randomizers  $\{(S_i, T_i)\}$  so that:

$$(\forall i) \quad U_i = u_{\pi(i)} + (S_i)Q \quad \wedge \quad V_i = v_{\pi(i)} + (T_i)R \quad \wedge \quad W_i = w_{\pi(i)} + (S_i + T_i)P.$$

---

<sup>1</sup>Recall that we use additive notation for our source groups. By addition here we mean the group operation.



### 4.1.3 Non-interactive Zero-Knowledge Arguments

We will construct non-interactive zero-knowledge (NIZK) arguments for correctness of a shuffle of  $n$  BBS ciphertexts. Informally, such an argument will demonstrate that the shuffle is correct, but will not reveal anything else, in particular the permutation will remain secret. We will now define NIZK arguments with perfect completeness, perfect zero-knowledge and  $\mathcal{R}_{\text{co}}$ -soundness. The notion of co-soundness in NIZK arguments for NP-languages was introduced in the full paper of [GOS06b, GOS06a]. We will give some further intuition about these arguments after the formal definitions.

An NIZK argument for  $\mathcal{R}$  with  $\mathcal{R}_{\text{co}}$ -soundness consists of six probabilistic polynomial time algorithms: a setup algorithm  $\text{Setup}$ , a CRS generation algorithm  $\text{K}$ , a prover  $\text{Pr}$ , a verifier  $\text{V}$  and simulators  $(S_1, S_2)$ . The setup algorithm outputs some initial information  $\text{GK}$ . The CRS generation algorithm produces a common reference string  $\sigma$  corresponding to the setup. The prover takes as input  $(\text{GK}, \sigma, x, w)$  and produces a proof  $\psi$ . The verifier takes as input  $(\text{GK}, \sigma, x, \psi)$  and outputs `valid` if the proof is acceptable and `invalid` if the proof is rejected. The simulator  $S_1$  takes as input  $\text{GK}$  and outputs a simulated common reference string  $\sigma$  as well as a simulation trapdoor  $\tau$ .  $S_2$  takes as input  $\text{GK}, \sigma, \tau, x$  and simulates a proof  $\psi$ .

**Definition 4.1.2.** We call  $(\text{Setup}, \text{K}, \text{Pr}, \text{V}, S_1, S_2)$  an NIZK argument for a relation  $\mathcal{R}$  with  $\mathcal{R}_{\text{co}}$ -soundness if for all non-uniform adversaries  $\mathcal{A}$  we have completeness, soundness and zero-knowledge as described below.

**Perfect completeness:**

$$\Pr \left[ \text{GK} \leftarrow \text{Setup}(1^k); \sigma \leftarrow \text{K}(\text{GK}); (x, w) \leftarrow \mathcal{A}(\text{GK}, \sigma); \right. \\ \left. \psi \leftarrow \text{Pr}(\text{GK}, \sigma, x, w); \right. \\ \left. (\text{GK}, x, w) \notin \mathcal{R} \vee \text{V}(\text{GK}, \sigma, x, \psi) = \text{valid} \right] = 1.$$

**Computational  $\mathcal{R}_{\text{co}}$ -soundness:**

$$\Pr \left[ \text{GK} \leftarrow \text{Setup}(1^k); \sigma \leftarrow \text{K}(\text{GK}); (x, \psi, w_{\text{co}}) \leftarrow \mathcal{A}(\text{GK}, \sigma) : \right. \\ \left. \text{V}(\text{GK}, \sigma, x, \psi) = \text{valid} \wedge (\text{GK}, x, w_{\text{co}}) \in \mathcal{R}_{\text{co}} \right] \approx 0.$$

**Perfect zero-knowledge:**

$$\Pr \left[ \text{GK} \leftarrow \text{Setup}(1^k); \sigma \leftarrow \text{K}(\text{GK}); (\text{St}, x, w) \leftarrow \mathcal{A}(\text{GK}, \sigma); \right. \\ \left. \psi \leftarrow \Pr(\text{GK}, \sigma, x, w) : (\text{GK}, x, w) \in \mathcal{R} \wedge \mathcal{A}(\text{St}, \psi) = 1 \right] \\ = \Pr \left[ \text{GK} \leftarrow \text{Setup}(1^k); (\sigma, \tau) \leftarrow S_1(\text{GK}); (\text{St}, x, w) \leftarrow \mathcal{A}(\text{GK}, \sigma); \right. \\ \left. \psi \leftarrow S_2(\text{GK}, \sigma, \tau, x) : (\text{GK}, x, w) \in \mathcal{R} \wedge \mathcal{A}(\text{St}, \psi) = 1 \right].$$

We remark that if  $\mathcal{R}$  ignores GK then  $\mathcal{R}$  defines a language in NP. The definition given here generalizes the notion of NIZK arguments by allowing  $\mathcal{R}$  to depend on a setup. The setup we have in mind in this chapter, is to let GK be a description of a bilinear group. Given GK describing a bilinear group, the relation  $\mathcal{R}$  defines a *group-dependent* language  $L$ . It is common in the cryptographic literature to assume an appropriate finite group or bilinear group has already been chosen and build protocols in this setting, so it is natural to consider NIZK arguments for setup-dependent languages as we do here.

Our definition also differs in the definition of soundness, where we let  $\mathcal{R}_{\text{co}}$  be a relation that specifies what it means to break soundness. Informally, computational  $\mathcal{R}_{\text{co}}$ -soundness can be interpreted as it being infeasible for the adversary to prove  $x \in L$  if it knows  $x \in L_{\text{co}}$ . We remark that the standard definition of soundness is a special type of  $\mathcal{R}_{\text{co}}$ -soundness. If  $R$  ignores GK and  $\mathcal{R}_{\text{co}}$  ignores GK,  $w_{\text{co}}$  and contains all  $x \notin L$ , then the definition given above corresponds to saying that it is infeasible to construct a valid proof for  $x \notin L$ .

Let us explain further, why it is worthwhile to consider  $\mathcal{R}_{\text{co}}$ -soundness in the context of non-interactive arguments with perfect zero-knowledge instead of just using

the standard definition of soundness. The problem with the standard definition appears when the adversary produces a statement  $x$  and a valid NIZK argument without knowing whether  $x \in L$  or  $x \notin L$ . In these cases it may not be possible to reduce the adversary's output to a breach of some underlying (polynomial) cryptographic hardness assumption. Abe and Fehr [AF07] give a more formal argument for this. They consider NIZK arguments with direct black-box reductions to a cryptographic hardness assumption and show that only languages in  $P/\text{poly}$  can have direct black-box NIZK arguments with perfect zero-knowledge. Since all known constructions of NIZK arguments rely on direct black-box reductions this indicates that the “natural” definition of soundness is not the right definition of soundness for perfect NIZK arguments. We note that for NIZK *proofs* there is no such problem since they are not perfect zero-knowledge except for trivial languages; and in the case of interactive arguments with perfect zero-knowledge this problem does not appear either because the security proofs rely on rewinding techniques which make it possible to extract a witness for the statement being proven.

The generalization to  $\mathcal{R}_{\text{co}}$ -soundness makes it possible to get around the problem we described above. The adversary only breaks  $\mathcal{R}_{\text{co}}$ -soundness when it knows a witness  $w_{\text{co}}$  for  $x \in L_{\text{co}}$ . By choosing  $\mathcal{R}_{\text{co}}$  the right way, this witness can make it possible to reduce a successful  $\mathcal{R}_{\text{co}}$ -soundness attack to a breach of a standard polynomial cryptographic complexity assumption.

At this point, one may wonder whether it is natural to consider a soundness definition where we require the adversary to supply some  $w_{\text{co}}$ . It turns out that many cryptographic schemes assume a setup where such a  $w_{\text{co}}$  is given automatically. One example is shuffling that we consider in this chapter: when setting up a mix-net using a homomorphic threshold cryptosystem, the threshold decryption keys can be used to decrypt the ciphertexts and check whether indeed they do constitute a shuffle or not.

In this chapter, the setup algorithm will be **Setup** that outputs a description of a bilinear group. The relation  $\mathcal{R}$  will consist of statements that contain a public key for the BBS cryptosystem using the bilinear group and a shuffle of  $n$  ciphertexts. The witness will be the permutation used in the shuffle as well as the randomness used for re-randomizing the ciphertexts. In other words:

$$\mathcal{R} = \left\{ \left( (p, \mathbb{G}, \mathbb{G}_T, e, P), (f, h, \{(u_i, v_i, w_i)\}, \{(U_i, V_i, W_i)\}), (\pi, \{(S_i, T_i)\}) \right) \mid \right. \\ \left. \pi \in S_n \wedge \forall i : U_i = u_{\pi(i)} + (S_i)Q \wedge V_i = v_{\pi(i)} + (T_i)R \wedge \right. \\ \left. W_i = w_{\pi(i)} + (S_i + T_i)P \right\}.$$

The relation  $\mathcal{R}_{\text{co}}$  will consist of non-shuffles. The witness  $w_{\text{co}}$  will be the decryption key, which makes it easy to decrypt and check that there is no permutation matching the input plaintexts with the output plaintexts. As we remarked above, NIZK arguments for correctness of a shuffle are usually deployed in a context where such a decryption key can be found. It is for instance common in mix-nets that the mix-servers have a threshold secret sharing of the decryption key for the cryptosystem used in the shuffle. NIZK arguments with  $\mathcal{R}_{\text{co}}$ -soundness for correctness of a shuffle therefore give us exactly the guarantee we need for the shuffle being correct.

$$\mathcal{R}_{\text{co}} = \left\{ \left( (p, \mathbb{G}, \mathbb{G}_T, e, P), (f, h, \{(u_i, v_i, w_i)\}, \{(U_i, V_i, W_i)\}), (x, y) \right) \mid \right. \\ \left. x, y \in \mathbb{Z}/p\mathbb{Z} \wedge Q = xP \wedge R = yP \wedge \forall \pi \in S_n \exists i : \right. \\ \left. W_i - (1/x)U_i - (1/y)V_i \neq w_{\pi(i)} - (1/x)u_{\pi(i)} - (1/y)v_{\pi(i)} \right\}.$$

#### 4.1.4 Non-interactive WI Proofs for Bilinear Groups

We will employ the non-interactive proof techniques of Groth and Sahai [GS08]. They allow a prover to give short proofs for the existence of group elements which satisfy a list of so-called pairing product equations. With their techniques, one can prove that there exists  $x_1, \dots, x_n \in \mathbb{G}$  and  $\phi_1, \dots, \phi_n \in \mathbb{Z}/p\mathbb{Z}$  such that they simultaneously satisfy a set of pairing product equations, for instance  $\prod_{i=1}^n e(a_i, x_i) = 1$  and

$\sum_{i=1}^n (\phi_i)x_i = 0$ . One instantiation of their scheme works over bilinear groups where the Decisional Linear Assumption holds.

Their scheme has the following properties. It has a key generation algorithm that outputs a common reference string consisting of 8 group elements. These 8 group elements specify the public key for two commitment schemes: one for group elements in  $\mathbb{G}$  and one for elements in  $\mathbb{Z}/p\mathbb{Z}$ . In their proof, the prover commits to the witness by committing to the group elements  $x_1, \dots, x_n \in \mathbb{G}$  and the values  $\phi_1, \dots, \phi_n \in \mathbb{Z}/p\mathbb{Z}$ . After that the prover makes non-interactive proofs that the committed elements satisfy all the pairing product equations.

There are two ways of setting up the commitment schemes. One can choose the common reference string such that both commitment schemes are perfectly binding, in which case the proof has perfect completeness and perfect soundness. With a perfect binding key, the commitments to group elements are BBS ciphertexts, so we can decrypt the commitments to learn  $x_1, \dots, x_n$ .

Another way to choose the common reference string is to have perfectly hiding commitment schemes. In this case, we can set up the commitment to the values  $\phi_1, \dots, \phi_n$  as a perfect trapdoor commitment scheme. We can create a commitment and two different openings to respectively 0 and 1 for instance. When we have perfectly hiding keys in the common reference string, the non-interactive proof has perfect completeness and perfect witness-indistinguishability. In other words, an adversary that sees a proof for a statement for which two or more witnesses exist, gets no information whatsoever as to whether one witness or the other was used in the non-interactive proof.

We write  $(\sigma_{\text{binding}}, \xi_{\text{extraction}}) \leftarrow \mathbf{K}_{\text{binding}}(p, \mathbb{G}, \mathbb{G}_T, e, P)$ , when creating a perfectly binding common reference string with extraction key  $\xi_{\text{extraction}}$  for the commitments to group elements in  $G$ . We write  $(\sigma_{\text{hiding}}, \tau_{\text{trapdoor}}) \leftarrow \mathbf{K}_{\text{hiding}}(p, \mathbb{G}, \mathbb{G}_T, e, P)$  when

creating a perfect hiding common reference string with trapdoor  $\tau_{\text{trapdoor}}$  for the commitments to values in  $\mathbb{Z}/p\mathbb{Z}$ . Perfect binding common reference strings and perfect hiding common reference strings are computationally indistinguishable if the Decisional Linear Assumption holds for the bilinear group we are working over.

## 4.2 Cryptographic Assumptions

The security of our NIZK argument for correctness of a shuffle will be based on three assumptions: the Decisional Linear Assumption, the Permutation Pairing Assumption and the Simultaneous Pairing Assumption. The BBS cryptosystem and the non-interactive proofs of Groth and Sahai rely on the Decisional Linear Assumption. The other two assumptions are needed for the NIZK argument for correctness of a shuffle. We will now formally define these two new assumptions and give heuristic reasons for believing them by showing that they hold in the generic group model.

### 4.2.1 Permutation Pairing Assumption

The Permutation Pairing Problem is: Given  $(p, \mathbb{G}, \mathbb{G}_T, e, P)$  and  $g_1 := x_1P, \dots, g_n := x_nP, \gamma_1 := x_1^2P, \dots, \gamma_n := x_n^2P$  for random  $x_1, \dots, x_n \in \mathbb{Z}/p\mathbb{Z}$  find elements  $a_1, \dots, a_n, b_1, \dots, b_n \in \mathbb{G}$  such that the following holds:

$$\begin{aligned} \sum_{i=1}^n a_i &= \sum_{i=1}^n g_i \\ \sum_{i=1}^n b_i &= \sum_{i=1}^n \gamma_i \\ e(a_i, a_i) &= e(P, b_i) \text{ for } i = 1 \dots n \\ \{a_i\} &\text{ is not a permutation of } \{g_i\} \end{aligned}$$

Note that if  $\{a_i\}$  is a permutation of  $\{g_i\}$ , then by the third equation  $\{b_i\}$  is  $\{\gamma_i\}$  permuted in the same way.

Observe that permutations trivially satisfy the first three conditions and not the fourth, but one could imagine some particular choice of the  $\{a_i\}$  and  $\{b_i\}$  would satisfy all four conditions. The *Permutation Pairing Assumption* holds if finding such a suitable choice is computationally infeasible.

**Definition 4.2.1.** The Permutation Pairing Assumption holds if for all non-uniform polynomial time adversaries  $\mathcal{A}$  we have:

$$\Pr \left[ \text{GK} := (p, \mathbb{G}, \mathbb{G}_T, e, P) \leftarrow \text{Setup}(1^k); x_1, \dots, x_n \xleftarrow{R} \mathbb{Z}/p\mathbb{Z}; \right. \\ \{g_i\} := \{x_i P\}; \{\gamma_i\} := \{x_i^2 P\}; (\{a_i\}, \{b_i\}) \leftarrow \mathcal{A}(\text{GK}, \{g_i\}, \{\gamma_i\}) : \\ \sum_{i=1}^n (a_i - g_i) = 0 \wedge \sum_{i=1}^n (b_i - \gamma_i) = 0 \wedge \\ (\forall i) e(a_i, a_i) \cdot e(P, b_i)^{-1} = 1 \wedge \\ \left. \{a_i\} \text{ is not a permutation of } \{g_i\} \right] \approx 0$$

#### 4.2.2 Simultaneous Pairing Assumption

The Simultaneous Pairing Problem is: Given  $(p, \mathbb{G}, \mathbb{G}_T, e, P)$  and  $g_1 := x_1 P, \dots, g_n := x_n P, \gamma_1 := x_1^2 P, \dots, \gamma_n := x_n^2 P$  for random  $x_1, \dots, x_n \in \mathbb{Z}/p\mathbb{Z}$  find a non-trivial set of elements  $\mu_1, \dots, \mu_n \in \mathbb{G}$  such that the following holds:

$$\prod_{i=1}^n e(\mu_i, g_i) = 1 \quad \wedge \quad \prod_{i=1}^n e(\mu_i, \gamma_i) = 1.$$

The intuition behind this problem is that it may be hard to find a set of non-trivial elements to simultaneously satisfy two pairing products of “independent” sets of elements. The *Simultaneous Pairing Assumption* holds if this problem is hard.

**Definition 4.2.2.** The Simultaneous Pairing Assumption holds if for all non-uniform

polynomial time adversaries  $\mathcal{A}$  we have:

$$\Pr \left[ \text{GK} := (p, \mathbb{G}, \mathbb{G}_T, e, P) \leftarrow \text{Setup}(1^k); x_1, \dots, x_n \xleftarrow{R} \mathbb{Z}/p\mathbb{Z}; \right. \\ \left. \{g_i\} := \{x_i P\}; \{\gamma_i\} := \{x_i^2 P\}; \{\mu_i\} \leftarrow \mathcal{A}(\text{GK}, \{g_i\}, \{\gamma_i\}) : \right. \\ \left. \prod_{i=1}^n e(\mu_i, g_i) = 1 \wedge \prod_{i=1}^n e(\mu_i, \gamma_i) = 1 \wedge \exists i : \mu_i \neq 1 \right] \approx 0$$

### 4.2.3 Our Assumptions in the Generic Group Model

We will provide heuristic evidence for our new assumptions by showing that they hold in the generic group model [Sho97]. In this model the adversary is restricted to using only generic bilinear group operations and evaluating equality of group elements.

We accomplish this restriction of the adversary by using a model of the bilinear group where we encode the group elements (or equivalently we encode their discrete logarithms) as unique random strings and letting the adversary see only this representation of the group elements. We then provide the adversary with a bilinear group operation oracle such that it can still perform group operations.

Let us give a few more details. We start by picking a random bilinear group  $(p, \mathbb{G}, \mathbb{G}_T, e, P) \leftarrow \text{Setup}(1^k)$ , which the adversary gets as input. We also pick random bijections  $[\cdot] : \mathbb{Z}/p\mathbb{Z} \rightarrow \mathbb{G}$  and  $[[\cdot]] : \mathbb{Z}/p\mathbb{Z} \rightarrow \mathbb{G}_T$ . We give the adversary access to an oracle that operates as follows:

- On input (**scalar**,  $a$ ) return  $[a]$ .
- On input (**add**,  $[a], [b]$ ) return  $[a + b]$ .
- On input (**mult**,  $[[a]], [[b]]$ ) return  $[[a + b]]$ .
- On input (**map**,  $[a], [b]$ ) return  $[[ab]]$ .

This oracle corresponds to the effect scalar multiplications (e.g.  $aP$ ), group operations



and using the bilinear map have on the discrete logarithms (base  $P$  in  $\mathbb{G}$  and base  $e(P, P)$  in  $\mathbb{G}_T$ ) of group elements. Please note that other operations such as inversion of a group element for instance can be easily computed using these group operations since the group order  $p$  is known to the adversary.

**Theorem 4.2.3.** *The Permutation Pairing Assumption holds in the generic (bilinear) group model.*

*Proof.* Let us first formulate the Permutation Pairing Assumption in the generic group model. We generate  $(p, \mathbb{G}, \mathbb{G}_T, e, P) \leftarrow \text{Setup}(1^k)$ . We pick  $[\cdot] : \mathbb{Z}/p\mathbb{Z} \rightarrow \mathbb{G}$  and  $[[\cdot]] : \mathbb{Z}/p\mathbb{Z} \rightarrow \mathbb{G}_T$  as random bijective functions. We pick  $x_1, \dots, x_n \leftarrow \mathbb{Z}/p\mathbb{Z}$ . We now give the adversary  $\mathcal{A}$  the following input:  $(p, \mathbb{G}, \mathbb{G}_T, e, P, \{[x_i]\}, \{[x_i^2]\})$  and access to the bilinear group operation oracle.  $\mathcal{A}$  is computationally unbounded but can only make a polynomial number of queries to the bilinear group operation oracle. The challenge for  $\mathcal{A}$  is to find  $\{([a_i], [b_i])\}$  so:

$$\sum_{i=1}^n a_i = \sum_{i=1}^n x_i \quad \wedge \quad \sum_{i=1}^n b_i = \sum_{i=1}^n x_i^2 \quad \wedge \quad \forall i : a_i^2 = b_i \quad \wedge \quad \forall \pi \exists i : a_i \neq x_{\pi(i)}.$$

In the generic group model we can without loss of generality assume the adversary computes  $[a_i], [b_i]$  via repeated calls to the group operation oracle. This means we have

$$a_i = \sum_{j=1}^n x_j a_{ij} + \sum_{j=1}^n x_j^2 \alpha_{ij} + r_i \quad , \quad b_i = \sum_{j=1}^n x_j b_{ij} + \sum_{j=1}^n x_j^2 \beta_{ij} + s_i$$

for values  $\{a_{ij}\}, \{\alpha_{ij}\}, \{r_i\}, \{b_{ij}\}, \{\beta_{ij}\}, \{s_i\}$  that can be deduced from the calls to the group operation oracle.

Consider now the first conditions on the adversary being successful:

$$\sum_{i=1}^n a_i - \sum_{i=1}^n x_i = 0 \quad \wedge \quad \sum_{i=1}^n b_i - \sum_{i=1}^n x_i^2 = 0 \quad \wedge \quad \forall i : a_i^2 = b_i.$$

These are polynomials over unknowns  $x_1, \dots, x_n$  that are randomly chosen. The adversary only has indirect access to them by using the bilinear group operation oracle.

The adversary can choose two strategies for satisfying the equations. It can pick the values  $a_{ij}, \alpha_{ij}, r_i, b_{ij}, \beta_{ij}, s_i$  so the polynomials are identical to zero in  $\mathbb{Z}/p\mathbb{Z}[x_1, \dots, x_n]$  or it can hope to be lucky that the polynomials evaluate to zero on the random choice of  $x_1, \dots, x_n \leftarrow \mathbb{Z}/p\mathbb{Z}$ . The Schwartz-Sippel theorem tells us that a guess according to the latter strategy has only negligible probability of being successful. Since the adversary can access the bilinear group operation oracle only a polynomial number of times, it can only verify a polynomial number of guesses, so the latter strategy has negligible success probability.

Let us now see what happens if the adversary follows the first strategy. The first equation gives us:

$$\sum_{i=1}^n \left( \sum_{j=1}^n x_j a_{ij} + \sum_{j=1}^n x_j^2 \alpha_{ij} + r_i \right) - \sum_{i=1}^n x_i = 0.$$

Viewed as a multivariate polynomial equation over variables  $x_1, \dots, x_n$  we must have for all  $j$ ,  $\sum_{i=1}^n a_{ij} = 1$  and  $\sum_{i=1}^n \alpha_{ij} = 0$  and  $\sum_{i=1}^n r_i = 0$ .

Next, if  $\prod_{i=1}^n b_i = \sum_{i=1}^n x_i^2$  then it must be the case that

$$\sum_{i=1}^n \left( \sum_{j=1}^n x_j b_{ij} + \sum_{j=1}^n x_j^2 \beta_{ij} + s_i \right) - \sum_{i=1}^n x_i^2 = 0.$$

When viewed as a polynomial in  $x_1, \dots, x_n$ , we see that we must have for all  $j$ ,  $\sum_{i=1}^n b_{ij} = 0$  and  $\sum_{i=1}^n \beta_{ij} = 1$  and  $\sum_{i=1}^n s_i = 0$ .

Finally, if  $(\forall i) a_i^2 = b_i$  then it must be the case that

$$\begin{aligned} & \sum_{j=1}^n \sum_{k=1}^n x_j x_k a_{ij} a_{ik} + \sum_{j=1}^n \sum_{k=1}^n x_j^2 x_k^2 \alpha_{ij} \alpha_{ik} + r_i^2 \\ & + 2 \sum_{j=1}^n \sum_{k=1}^n x_j x_k^2 a_{ij} \alpha_{ik} + 2 \sum_{j=1}^n x_j a_{ij} r_i + 2 \sum_{j=1}^n x_j^2 \alpha_{ij} r_i \\ & = \sum_{j=1}^n x_j b_{ij} + \sum_{j=1}^n x_j^2 \beta_{ij} + s_i \end{aligned}$$

Once again by viewing this as a polynomial equation, for all  $i$  we must have that  $a_{ij}\alpha_{ik} = 0$ . Also  $a_{ij}\alpha_{ik} = 0$  when  $j \neq k$ ,  $r_i^2 = s_i$ ,  $b_{ij} = 2a_{ij}r_i$ ,  $\beta_{ij} = a_{ij}^2 + 2\alpha_{ij}r_i$ .

We now consider what the matrix  $A = (a_{ij})$  must be. Each row of  $A$  has at most one non-zero entry by the fact that  $a_{ij}\alpha_{ik} = 0$  when  $j \neq k$ . Also, each column must sum to 1 by  $\sum_{i=1}^n a_{ij} = 1$ . These two facts combined imply  $A$  has exactly one 1 in each column and each row, thus  $A$  is a permutation matrix. Since permutation matrices are invertible, from the equations  $\sum_{i=1}^n a_{ij}\alpha_{ik} = \sum_{i=1}^n 0 = 0$ ,  $\sum_{i=1}^n a_{ij}r_i = \frac{1}{2}\sum_{i=1}^n b_{ij} = 0$ , we obtain that  $\alpha_{ik} = 0$  and  $r_i = 0$ . Therefore, the  $\{a_i\}$  are a permutation of the  $\{x_i\}$ .  $\square$

**Theorem 4.2.4.** *The Simultaneous Pairing Assumption holds in the generic (bilinear) group model.*

*Proof.* Let us first formulate the Simultaneous Pairing Assumption in the generic group model. We generate  $(p, \mathbb{G}, \mathbb{G}_T, e, P) \leftarrow \text{Setup}(1^k)$ . We pick  $[\cdot] : \mathbb{Z}/p\mathbb{Z} \rightarrow G$  and  $[[\cdot]] : \mathbb{Z}/p\mathbb{Z} \rightarrow G_T$  as random bijective functions. We pick  $x_1, \dots, x_n \leftarrow \mathbb{Z}/p\mathbb{Z}$ . We now give the adversary  $\mathcal{A}$  the following input:  $(p, \mathbb{G}, \mathbb{G}_T, e, P, \{[x_i]\}, \{[x_i^2]\})$  and access to the bilinear group operation oracle.  $\mathcal{A}$  is computationally unbounded but can only make a polynomial number of queries to the bilinear group operation oracle. The challenge for  $\mathcal{A}$  is to find non-trivial  $\{[\mu u_i]\}$  so  $\sum_{i=1}^n \mu_i x_i = 0$  and  $\sum_{i=1}^n \mu_i x_i^2 = 0$ . The Simultaneous Pairing Assumption in the generic model says that any adversary  $\mathcal{A}$  has negligible probability of succeeding in this game.

Without loss of generality we can think of  $\mathcal{A}$  as being restricted to computing  $\{[\mu_i]\}$  using the bilinear group operation oracle only. This means it chooses

$$\mu_i = \sum_{j=1}^n x_j a_{ij} + \sum_{j=1}^n x_j^2 \alpha_{ij} + r_i$$

for known  $a_{ij}$ ,  $\alpha_{ij}$  and  $r_i$ .

A successful adversary chooses these values so both of these equations are satisfied:

$$\sum_{i=1}^n \left( \sum_{j=1}^n x_j a_{ij} + \sum_{j=1}^n x_j^2 \alpha_{ij} + r_i \right) x_i = 0$$

$$\sum_{i=1}^n \left( \sum_{j=1}^n x_j a_{ij} + \sum_{j=1}^n x_j^2 \alpha_{ij} + r_i \right) x_i^2 = 0$$

We can view them as multi-variate polynomials in  $x_1, \dots, x_n$  which are chosen at random. The adversary never sees  $x_1, \dots, x_n$ , it only has indirect access to them through the group operation oracle. There are two strategies the adversary can use: It can select  $a_{ij}, \alpha_{ij}, r_i$  so the two polynomials have zero-coefficients or it can hope to be lucky that the random choice of  $x_1, \dots, x_n$  actually evaluates zero. The Schwartz-Sippel theorem tells us that a guess has negligible chance of being correct when  $x_1, \dots, x_n$  are chosen at random from  $\mathbb{Z}/p\mathbb{Z}$ . Since the adversary can access the bilinear group operations oracle only a polynomial number of times, it can only verify the correctness of a polynomial number of guesses. The latter strategy therefore has negligible success-probability.

Let us now consider the former strategy, where the adversary chooses the coefficients of the polynomials in  $\mathbb{Z}/p\mathbb{Z}[x_1, \dots, x_n]$  so they are the zero-polynomials. Looking at the coefficients for the first polynomial we see that we must have  $r_i = 0$  and  $\alpha_{ij} = 0$ . Looking at the coefficients of the second polynomial we see that  $a_{ij} = 0$ . The adversary can therefore only find the trivial solution  $\mu_1 = \dots = \mu_n = 0$ .  $\square$

### 4.3 NIZK Argument for Correctness of a Shuffle

We will now present an NIZK argument for correctness of a shuffle of BBS ciphertexts. The common reference string contains  $2n$  elements  $\{g_i := x_i P\}$  and  $\{\gamma_i := x_i^2 P\}$  for random  $x_1, \dots, x_n \in \mathbb{Z}/p\mathbb{Z}$ . The statement contains a public key  $(f, h)$  and a set of  $n$

input ciphertexts  $\{(u_i, v_i, w_i)\}$  and a set of output ciphertexts  $\{(U_i, V_i, W_i)\}$  that may be a shuffle of the input ciphertexts.

The first part of the NIZK argument consists of setting up pairing product equations that can only be satisfied if indeed we are dealing with a shuffle. The prover will use a set of variables  $\{a_i\}$  and  $\{b_i\}$  in these pairing product equations. She will set up a Permutation Pairing Problem over these variables to guarantee that  $\{(a_i, b_i)\}$  are a permutation of  $\{(g_i, \gamma_i)\}$ .

Assume now that  $\{(a_i, b_i)\}$  are a permutation of  $\{(g_i, \gamma_i)\}$ . Let  $\{m_i\}$  be the plaintexts of  $\{(u_i, v_i, w_i)\}$  and  $\{M_i\}$  be the plaintexts of  $\{(U_i, V_i, W_i)\}$ . The prover also sets up equations such that  $\prod_{i=1}^n e(a_i, M_i) = \prod_{i=1}^n e(g_i, m_i)$  and  $\prod_{i=1}^n e(b_i, M_i) = \prod_{i=1}^n e(\gamma_i, m_i)$ . Since  $\{(a_i, b_i)\}$  are a permutation of  $\{(g_i, \gamma_i)\}$ , then there exists a permutation  $\pi \in S_n$  so

$$\prod_{i=1}^n e(g_i, M_{\pi^{-1}(i)} - m_i) = 1 \quad \wedge \quad \prod_{i=1}^n e(\gamma_i, M_{\pi^{-1}(i)} - m_i) = 1.$$

This is a Simultaneous Pairing Problem, and assuming the hardness of this problem we will have  $M_{\pi^{-1}(i)} = m_i$  for all  $i$ .

To give further intuition of the construction, consider a naïve protocol where the prover sends the permutation directly to the verifier. Denote  $a_i := g_{\pi(i)}$  and  $b_i := \gamma_{\pi(i)}$ . With  $U_i = u_{\pi(i)} + S_i Q$ ,  $V_i = v_{\pi(i)} + T_i R$ ,  $W_i = w_{\pi(i)} + (S_i + T_i)P$  we have:

$$\begin{aligned} \prod_{i=1}^n e(a_i, u_{\pi(i)} + S_i Q) &= e\left(\sum_{i=1}^n S_i a_i, Q\right) \prod_{i=1}^n e(g_{\pi(i)}, u_{\pi(i)}) \\ &= e(c_u, Q) \prod_{i=1}^n e(g_i, u_i), \end{aligned}$$

$$\begin{aligned} \prod_{i=1}^n e(a_i, v_{\pi(i)} + T_i R) &= e\left(\sum_{i=1}^n S_i a_i, R\right) \prod_{i=1}^n e(g_{\pi(i)}, v_{\pi(i)}) \\ &= e(c_v, R) \prod_{i=1}^n e(g_i, v_i), \end{aligned}$$

$$\begin{aligned}
\prod_{i=1}^n e(a_i, w_{\pi(i)} + (S_i + T_i)P) &= e\left(\sum_{i=1}^n S_i a_i, P\right) \prod_{i=1}^n e(g_{\pi(i)}, w_{\pi(i)}) \\
&= e(c_w, P) \prod_{i=1}^n e(g_i, w_i),
\end{aligned}$$

where  $c_u = \sum_{i=1}^n S_i a_i$ ,  $c_v = \sum_{i=1}^n T_i a_i$  and  $c_w = \sum_{i=1}^n (S_i + T_i) a_i$ . By construction,  $c_w = c_u + c_v$ . In addition, we may look at the equations by pairing the  $\{b_i\}$  with the  $U_i, V_i$ , and  $W_i$ . From this we obtain another three equations, and we define new elements  $c'_u = \sum_{i=1}^n S_i b_i$ ,  $c'_v = \sum_{i=1}^n T_i b_i$ ,  $c'_w = c'_u + c'_v$ . In total we have six equations:

$$\begin{aligned}
\prod_{i=1}^n e(a_i, U_i) &= e(c_u, Q) \prod_{i=1}^n e(g_i, u_i) & \prod_{i=1}^n e(b_i, U_i) &= e(c'_u, Q) \prod_{i=1}^n e(\gamma_i, u_i) \\
\prod_{i=1}^n e(a_i, V_i) &= e(c_v, R) \prod_{i=1}^n e(g_i, v_i) & \prod_{i=1}^n e(b_i, V_i) &= e(c'_v, R) \prod_{i=1}^n e(\gamma_i, v_i) \\
\prod_{i=1}^n e(a_i, W_i) &= e(c_u + c_v, P) \prod_{i=1}^n e(g_i, w_i) & \prod_{i=1}^n e(b_i, W_i) &= e(c'_u + c'_v, P) \prod_{i=1}^n e(\gamma_i, w_i)
\end{aligned}$$

A naïve non-interactive argument would be to let the prover send  $\pi, c_u, c_v, c'_u, c'_v$  to the verifier. The verifier can check the six above equations himself for the verification step. The naïve protocol described is complete by observation. We also have the following lemma:

**Lemma 1.** The naïve protocol is  $\mathcal{R}_{\text{co}}$ -sound.

*Proof.* The idea behind  $\mathcal{R}_{\text{co}}$ -soundness is to look at the underlying messages. If a dishonest prover were to convince a verifier with a non-shuffle as well as produce a witness (decryption key)  $w_{\text{co}} = (x, y)$ , we can “decrypt” the equations checked by the verifier. Namely, if we let  $m_i = w_i - (1/x)u_i - (1/y)v_i$  and  $M_i = W_i - (1/x)U_i - (1/y)V_i$ , then by applying the same algebraic manipulations to the equations, we obtain:

$$\begin{aligned}
&\left[\prod_{i=1}^n e(a_i, U_i)\right]^{-1/x} \left[\prod_{i=1}^n e(a_i, V_i)\right]^{-1/y} \left[\prod_{i=1}^n e(a_i, W_i)\right] \\
&= \left[e(c_u, Q) \prod_{i=1}^n e(g_i, u_i)\right]^{-1/x} \left[e(c_v, R) \prod_{i=1}^n e(g_i, v_i)\right]^{-1/y} \left[e(c_u + c_v, P) \prod_{i=1}^n e(g_i, w_i)\right].
\end{aligned}$$

This gives us  $\prod_{i=1}^n e(a_i, M_i) = e(-c_u, P)e(-c_v, P)e(c_u + c_v, P)\prod_{i=1}^n e(g_i, m_i) = \prod_{i=1}^n e(g_i, m_i)$ .

In a similar way we can show that  $\prod_{i=1}^n e(b_i, M_i) = \prod_{i=1}^n e(\gamma_i, m_i)$ . Observe that the equations may be rearranged to be  $\prod_{i=1}^n e(\mu_i, g_i) = 1$  and  $\prod_{i=1}^n e(\mu_i, \gamma_i) = 1$  where  $\mu_i = m_i - M_{\pi^{-1}(i)}$ . By the Simultaneous Pairing Assumption, it is infeasible for the prover to find non-trivial  $\mu_i$  satisfying these two equations and thus we reach a contradiction.  $\square$

The downfall of the naïve protocol is that it completely reveals the permutation. In the actual NIZK argument, we will instead argue that there exist elements  $\{a_i\}$  and  $\{b_i\}$  that satisfy the equations above rather than revealing them directly. We accomplish this by making a GS proof for the set of pairing product equations given earlier. We now give a description of our NIZK argument for a valid shuffle, *GL-SHUF*.

**Setup:** Generate a bilinear group  $\text{GK} := (p, \mathbb{G}, \mathbb{G}_T, e, P) \leftarrow \text{Setup}(1^k)$ .

**Common reference string:** Generate a perfectly hiding common reference string  $(\sigma_{\text{hiding}}, \tau_{\text{trapdoor}}) \leftarrow \mathcal{K}_{\text{hiding}}(p, \mathbb{G}, \mathbb{G}_T, e, P)$  to get perfectly witness-indistinguishable GS proofs. Pick random  $x_1, \dots, x_n \leftarrow \mathbb{Z}/p\mathbb{Z}$  and compute  $\forall i : g_i := x_i P, \gamma_i := x_i^2 P$ .

The common reference string is  $\sigma := (\sigma_{\text{hiding}}, \{g_i\}, \{\gamma_i\})$ .

**Shuffle statement:** Public key  $(Q, R)$  for the BBS cryptosystem. Input ciphertexts  $\{(u_i, v_i, w_i)\}$  and output ciphertexts  $\{(U_i, V_i, W_i)\}$ .

**Prover's input:** Permutation  $\pi \in S_n$  and randomizers  $\{(S_i, T_i)\}$  so  $U_i = u_{\pi(i)} + S_i Q$ ,  $V_i = v_{\pi(i)} + T_i R$  and  $W_i = w_{\pi(i)} + (S_i + T_i)P$  for all  $i$ .

**Proof:** The prover sets up the following pairing product equations:

$$\phi = 1 \pmod p, \quad \phi d_u = 0, \quad \phi d_v = 0, \quad \phi d_w = 0, \quad \phi d'_u = 0, \quad \phi d'_v = 0, \quad \phi d'_w = 0,$$

$$\sum_{i=1}^n (\phi a_i - \phi g_i) = 0, \quad \sum_{i=1}^n (\phi b_i - \phi \gamma_i) = 0, \quad (\forall i) e(a_i, a_i) = e(P, b_i)$$

$$\begin{aligned} e(d_u, P) \prod e(a_i, U_i) &= e(c_u, Q) \prod e(g_i, u_i) \\ e(d'_u, P) \prod e(b_i, U_i) &= e(c'_u, Q) \prod e(\gamma_i, u_i) \\ e(d_v, P) \prod e(a_i, V_i) &= e(c_v, R) \prod e(g_i, v_i) \\ e(d'_v, P) \prod e(b_i, V_i) &= e(c'_v, R) \prod e(\gamma_i, v_i) \\ e(d_w, P) \prod e(a_i, W_i) &= e(c_u + c_v, g) \prod e(g_i, w_i) \\ e(d'_w, P) \prod e(b_i, W_i) &= e(c'_u + c'_v, g) \prod e(\gamma_i, w_i) \end{aligned}$$

A witness for satisfiability of the equations can be computed as:

$$\begin{aligned} \phi := 1, \quad c_u := \sum_{i=1}^n S_i a_i, \quad c_v := \sum_{i=1}^n T_i a_i, \quad c'_u := \sum_{i=1}^n S_i b_i, \quad c'_v := \sum_{i=1}^n T_i b_i, \\ \forall i : a_i := g^{\pi(i)}, \quad b_i := \gamma^{\pi(i)}, \end{aligned}$$

and setting the remaining variables to 0. The prover generates a GS proof  $\psi$  that there exists a  $\phi \in \mathbb{Z}/p\mathbb{Z}$  and group elements  $\{a_i\}, \{b_i\}, c_u, c_v, c'_u, c'_v, d_u, d_v, d_w, d'_u, d'_v, d'_w$  that satisfy the equations.

**Verification:** The verifier accepts the non-interactive argument if and only if the GS proof  $\psi$  is valid.

**Theorem 4.3.1.** *The protocol  $\mathcal{GL}\text{-SHUF}$  is a non-interactive perfectly complete, computationally  $\mathcal{R}_{\text{co}}$ -sound, perfect zero-knowledge argument of a correct shuffle of BBS ciphertexts under the Decisional Linear Assumption, Permutation Pairing Assumption, and Simultaneous Pairing Assumption.*

*Proof.* As we see in the protocol, the prover can generate the witness for the GS proof herself. Perfect completeness follows from the perfect completeness of the GS proofs.

We will now prove that we have perfect zero-knowledge. The following is a description of how the simulator  $S = (S_1, S_2)$  will generate a transcript.



**Simulated common reference string:** The simulator  $S_1$  runs the common reference string generation protocol. It sets  $\tau := (\tau_{\text{trapdoor}}, x_1, \dots, x_n)$  and outputs  $(\sigma, \tau)$ .

**Shuffle statement:** Public key  $(Q, R)$  for the BBS cryptosystem. Input ciphertexts  $\{(u_i, v_i, w_i)\}$  and output ciphertexts  $\{(U_i, V_i, W_i)\}$ .

**Simulator's input:** The simulator  $S_2$  receives the shuffle statement and  $(\sigma, \tau)$ .

**Simulated proof:** Create a trapdoor commitment with double opening to  $\phi = 0$  and  $\phi = 1$ . Compute

$$\begin{aligned} d_u &:= \sum_{i=1}^n x_i u_i, & d_v &:= \sum_{i=1}^n x_i v_i, & d_w &:= \sum_{i=1}^n x_i w_i, \\ d'_u &:= \sum_{i=1}^n x_i^2 u_i, & d'_v &:= \sum_{i=1}^n x_i^2 v_i, & d'_w &:= \prod_{i=1}^n x_i^2 w_i. \end{aligned}$$

Set the remaining variables to 0 and create a perfect witness indistinguishable GS proof  $\psi$  that there exists a  $\phi \in \mathbb{Z}/p\mathbb{Z}$  and group elements  $\{a_i\}, \{b_i\}, c_u, c_v, c'_u, c'_v, d_u, d_v, d_w, d'_u, d'_v, d'_w$  that satisfy the required equations.

By construction, the common reference strings are generated in the same way. The only difference between a real proof and a simulated proof is the witness given to the GS proof. By the perfect witness-indistinguishability of the GS proof, real proofs and simulated proofs are perfectly indistinguishable.

It remains to prove that we have computational  $\mathcal{R}_{\text{co}}$ -soundness. The adversary is trying to output a public key  $(Q, R)$  and a non-shuffle of  $n$  input ciphertexts and  $n$  output ciphertexts, a convincing NIZK argument  $\psi$  of it being a shuffle, and a decryption key  $(x, y)$ . The relation  $\mathcal{R}_{\text{co}}$  is a polynomial time decidable relation that tests that  $(x, y)$  is the decryption key for  $(Q, R)$  and that indeed we do have a non-shuffle.

We will change the way we construct the common reference string for the NIZK argument. Instead of generating  $\sigma = (\sigma_{\text{hiding}}, \{g_i\}, \{\gamma_i\})$  as in the scheme, we return

$\sigma := (\sigma_{\text{binding}}, \{g_i\}, \{\gamma_i\})$  where  $(\sigma_{\text{binding}}, \xi_{\text{extraction}}) \leftarrow \mathcal{K}_{\text{binding}}(p, \mathbb{G}, \mathbb{G}_T, e, P)$ . By the Decisional Linear Assumption, perfect binding and perfect hiding common reference strings for the GS proofs are computationally indistinguishable, so the adversary's success probability only changes negligibly.

The commitment with trivial randomness is now a perfectly binding commitment to the variable  $\phi = 1$ . The GS proof is a perfect proof of knowledge of variables  $c_u, c_v, c'_u, c'_v, d_u, d_v, d_w, d'_u, d'_v, d'_w, \{a_i\}, \{b_i\}$  satisfying the equations, which can be extracted using  $\xi_{\text{extraction}}$ . Since  $\phi = 1$ , the equations demonstrate that  $d_u = d_v = d_w = d'_u = d'_v = d'_w = 0$ . The elements  $\{a_i\}, \{b_i\}$  satisfy a Permutation Pairing problem and the hardness of this problem tells us that with overwhelming probability they are a permutation of  $\{(g_i, \gamma_i)\}$ . Lemma 1 now gives us that there is negligible probability of  $c_u, c_v, c'_u, c'_v, \{a_i\}, \{b_i\}$  satisfying the equations and at the same time the input and output ciphertexts not being a shuffle.  $\square$

**Size of the NIZK argument.** To commit to  $\phi = 1$  we can use trivial randomness, so the commitment to  $\phi$  does not have to be included in the proof – the verifier can compute it himself. There are  $2n + 10$  variables in  $G$  and it takes 3 group elements for each commitment, so the commitments contribute a total of  $6n + 30$  group elements towards the proof size.

The first 6 equalities cost 9 group elements each for a total of 54 group elements. The next two multi-exponentiation equations cost 9 group elements each for a total of 18 group elements. We then have  $n$  pairing product equations of the form  $e(a_i, a_i) = e(P, b_i)$  which cost a total of  $9n$  group elements. Finally, we have 6 pairing product equations, where one side of the pairings is publicly known and one side is committed. They each cost 3 group elements for a total of 18 group elements.

The total size of the proof is  $15n + 120$  group elements. The size of the common

reference string is  $2n + 8$  group elements.<sup>2</sup>

We remark that the cost of shuffling multiple sets of ciphertexts with the same permutation may be amortized to a constant number of group elements. The first set of ciphertexts costs  $15n + 120$  group elements. But we only need to commit to  $a_i, b_i$  and prove  $e(a_i, a_i) = e(P, b_i)$  once. Regardless of  $n$ , the subsequent shuffles under the same permutation only cost 120 group elements each.

#### 4.4 Remark on Shuffling BGN Ciphertexts

Another homomorphic cryptosystem over bilinear groups was introduced by Boneh, Goh and Nissim [BGN05]. This cryptosystem is based on the Subgroup Decision Assumption over composite order bilinear groups. The ciphertexts consist of one group element each, so with  $n$  input ciphertexts and  $n$  outputs ciphertexts, the shuffle statement contains  $2n$  group elements and another  $O(1)$  group elements to describe the public key. The techniques we have presented in this chapter can also be used to shuffle BGN ciphertexts. Assuming the Subgroup Decision Assumption holds and assuming suitable variants of the Permutation Pairing and the Simultaneous Pairing Assumptions hold, we can make an NIZK argument for correctness of a shuffle consisting of  $3n + O(1)$  group elements. Since the Subgroup Decision Assumption only holds when factoring the group order is hard, the group elements in this scheme are quite large.

While this scheme may have applications, we note that there is one subtle issue that one must be careful about. The GS proofs can be instantiated with bilinear groups of composite order where the Subgroup Decision Problem is hard, but they are only

---

<sup>2</sup>One could wish for a common reference string that has only a constant number of group elements, but currently even all known 3-move zero-knowledge arguments have common reference strings of size  $\Omega(n)$ .

secure if the factorization of the composite group is unknown. The decryption key for the cryptosystem is the factorization of the group order. The  $\mathcal{R}_{\text{co}}$ -soundness of the scheme therefore only holds as long as the adversary does not know the decryption key for the cryptosystem. The NIZK argument is therefore not  $\mathcal{R}_{\text{co}}$ -sound as defined in this chapter, albeit it will satisfy a suitably weakened  $\mathcal{R}_{\text{co}}$ -soundness definition.

## 4.5 Conclusions and Open Problems

In this chapter, we presented an efficient non-interactive pairing-based verifiable shuffle. This new scheme relied on two new assumptions — the Permutation Pairing Assumption and the Simultaneous Pairing Assumption — and made use of GS proofs. We gave a heuristic argument for the validity of these assumptions by showing that they hold in the generic group model.

Intuitively, our assumptions regard finding linear relations “in the exponent”, which are different in essence to multi-exponent assumptions such as the q-BDHE assumption (introduced by Boneh, Boyen, Goh [BBG05]). Discovering a reduction between the assumptions or remodeling the scheme to work under different assumptions remain an interesting open problem.

Subsequent to the results in this chapter, Groth and Ishai [GI08] proposed an interactive sub-linear zero knowledge argument for the correctness of a shuffle. Although our scheme can be amortized to perform multiple shuffles under the same permutation with an  $O(1)$  sized proof, the first shuffle proof is more costly than existing interactive shuffle proofs. It is an open problem to improve the efficiency of non-interactive shuffles to be competitive with its interactive counterparts.

## CHAPTER 5

### Accountable Authority Identity-Based Encryption

We briefly describe the organization of this chapter (see the introductory discussion in Chapter 1, Section 1.3).

First, in Section 5.1, we review background information related to our constructions. In Section 5.2, we formally define the model for an accountable authority identity-based encryption scheme. In Section 5.3.1, we discuss the requirements for the building blocks used in our general construction, which we present in Section 5.3.2. We then prove the security of our general construction in Section 5.4. In Section 5.5, we give an efficient realization of an A-IBE scheme as a concrete example of our general construction. Finally, in Section 5.6, we conclude by addressing some important open problems for future work.

#### 5.1 Background

In this section, we review some of the building blocks which we will use to construct A-IBE schemes. We begin by reminding the reader of the three schemes most relevant to our generic construction: identity-based encryption, fully simulatable oblivious transfer, and (key-policy) attribute-based encryption. We then review background information that will be used in our concrete constructions.

### 5.1.1 Identity-Based Encryption

We review the definitions of an identity-based encryption scheme as stated in [BF01, BF03]. An identity-based encryption scheme  $\mathcal{IBE}$  consists of four poly-time algorithms: **Setup**, **KeyGen**, **Encrypt**, **Decrypt**.

**Setup** The setup algorithm takes in a security parameter  $\lambda$  and outputs the public parameters  $\text{PK}$  and a master secret key  $\text{MK}$ .

**KeyGen** The user key generation algorithm takes as input  $\text{PK}$ ,  $\text{MK}$  and an identity  $\text{ID}$  and returns the private key  $d_{\text{ID}}$  corresponding to  $\text{ID}$ .

**Encrypt** The encryption algorithm takes as input  $\text{PK}$ ,  $\text{ID}$  and a message  $M$  and outputs a ciphertext  $C$ .

**Decrypt** The decryption algorithm takes as input  $\text{PK}$ ,  $C$ ,  $d_{\text{ID}}$  and outputs a message  $M$  or a special symbol  $\perp$  indicating failure.

The correctness property of an IBE scheme requires that  $d_{\text{ID}}$  should properly decrypt messages encrypted for  $\text{ID}$  with overwhelming (or perfect) probability. We also define the following IND-ID-CPA game for an IBE scheme:

**Setup** The challenger runs the Setup algorithm of IBE and gives the public parameters  $\text{PK}$  to the adversary.

**Phase 1** The adversary queries several adaptively chosen identities  $\text{ID}^1, \dots, \text{ID}^q$  and the challenger runs the key generation algorithm which returns the decryption keys  $d_{\text{ID}^1}, \dots, d_{\text{ID}^q}$ .

**Challenge** The adversary submits two equal length messages  $M_0$  and  $M_1$  and an identity  $\text{ID}^*$  not equal to any of the identities' queries in Phase 1. The challenger flips

a random coin  $b$  and encrypts  $M_b$  with ID. The ciphertext  $C$  is passed on to the adversary.

**Phase 2** This is identical to Phase 1 except that the adversary is not allowed to ask for a decryption key for  $ID^*$ .

**Guess** The adversary outputs a guess  $b'$  of  $b$ .

We say a poly-time adversary  $\mathcal{A}$  succeeds in the IND-ID-CPA game if it correctly guessed  $b' = b$ , and we define the advantage of the adversary as  $Adv_{\mathcal{A}}^{\text{IBE}}(\lambda) = |\Pr[b = b'] - \frac{1}{2}|$ . The probability is taken over the random bits used by the challenger and the adversary. Finally, we say an IBE scheme  $\text{IBE}$  is IND-ID-CPA secure if for all poly-time adversaries  $\mathcal{A}$ , the function  $Adv_{\mathcal{A}}^{\text{IBE}}(\lambda)$  is negligible in  $\lambda$ .

### 5.1.2 Fully Simulatable k-out-of-n Oblivious Transfer

Informally speaking, a k-out-of-n oblivious transfer protocol (see [EGL85]) allows a receiver to choose and receive exactly k of the n strings from the sender, such that the remaining strings are hidden from the receiver and the choice of the receiver is hidden from the sender. We require the oblivious transfer protocol to be fully simulatable (i.e. satisfy the standard Ideal/Real world definition of security, see Canetti [Can00] for more details). Various efficient constructions of k-out-of-n oblivious transfer are known based on specific assumptions such as DBDH and DDH [Lin08, GH07, CNS07].

### 5.1.3 Attribute-Based Encryption

The notion of key-policy attribute-based encryption (KP-ABE), which was introduced by Sahai and Waters [SW05], considered a user having a set of attributes ( $\mathcal{I}$ ) associ-

ated to him or her. Similarly, when encrypting, the ciphertext also has a set of attributes ( $\mathcal{J}$ ) associated to it. At a high level view, this scheme allowed a PKG to distribute user keys with a policy that a user can only decrypt when their set of attributes “properly matched” the set of attributes in the ciphertext. The original Sahai-Waters work gave constructions for threshold policies (i.e.  $|\mathcal{I} \cap \mathcal{J}| \geq \tau$  for some threshold  $\tau$ ), and this was further generalized by Goyal et al. [GPS06] for more advanced policies including those representable by trees of threshold functions. Another useful development is the Ostrovsky-Sahai-Waters non-monotonic ABE scheme which allows the access structures defining the policies to be non-monotonic. Our constructions are partially based off of these schemes; we will also have sets associated to the user and the ciphertext, and it will be convenient to keep the notion of “attributes” in mind. We refer the reader to [GPS06] for the details of the construction of an attribute-based encryption scheme. Here, we review the basic definitions of such a scheme:

An attribute-based encryption scheme  $\mathcal{ABE}$  consists of four poly-time algorithms: Setup, KeyGen, Encrypt, Decrypt.

**Setup** The setup algorithm takes in a security parameter ( $\lambda$ ) and outputs the public parameters PK and a master secret key MK.

**KeyGen** The user key generation algorithm takes as input PK, MK and a policy (e.g. an access structure)  $\mathbb{A}$  and returns the corresponding decryption key  $d_{\mathbb{A}}$ .

**Encrypt** The encryption algorithm takes as input PK, a set of attributes  $\gamma$  and a message  $M$  and outputs a ciphertext  $C$ .

**Decrypt** The decryption algorithm takes as input PK,  $C$ ,  $d_{\mathbb{A}}$  and outputs a message  $M$  or a special symbol  $\perp$  indicating failure.

The correctness property of an ABE scheme requires that  $d_{\mathbb{A}}$  should properly decrypt messages encrypted with satisfying attributes  $\gamma \in \mathbb{A}$  with overwhelming (or



perfect) probability. We also define the following Selective-Set security game for an ABE scheme:

**Init** The adversary declares a set of attributes  $\gamma^*$ .

**Setup** The challenger runs the Setup algorithm of IBE and gives the public parameters PK to the adversary.

**Phase 1** The adversary queries several adaptively chosen policies  $\mathbb{A}_1, \dots, \mathbb{A}_q$  and the challenger generates keys for these policies and returns them to the adversary. The declared set of attributes  $\gamma^*$  should not satisfy any of these policies.

**Challenge** The adversary submits two equal length messages  $M_0$  and  $M_1$ . The challenger flips a random coin  $b$  and encrypts  $M_b$  with  $\gamma^*$ . The ciphertext  $C$  is passed on to the adversary.

**Phase 2** This is identical to Phase 1.

**Guess** The adversary outputs a guess  $b'$  of  $b$ .

We say a poly-time adversary  $\mathcal{A}$  succeeds in the Selective-Set game if it correctly guessed  $b' = b$ , and we define the advantage of the adversary as  $Adv_{\mathcal{A}}^{ABE}(\lambda) = |Pr[b = b'] - \frac{1}{2}|$ . The probability is taken over the random bits used by the challenger and the adversary. Finally, we say an ABE scheme  $ABE$  is secure in the Selective-Set model if for all poly-time adversaries  $\mathcal{A}$ , the function  $Adv_{\mathcal{A}}^{ABE}(\lambda)$  is negligible in  $\lambda$ .

## 5.2 Definitions and the Model

An Accountable Authority Identity-Based Encryption scheme  $AIIBE$  consists of five components: Setup, KeyGen, Encrypt, Decrypt, Trace. These definitions are pri-

marily adapted from [Goy07] with a critical enhancement to account for fully black-box tracing.

**Setup:** There is a randomized algorithm  $\text{Setup}(\lambda)$  that takes as input: the security parameter,  $\lambda$ , and outputs the public parameters PK and a master key MK.

**Key Generation Protocol:** There is an interactive protocol  $\text{KeyGen}$  between the public parameter generator PKG and the user  $U$ . The common inputs to PKG and  $U$  are: the public parameters PK and the identity ID (of  $U$ ) for which the decryption key has to be generated. The private input to PKG is the master key MK. Additionally, PKG and  $U$  may use a sequence of random coin tosses as private inputs. At the end of the protocol,  $U$  receives a decryption key  $d_{\text{ID}}$  as its private output. At any time, either party may abort.

**Encryption:** There is a randomized algorithm  $\text{Encrypt}(M, \text{ID}, \text{PK})$  that takes as input: a message  $M$ , an identity ID, and the public parameters PK. It outputs the ciphertext  $C$ .

**Decryption:** There is an algorithm  $\text{Decrypt}(C, \text{ID}, d_{\text{ID}})$  that takes as input: the ciphertext  $C$  that was encrypted under the identity ID, the decryption key  $d_{\text{ID}}$  for ID and the public parameters PK. It outputs a message  $M$  or  $\perp$ .

**Trace:** There is a randomized algorithm  $\text{Trace}^{\text{D}}(\text{ID}, d_{\text{ID}}, \epsilon)$  that takes as input: an identity ID, a well-formed decryption key  $d_{\text{ID}}$ , a parameter  $\epsilon$  (which must be polynomially related to  $\lambda$ ), and has black-box access to a decoder box D. It runs in time polynomial in  $\lambda$  and  $1/\epsilon$  and outputs PKG, User, or Fail.

Loosely speaking, the idea behind the tracing algorithm is to allow an honest user to present her decryption key along with a captured decoder box (which decrypts her messages) to a judge to implicate the PKG of wrongdoing. At the same time, the tracing algorithm should also prevent a dishonest user from being able to falsely implicate the PKG of having created the decoder box.

To define security for an accountable authority identity-based encryption system, we first define the three following games.

**The IND-ID-CPA game.** The IND-ID-CPA game for A-IBE is very similar to the IND-ID-CPA game for standard IBE [BF01, BF03].

**Setup** The challenger runs the Setup algorithm of A-IBE and gives the public parameters PK to the adversary.

**Phase 1** The adversary runs the Key Generation protocol with the challenger for several distinct adaptively chosen identities  $ID^1, \dots, ID^q$  and gets the decryption keys  $d_{ID^1}, \dots, d_{ID^q}$ .

**Challenge** The adversary submits two equal length messages  $m_0$  and  $m_1$  and an identity ID not equal to any of the identities' queries in Phase 1. The challenger flips a random coin  $b$  and encrypts  $m_b$  with ID. The ciphertext  $C$  is passed on to the adversary.

**Phase 2** This is identical to Phase 1 except that the adversary is not allowed to ask for a decryption key for ID.

**Guess** The adversary outputs a guess  $b'$  of  $b$ .

The advantage of an adversary  $\mathcal{A}$  in this game is defined as  $|\Pr[b' = b] - \frac{1}{2}|$ .

We note that the above game can be extended to handle chosen-ciphertext attacks in the natural way by allowing for decryption queries in Phase 1 and Phase 2. Naturally, we call such an extension the IND-ID-CCA game.

We now define two games which should model the usefulness of the tracing algorithm: any decoder box D should trace back to the person who created it.

**The DishonestPKG game.** The intuition behind this game is that an adversarial PKG attempts to create a decoder box which will frame the user. Both the adver-

sary and challenger are given the security parameter  $\lambda$  as input. A second parameter  $\epsilon = \frac{1}{\text{poly}(\lambda)}$  is also given as input. The DishonestPKG game for A-IBE is defined as follows.

**Setup** The adversary (acting as an malicious PKG) generates and passes the public parameters PK and an identity ID on to the challenger. The challenger checks that PK and ID are well-formed and aborts if the check fails.

**Key Generation** The challenger and the adversary then engage in the key generation protocol to generate a decryption key for the identity ID. If neither party aborts, then the challenger gets the decryption key  $d_{\text{ID}}$  as output.

**Decryption Queries** The adversary adaptively queries ciphertexts  $C_1, \dots, C_q$  to the challenger and the challenger replies with the decrypted values.

**Create Decoder Box** The adversary outputs a decoder box D.

Let  $SF$  denote the event that the adversary wins this game, which happens if the following two conditions hold:

1. The decoder box D is  $\epsilon$ -useful for ID, i.e.  $\Pr[\text{D}(\text{Encrypt}(M, \text{ID}, \text{PK})) = M] > \epsilon$ .
2. The tracing algorithm fails to implicate the PKG, i.e.  $\text{Trace}^{\text{D}}(\text{ID}, d_{\text{ID}}, \epsilon) = \text{User}$ .

The advantage of an adversary  $\mathcal{A}$  in this game is defined as  $\Pr[SF]$  where the probability is taken over the random coins of Trace.

We note that unlike in Goyal [Goy07], the above game includes a *decryption queries* phase where the adversary adaptively queries the challenger with a sequence of ciphertexts. This phase could potentially help the adversary deduce information

about the decryption key of  $d_{ID}$  if it is able to present a *maliciously formed* ciphertext and get the challenger try to decrypt it.

**The Selective-ID DishonestUser game.** The intuition behind this game is that some colluding set of users  $ID_1, \dots, ID_q$  attempt to create a decoder box which will frame the PKG. Both the adversary and challenger are given the security parameter  $\lambda$  as input. A second parameter  $\epsilon = \frac{1}{poly(\lambda)}$  is also given as input. The Selective-ID DishonestUser game for A-IBE is defined as follows.

**Select ID** The adversary announces an  $ID^*$  to the challenger.

**Setup** The challenger runs the Setup algorithm of A-IBE and sends the public parameters PK to the adversary.

**Key Generation Queries** The adversary runs the Key Generation protocol with the challenger for several *distinct* adaptively chosen identities  $ID_1, \dots, ID_q$  and gets the decryption keys  $d_{ID_1}, \dots, d_{ID_q}$ .

**Create Decoder Box** The adversary outputs a decryption key  $d_{ID^*}$  and a decoder box D for the identity  $ID^*$  announced in the Select ID phase.

Let  $DF$  denote the event that the adversary wins this game, which happens if the following two conditions hold:

1. The decoder box D is  $\epsilon$ -useful for ID, i.e.  $\Pr[D(\text{Encrypt}(M, ID, PK)) = M] > \epsilon$ .
2. The tracing algorithm incorrectly implicates the PKG, i.e.  $\text{Trace}^D(ID, d_{ID}, \epsilon) = \text{PKG}$ .

The advantage of an adversary  $\mathcal{A}$  in this game is defined as  $\Pr[DF]$  where the probability is taken over the random coins of Trace.

We note that one can also define a full DishonestUser game where the adversary does not have to declare  $ID^*$  in advance. Our construction is only proven secure with the Selective-ID DishonestUser game, and this weakening can be seen as similar to weakening of the IND-ID-CPA game by some previously published papers [CHK03, BB04a, SW05, GPS06].

**Definition 1.** An Accountable Authority Identity-Based encryption scheme is (Selective-ID) secure if for any polynomial time adversary  $\mathcal{A}$  and any parameter  $\epsilon = \frac{1}{\text{poly}(\lambda)}$ ,  $\mathcal{A}$  has at most a negligible advantage (in  $\lambda$ ) in the IND-ID-CPA game, the DishonestPKG game and the (Selective-ID) DishonestUser game.

### 5.3 Generic construction of A-IBE

In this section, we describe a general construction of an A-IBE scheme. This construction compiles any IBE scheme into an A-IBE scheme by using (in a black-box manner) oblivious transfer and attribute-based encryption. The basic idea is to use the attribute-based encryption to ascribe a set of “dummy attributes” (cf. [Goy07]) to each user key and ciphertext. The dummy attributes of the user are shielded from the PKG by using oblivious transfer. The choice of these attributes determine which ciphertexts can be decrypted. Intuitively, there is an information gap between the PKG, which can decrypt everything, and the user, who can only decrypt based on his dummy attributes. Our construction uses a specific combinatorial structure wherein this information gap can be efficiently exploited to perform black-box tracing.

To give some more intuition, we begin by demonstrating a combinatorial example, which will be at the heart of our construction. Consider the following setup: Let  $m, n > 0$  be positive integers and  $S_1, \dots, S_m$  be sets of size  $n$ . Fix subsets  $T_i \subset S_i$  of size  $k$  each. For the sake of this example, we take  $k = \frac{3n}{5}$ . If we uniformly choose

random size  $k$  subsets  $U_i \subset S_i$ , the expected size of  $|T_i \cap U_i|$  is  $\frac{9n}{25}$ . By applying the Chernoff bound, the probability, for a fixed  $i$ , that  $|T_i \cap U_i| < \frac{7n}{25}$  is exponentially small in  $n$ . Asymptotically, if  $m$  and  $n$  are polynomially related, then by the union bound, with all but an exponentially small probability *every*  $U_i$  will intersect  $T_i$  in more than  $\frac{7n}{25}$  elements. Note, however, that there are still exponentially many size  $k$  subsets which *do not* intersect any  $T_i$  in more than  $\frac{7n}{25}$  elements.

In the context of the example above, one can imagine associating the  $T_i$  sets to a user. When someone wants to encrypt a message, he or she randomly generates the  $U_i$  sets (without knowledge of the  $T_i$  sets) and associates those to the ciphertext. If the decryption policy states that a ciphertext can be decrypted if and only if every  $T_i \cap U_i$  has more than  $\frac{7n}{25}$  elements, then an overwhelming fraction of valid ciphertexts can be properly decrypted. On the other hand, the tracing algorithm will hone in on the exponentially many ciphertexts which the user *cannot* decrypt in order to implicate a party.

As we shall see, the constants used above are not of much importance other than to guarantee certain combinatorial properties. Indeed, the intersection threshold and size of  $k$  can be replaced by any suitable constant fractions of  $n$ .

### 5.3.1 Discussion on Requirements for Building Blocks

Given an IBE scheme  $IBE$ , a  $k$ -out-of- $n$  oblivious transfer protocol  $OT$ , and an KP-ABE scheme  $ABE$ , we will show how to construct an accountable authority IBE scheme  $AIBE$  only using  $IBE$ ,  $OT$ ,  $ABE$  as black-boxes. The reader may refer back to Section 5.1 for the definitions of these schemes. While we place no restrictions on  $IBE$  and  $OT$ , we require certain properties to hold true for the key-policy attribute-based encryption scheme  $ABE$ .

The most obvious property that  $ABE$  should satisfy is the ability to capture the

scenario described above. Specifically, if identities are represented by  $\ell$ -bit strings and  $k, n, m$  are parameters (to be defined later) which are polynomially related to the security parameter  $\lambda$ , we require the following:

- $\mathcal{ABE}$  can support  $2\ell + n \cdot m$  distinct attributes.
- The encryption algorithm of  $\mathcal{ABE}$  can support  $\ell + k \cdot m$  of these attributes as input.
- Given  $\ell + n \cdot m$  attributes written as  $u_1, \dots, u_\ell$  and  $n$ -element sets  $T_j$  for  $1 \leq j \leq m$ , the key generation algorithm of  $\mathcal{ABE}$  can support user key policies of the form “The ciphertext contains all  $u_i$  and at least  $\tau$  elements from each  $T_j$ ”.

Any sufficiently expressive ABE scheme should easily satisfy the above requirements. However, we require two additional requirements for  $\mathcal{ABE}$  to satisfy. Although these two specific requirements do not necessarily apply to general ABE schemes, we emphasize that the concrete ABE schemes used in this chapter *trivially satisfy these requirements*.

**Key Components.** The first requirement is that the user key can be derived from a set of “key components”. In particular, threshold policies of the form “The ciphertext contains all  $u_i$  and at least  $\tau$  elements from each  $T_j$ ” (as above) must be representable as a collection of key components: (at least) one corresponding to “The ciphertext contains all  $u_i$ ” and one component corresponding to each attribute in  $T_j$  for all  $j$ . Furthermore, given a user key for the policy above, a new user key for the more restrictive policy “The ciphertext contains all  $u_i$  and at least  $\tau$  elements from each  $\mathcal{I}_j$ ” for any  $\mathcal{I}_j \subset T_j$  can be derived by selecting the appropriate subset of key components. The distribution of these derived user keys should also be identical to the distribution of keys which are generated directly by running **KeyGen** with the restrictive policy.



We discuss the reasoning behind this requirement. The combinatorial trick which lies in the heart of our construction can be described as: For each  $T_j$ , the user chooses a  $k$ -element subset  $\mathcal{I}_j$  *oblivious* to the PKG. This should correspond to the restricted threshold policy of the form “The ciphertext contains at least  $\tau$  elements from  $\mathcal{I}_j$ ”. The PKG cannot directly generate a private key in  $\mathcal{ABE}$  using this policy, as it should never learn what the  $\mathcal{I}_j$  are. One can imagine getting around this by using secure 2-party computation, but instead we use this requirement because it leads to an efficient realization when combined with OT.

Indeed, in our protocol definition, we will have the PKG generate a private key for the ABE scheme under a threshold policy for the entire set  $T_j$ : “The ciphertext contains at least  $\tau$  elements from  $T_j$ ”. If the decryption key decomposes in a natural way as required above, then we can use our OT protocol to let the user select the key components, thus giving him the desired key. We once again point out that the concrete ABE schemes used in this chapter trivially satisfy this requirement.

**Sanity Check.** We also require a notion of a “sanity check” for decryption keys and ciphertexts. The idea of using a formalized sanity check is also discussed in Au et al. [AHL08]. The reasoning for this requirement stems from the possibility that the PKG is now dishonest as well. Indeed, no formal definition of an ABE scheme prevents the scenario in which a dishonest PKG tricks an honest user into accepting a “bad” decryption key. For technical reasons in our proofs<sup>1</sup>, we require there exists two efficiently computable predicates `CiphertextSanityCheck` and `KeySanityCheck` to be associated with the ABE scheme  $\mathcal{ABE}$ . Given the public parameters and a ciphertext (resp. user key), the predicate indicates whether or not that ciphertext (resp. user key) is “sane”. These predicates should satisfy the properties:

---

<sup>1</sup>Loosely, we use this to argue that all sane keys behave identically from the point of view of a polynomially bounded adversary.

**Correctness** Honest executions of the algorithms in  $\mathcal{ABE}$  always result in sane ciphertexts and user keys, i.e.

$$\Pr[\text{CiphertextSanityCheck}(C, \text{PK}) = 1; \\ (\text{PK}, \text{MK}) \leftarrow \text{Setup}(\lambda), C \leftarrow \text{Encrypt}(\text{PK}, \gamma, M)] = 1$$

and

$$\Pr[\text{KeySanityCheck}(d_{\mathbb{A}}, \text{PK}) = 1; \\ (\text{PK}, \text{MK}) \leftarrow \text{Setup}(\lambda), d_{\mathbb{A}} \leftarrow \text{KeyGen}(\text{PK}, \text{MK}, \mathbb{A})] = 1.$$

**Soundness** If  $d, d'$  are sane keys, both of which contain policies that can decrypt a sane ciphertext  $C$ , then they must decrypt to the same message.

We mention that [AHL08] defines the notion of a sanity check game, which is similar to the soundness property above. In their weaker definition, it should only be infeasible for a polynomially bounded adversary to come up with two sane decryption keys that decrypt a ciphertext  $C$  in two different ways. As we shall see, the concrete ABE schemes used in this chapter all admit sanity checks which satisfy our stronger definition.

### 5.3.2 The Construction

Let  $\lambda$  represent a global security parameter throughout this construction.

In our construction, each user  $U$  is identified by her identity string  $\text{ID}$ . These are represented by  $\ell$ -bit strings where  $\ell$  is polynomial in  $\lambda$ . Let  $\mathcal{IBE} = (\text{IBE-Setup}, \text{IBE-KeyGen}, \text{IBE-Encrypt}, \text{IBE-Decrypt})$  be an IBE scheme,  $\mathcal{OT}$  be a  $k$ -out-of- $n$  oblivious transfer protocol, and  $\mathcal{ABE} = (\text{ABE-Setup}, \text{ABE-KeyGen}, \text{ABE-Encrypt}, \text{ABE-Decrypt})$  be an ABE scheme satisfying the requirements described in Section 5.3.1.

Let  $n$  and  $m$  be chosen as “deterrence” parameters: looking ahead, our proofs will show that a malicious PKG can only succeed with probability negligible in  $n$ . For the sake of our proofs, we set  $n$  to be equal to the global security parameter  $\lambda$  and  $m$  be super-logarithmic in  $n$ , e.g.  $m = \log^2(n)$ . We shall denote the sets  $\{1, \dots, \ell\}, \{1, \dots, n\}, \{1, \dots, m\}$  by  $[\ell], [n], [m]$ , respectively, and the  $i$ -th bit of the identity ID with  $ID_i$ . We fix a number of dummy attributes  $k$  that is a constant fraction of  $n$ , and a decryption threshold  $\tau$  as explained above (in the Appendix, we give an example using explicit values).

We also distinguish  $2\ell$  attributes and designate them as the “user attributes”  $u_{1,0}, u_{1,1}, u_{2,0}, u_{2,1}, \dots, u_{\ell,0}, u_{\ell,1}$ . The idea behind these user attributes is that if a user has identity ID then for  $i \in [\ell]$ , the attribute  $u_{i,ID_i}$  will be associated to the user. We also designate  $n \cdot m$  dummy attributes  $t_{i,j}$  with  $i \in [n], j \in [m]$ . We define  $T_j$  to be the set  $\{t_{i,j} | i \in [n]\}$ . Throughout our construction, we will use  $\mathcal{I}_j$ , which consists of  $k$  elements from 1 to  $n$ , to denote an indexing set of  $T_j$ . We also make the natural identification of  $\mathcal{I}_j$  as a subset of  $T_j$  when convenient. Then  $\mathcal{I}$  will denote the collection of all  $m$  of these indexing sets:  $\mathcal{I} = \{\mathcal{I}_j\}_{j \in [m]}$ . Finally, if  $\mathcal{I}$  and  $\mathcal{J}$  are two such collections, we define a relation  $\mathcal{R}(\mathcal{I}, \mathcal{J})$  which evaluates to 1 if for all  $j \in [m]$ ,  $|\mathcal{I}_j \cap \mathcal{J}_j| \geq \tau$ . The key to our construction is that  $k, n$ , and  $m$  have been chosen so that for a fixed  $\mathcal{I}$ ,  $Pr[\mathcal{R}(\mathcal{I}, \mathcal{J}) = 0]$  is negligible over a uniform choice of  $\mathcal{J}$ . Our main construction follows.

**Setup.** Run the setup algorithms of  $\mathcal{IBE}$  and  $\mathcal{ABE}$  to obtain  $(PK_{IBE}, MK_{IBE}) \leftarrow \text{IBE-Setup}$  and  $(PK_{ABE}, MK_{ABE}) \leftarrow \text{ABE-Setup}$ . Then (in the notation above) the public parameters are  $PK = (PK_{IBE}, PK_{ABE}, \{u_{i,j}\}, \{t_{i,j}\})$  and the master secret key is  $MK = (MK_{IBE}, MK_{ABE})$ .

**Key Generation Protocol.** The high level idea of our key generation protocol is to allow the user to obviously choose which dummy attributes he wants (using a  $k$ -

out-of- $n$  oblivious transfer) on each “repetition”. These repetitions are performed in parallel and will be viewed as individual components of our key. We want a policy that he can only decrypt when the ciphertext contains  $\tau$  of these attributes (for each component). Additional care needs to be taken to ensure the simulatability of this protocol (which is crucial to our security proofs) while still keeping it as efficient as possible. The interactive key generation protocol between PKG and a user  $U$  (with the identity ID) proceeds as follows.

1.  $U$  aborts if the published attributes  $\{u_{i,j}\}$  and  $\{t_{i,j}\}$  in the public key are not all different.
2. PKG runs IBE-KeyGen with ID and  $\text{MK}_{IBE}$  to obtain a decryption key  $d_{IBE}$ . This key is sent to  $U$ .
3. PKG sets  $u_i = u_{i,\text{ID}_i}$  for all  $i \in [\ell]$  and computes a decryption key by running ABE-KeyGen with the policy “The ciphertext contains all  $u_i$  and at least  $\tau$  elements from each  $T_j$ ”. PKG decomposes the decryption key into the components  $(d_0, \{d_{i,j}\}_{i \in [n], j \in [m]})$  and stores them.
4. PKG chooses random permutations  $\pi_1, \dots, \pi_m \in S_n$  where  $S_n$  is the set of permutations on  $n$  elements. Looking ahead, this step will help the simulator (in the proof of security) enforce a particular choice of the dummy attributes on him. We denote  $\pi = (\pi_1, \dots, \pi_m)$ .
5. PKG and  $U$  then engage in  $m$  executions of a  $k$ -out-of- $n$  oblivious transfer protocol where PKG acts as the sender and  $U$  acts as the receiver. In the  $j$ -th execution, the private input of PKG are the key components  $\{d_{\pi_j(i),j}\}_{i=1}^n$  and the private input of  $U$  is a set  $U_j$  of  $k$  randomly selected indices (from 1 to  $n$ ). The private output of  $U$  are the key components  $\{d_{\pi_j(i),j}\}_{i \in U_j}$ .

6. PKG sends  $U$  the permutation list  $\pi$  and  $d_0$ .
7. Finally,  $U$  sets  $\mathcal{I}_j = \pi_j(U_j)$  and  $d_{ABE} = (d_0, \{(\mathcal{I}_j, \{d_{i,j}\}_{i \in \mathcal{I}_j})\}_{j \in [m]})$  which is a decryption key for the policy “The ciphertext contains all  $u_i$  and at least  $\tau$  elements from each  $\mathcal{I}_j$ ” and checks that  $\text{KeySanityCheck}(\text{PK}, d_{ABE}) = 1$  for that policy.  $U$  aborts if the check fails. Finally,  $U$  sets the decryption key  $d_{\text{ID}} = (d_{IBE}, d_{ABE})$ .

**Encryption** To encrypt a message  $M$  under an identity  $\text{ID}$ , choose a  $\mathcal{J}$  uniformly at random. Namely, choose sets  $\mathcal{J}_j \subset [n]$  of size  $k$  for each  $j \in [m]$ . Let  $\gamma$  be the set of attributes  $\{u_{i, \text{ID}_i}\}_{i \in [\ell]} \cup \{t_{i,j}\}_{i \in \mathcal{J}_j, j \in [m]}$ . Compute the ciphertext  $C$  as follows.

Select a nonce  $R$  and run IBE-Encrypt with  $\text{PK}_{IBE}$  for identity  $\text{ID}$ , and message  $R$  to obtain the ciphertext  $C_{IBE}$ . Run ABE-Encrypt with  $\text{PK}_{ABE}$  and attributes  $\gamma$  for the message  $R \oplus M$  to obtain the ciphertext  $C_{ABE}$ .

Output the ciphertext  $C = (C_{IBE}, C_{ABE})$ .

**Decryption** To decrypt the ciphertext  $C = (C_{IBE}, C_{ABE})$  using the decryption key  $d_{\text{ID}} = (d_{IBE}, d_{ABE})$ , we first check that  $\text{CiphertextSanityCheck}(C_{ABE}) = 1$ . If the check fails, output  $\perp$ . Otherwise, let  $M_{IBE}$  and  $M_{ABE}$  be the respective messages recovered by decrypting each component. Output message  $M = M_{IBE} \oplus M_{ABE}$ .

**Trace** This algorithm takes an identity  $\text{ID}$ , a well-formed decryption key  $d_{\text{ID}}$  (with dummy attributes  $\mathcal{D}$ ) and a decoder box  $\text{D}$  which is  $\epsilon$ -useful. Our tracing algorithm will run in time polynomial in  $\lambda$  and  $\frac{1}{\epsilon}$ . The tracing algorithm will repeat the following experiment  $\eta = \frac{24m\lambda}{\epsilon}$  times:

1. Choose a random message  $M$  and random  $\mathcal{J}$  subject to the constraint that  $\mathcal{R}(\mathcal{I}, \mathcal{J}) = 0$ .
2. Encrypt  $M$  using the attributes  $\mathcal{J}$  to obtain a ciphertext  $C$ .

3. Attempt to decrypt  $C$  using the decoder box.

If  $D$  ever correctly decrypted a ciphertext, then the algorithm implicates the PKG by returning PKG, otherwise it returns User. In the next section, we show that the above simple tracing mechanism works except with negligible probability even though the ciphertexts on which we probe the box are coming from a specific distribution (rather than simply being random ciphertexts for the given identity).

This completes the construction of our generic A-IBE scheme, which we will henceforth denote as  $\mathcal{AIBE}\text{-GEN}$ . This construction is summarized as follows:

**Global:**  $\lambda, \epsilon, \ell, k, n, m$  which are all polynomially related. Identities ID are represented by  $\ell$ -bit strings. Let  $\mathcal{IBE} = (\text{IBE-Setup}, \text{IBE-KeyGen}, \text{IBE-Encrypt}, \text{IBE-Decrypt})$  be an IBE scheme,  $\mathcal{OT}$  be a  $k$ -out-of- $n$  oblivious transfer protocol, and  $\mathcal{ABE} = (\text{ABE-Setup}, \text{ABE-KeyGen}, \text{ABE-Encrypt}, \text{ABE-Decrypt})$  be an ABE scheme satisfying the requirements described in Section 5.3.1.

**Setup:**  $\text{Setup} (\text{PK}_{\text{IBE}}, \text{MK}_{\text{IBE}}) \leftarrow \text{IBE-Setup}$  and  $(\text{PK}_{\text{ABE}}, \text{MK}_{\text{ABE}}) \leftarrow \text{ABE-Setup}$ . Designate  $2\ell + n \cdot m$  special attributes  $\{u_{i,j}\}_{i \in [\ell], j=0,1}, \{t_{i,j}\}_{i \in [n], j \in [m]}$ . The public parameters are  $\text{PK} = (\text{PK}_{\text{IBE}}, \text{PK}_{\text{ABE}}, \{u_{i,j}\}, \{t_{i,j}\})$  and the master secret key is  $\text{MK} = (\text{MK}_{\text{IBE}}, \text{MK}_{\text{ABE}})$ .

**Key Generation:** The user and PKG engage in an interactive protocol. At the end of this protocol, the user should have  $m$  dummy attribute sets  $\mathcal{I}_j$ , each of size  $k$ , which are supposedly oblivious from the PKG. The user ID should have a decryption key  $d_{\text{IBE}}$ , which is a private key for ID in  $\mathcal{IBE}$ . The user should also have  $d_{\text{ABE}}$ , which is a decryption key in  $\mathcal{ABE}$  for the policy “The ciphertext contains all  $u_{i,\text{ID}_i}$  and at least  $\tau$  elements from each  $\mathcal{I}_j$ ”.

**Encryption:** To encrypt a message for ID, select a nonce  $R$  and encrypt it in  $\mathcal{IBE}$  to obtain  $C_{\text{IBE}}$ . Choose  $m$  dummy attribute sets  $\mathcal{J}_j$ , each of size  $k$ . Set  $\gamma =$

$\{u_{1, \text{ID}_1}, \dots, u_{\ell, \text{ID}_\ell}\} \cup \bigcup \mathcal{J}_j$ . Encrypt  $R \oplus M$  in  $\mathcal{ABE}$  under the attributes  $\gamma$  to obtain  $C_{\text{ABE}}$ .

Output the ciphertext  $C = (C_{\text{IBE}}, C_{\text{ABE}})$ .

**Decryption:** We decrypt each half of the ciphertext in the appropriate scheme to obtain  $M_{\text{IBE}}, M_{\text{ABE}}$ . Output  $M = M_{\text{IBE}} \oplus M_{\text{ABE}}$ .

**Trace:** Given an identity  $\text{ID}$ , a key  $d_{\text{ID}}$  and a decoder box  $D$  which is  $\epsilon$ -useful, the idea is to repeat the following experiment  $\eta = \frac{24m\lambda}{\epsilon}$  times: Create a ciphertext that  $d_{\text{ID}}$  cannot decrypt (due to the lack of attributes in the ciphertext) and attempt to decrypt it with  $D$ . If  $D$  ever properly decrypts, then output  $\text{PKG}$ , otherwise output  $\text{User}$ .

## 5.4 Security Proofs

In this section, we show that the generic construction  $\mathcal{AIBE}\text{-}\mathcal{GEN}$  given in the previous section is a secure, assuming the security of its building blocks. We first make the observation that if the underlying IBE scheme is  $\text{IND-ID-CPA}$  secure, then so is  $\mathcal{AIBE}\text{-}\mathcal{GEN}$ :

**Theorem 5.4.1.** *The advantage of an adversary in the  $\text{IND-ID-CPA}$  game is negligible for  $\mathcal{AIBE}\text{-}\mathcal{GEN}$  assuming the underlying IBE scheme is  $\text{IND-ID-CPA}$  secure.*

Given an adversary to break the  $\text{IND-ID-CPA}$  security of our construction, it is straightforward to construct an adversary to break the  $\text{IND-ID-CPA}$  security of underlying IBE scheme. For the remainder of the section, we will only focus on the ABE portion of the scheme by suppressing the IBE portion in our ciphertext and keys.

### 5.4.1 Dishonest PKG Game

In this subsection, we show our construction is secure relative to the DishonestPKG game. This is formulated as the following theorem.

**Theorem 5.4.2.** *Assuming that the underlying OT is fully simulatable (secure as per the ideal/real world security definition [Can00]), the advantage of any adversary in the DishonestPKG game is negligible for  $\mathcal{AIBE}\text{-GEN}$ .*

**Conceptual Argument:** We start off by considering the scheme where the OT in the key generation protocol is replaced by an ideal OT functionality. In this hybrid-OT world, the dummy attributes selected by the user are information theoretically hidden from the PKG's view at the end of the key generation protocol. In our proof, we will use the composition theorem of Canetti [Can00] to transform a real-world adversary into this one which operates in the hybrid-OT model. For our conceptual argument, we will just consider adversaries which operate in this hybrid-OT world.

The first phase of the DishonestPKG game is to set up the A-IBE scheme and run the key generation protocol with the challenger. The only reply the adversary receives in this phase is whether or not it aborted. The adversary can attempt to learn information about the dummy attributes by somehow choosing invalid values to test whether or not the challenger aborts. One can imagine this malicious behavior as setting a dial where, on one end, it almost never aborts, and on the other end, it almost always aborts. After the key generation phase, if the challenger did not abort, then the adversary learns a little or a lot of information depending on the setting of the dial. Of course, for the adversary to succeed in the DishonestPKG game, it cannot cause the user to abort too often, and so intuitively it cannot learn too much information about the dummy attributes. In our proof, we formalize this relationship between the success rate of the adversary and the amount of information learned about the dummy



attributes in Lemma 3.

The next phase in the DishonestPKG game is the decryption queries phase. Loosely speaking, because any valid decryption key can decrypt all but a negligible fraction of valid ciphertexts, there will almost never be an anomalous query that can't be decrypted by the challenger's decryption key. By the soundness of the sanity checks, we are guaranteed that the same message is recovered by *every* key that satisfies the policy (one of which will be the valid key that the user actually holds). Thus, the adversary can only learn information in the negligible fraction of cases when the policy held by the challenger is not satisfied by the dummy attributes in the ciphertext.

Finally, the adversary outputs a decoder box  $D$ . Because the adversary has learned almost nothing about the challenger's dummy attributes up to this point, we can imagine this box as being created almost independent of these attributes. It remains to show that no box can be "blindly" created to implicate a non-negligible fraction of possible dummy attribute choices. This will be proven combinatorially in Lemma 6. Thus, the adversary is doomed to failure: it must create a decoder box, which can only implicate a negligible fraction of all valid decryption keys, but only knows that the challenger's decryption key lies in some non-negligible space.

**Proof Organization:** We organize our proof as follows. We assume toward a contradiction that there exists an adversary  $\mathcal{A}_0$  that wins the DishonestPKG game with non-negligible success probability. We use the composition theorem of Canetti [Can00] (Lemma 2) to argue that there exists an adversary  $\mathcal{A}$  that succeeds in the hybrid-OT world with non-negligible probability  $\delta$  against a challenger  $\text{Ch}$ .

Next, we define the event  $\mathcal{E}_1$  to be  $\Pr[\text{Ch finishes KeyGen}] \geq \delta/2$  where the underlying variable is the random tape of the adversary. In terms of the intuition given above, this is analogous to saying the adversary did not "set the dial too low". We show

in Lemma 3 that  $\mathcal{E}_1$  occurs with probability at least  $\delta/2$ . We then define the event  $\mathcal{E}_2$  to be that the key generation phase was not aborted and focus only on the cases where  $\mathcal{E}_1 \wedge \mathcal{E}_2$  occur.

After the key generation phase, the challenger holds a key that passes the decryption key sanity check. In Lemma 4, we show that the views of the adversary when playing against *any* challenger whose decryption key can decrypt a valid ciphertext are all identical. We define  $\mathcal{E}_3$  to be the event that the adversary did not ask an “anomalous” query, i.e. all queries were either invalid or decryptable by the challenger’s key. Thus, if  $\mathcal{E}_3$  occurs, the only information the adversary learns from the conversation is that the challenger’s decryption key did not come from the negligible fraction of keys that cannot decrypt one of the ciphertext queries. We conclude in Lemma 5 that  $\neg\mathcal{E}_3$  only occurs with negligible probability when conditioned on  $\mathcal{E}_1 \wedge \mathcal{E}_2$ .

Conditioned on the events  $\mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3$ , we show that the only information the adversary has about the challenger’s dummy attributes is that it comes from a space consisting of at least a  $\approx \delta/2$  fraction of all possible dummy attributes. Furthermore, in Lemma 6, we show that any decoder box  $D$  can implicate only negligibly many user keys. These two facts combined mean the advantage of the adversary is only negligible, a contradiction. Our detailed proof follows.

*Proof of Theorem 5.4.2.* Assume toward a contradiction that an adversary  $\mathcal{A}_0$  has some non-negligible probability  $\varepsilon$  of success. We work to eventually contradict a combinatorial lemma (Lemma 6). We begin by describing two experiments  $\text{Expt}_0$  and  $\text{Expt}_1$ .

**Expt<sub>0</sub>( $\mathcal{A}$ ):** In this experiment, the adversary  $\mathcal{A}$  plays the DishonestPKG game against a challenger under the  $\mathcal{ATBE}\text{-}\mathcal{GEN}$  scheme.

**Expt<sub>1</sub>( $\mathcal{A}$ ):** We create the hybrid scheme  $\mathcal{ATBE}\text{-}\mathcal{GEN}'$  where the real OT protocol in the key generation is replaced by an ideal OT functionality. In this experiment,

the adversary  $\mathcal{A}$  plays the DishonestPKG game against a challenger under the hybrid  $\mathcal{ATBE}\text{-}\mathcal{GEN}'$  scheme.

By the Composition theorem [Can00], we have the following lemma:

**Lemma 2.** [Composition theorem (Canetti [Can00])] For every adversary  $\mathcal{A}_0$  that succeeds with probability  $\varepsilon$  in  $\text{Expt}_0$ , there exists an adversary  $\mathcal{A}$  that succeeds in  $\text{Expt}_1$  with probability  $\delta$  where  $|\varepsilon - \delta| < \nu_1$ , which is negligible.

Let **SUCC** be the event that the adversary  $\mathcal{A}$  succeeds in this game. Let  $r_{\mathcal{A}}$  denote the randomness for this adversary and  $r_C$  denote the randomness for the challenger. Recall that the only randomness used by the challenger is during the key generation protocol where it selects a set of indices (which correspond to dummy attributes). We henceforth identify  $r_C$  as also being a set of dummy attributes  $\{\mathcal{I}_j\}$ . Let  $\mathcal{E}_1$  be the event that the execution of the adversary does not cause an abort in the key generation phase with probability at least  $\delta/2$ . That is to say,  $\mathcal{E}_1$  holds for the set of  $r_{\mathcal{A}}$  on which  $\Pr[\text{Ch finishes KeyGen}] \geq \delta/2$  where the probability is taken over the randomness of the challenger.

**Lemma 3.** The probability that event  $\mathcal{E}_1$  occurs is at least  $\delta/2$ .

*Proof of Lemma 3:* Let  $p$  be the probability that event  $\mathcal{E}_1$  occurs. Observe that when  $\mathcal{E}_1$  does not occur,  $\mathcal{A}$  has at most a  $\delta/2$  chance of success due to the fact that the challenger will abort in the key generation phase with at least a  $1 - \delta/2$  probability. Thus, by the Markov bound, it follows that  $p > \delta/2$ .  $\square$

We now focus on the executions for which  $\mathcal{E}_1$  occurs. The expected success probability of the adversary must still be at least  $\delta$  because every execution where  $\mathcal{E}_1$  does not occur does not fail in the key generation phase with probability at least  $\delta/2$ . The challenger selects dummy attributes uniformly at random in the key generation protocol, which implies at least a  $\delta/2$  fraction of all dummy attribute sets will lead to the

challenger receiving a well-formed decryption key. We now argue that even after the decryption query phase, there are still too many possible choices of dummy attributes for the adversary's decoder box to succeed against a non-negligible fraction of them.

Let  $\mathcal{E}_2$  be the event that the challenger does not abort in the key generation phase. Indeed, the success probability of the adversary can only increase if we condition on  $\mathcal{E}_2$  occurring: any time  $\mathcal{E}_2$  does not occur, the adversary immediately loses. Now focus only on the executions where  $\mathcal{E}_1$  and  $\mathcal{E}_2$  occur. Since the challenger did not abort in the key generation phase, it now has a well-formed decryption key which passes the sanity check. If the challenger has the dummy attribute collection  $\mathcal{I}$ , then let  $\nu_2$  denote the negligible probability that  $\mathcal{R}(\mathcal{I}, \mathcal{J}) = 0$  over a random choice of  $\mathcal{J}$ .

Let  $\mathcal{E}_3$  be the event that all well-formed ciphertexts are properly decrypted (i.e. the challenger does not fail on any query to decrypt due to intersection of dummy attributes). We now analyze the probability of this event occurring and how it affects the view of the adversary. Let  $\nu_3$  be the probability that  $\mathcal{E}_3$  does not occur given  $\mathcal{E}_1 \wedge \mathcal{E}_2$ . We shall argue that  $\nu_3$  is negligible.

To prove this, we stratify  $\mathcal{E}_3$  as the conjunction of the events “Ch did not fail on query  $i$ ” for  $i = 1, \dots, q$ . Let  $\mathcal{I}$  be the dummy attributes that the challenger holds after the key generation phase, and let  $\mathcal{J}^i$  be the attributes in the  $i$ -th ciphertext query. We define  $\text{GOOD}_i$  to be the event that either the ciphertext in query  $i$  is malformed or  $\mathcal{R}(\mathcal{I}, \mathcal{J}^i) = 1$ . Define  $\mathcal{F}_i = \text{GOOD}_1 \wedge \dots \wedge \text{GOOD}_i$ . We state a lemma about the view of the adversary.

**Lemma 4.** Fix a random tape  $r_{\mathcal{A}}$  of the adversary such that  $\mathcal{E}_1$  occurs (recall that  $\mathcal{E}_1$  is independent of the challenger). Fix a query number  $1 \leq i \leq q$ . Let  $r_C, r'_C$  be arbitrary elements from the set  $\{r_C : \mathcal{E}_2 \wedge \mathcal{F}_{i-1} \text{ holds}\}$ . Before query  $i$  is made, the view of the adversary in the execution where Ch uses  $r_C$  as its random tape is *identical* to the view in the execution where Ch uses  $r'_C$  as its random tape.

In particular, the queries made by the adversary in either execution are identical.

*Proof.* We prove this by induction on  $i$ . When  $i = 1$  (after the key generation phase, but before the first query is made), the adversary learns only whether or not the challenger aborted. Up to this point, because we are in the ideal OT world, this is the only information the adversary learns. Since both  $r_C$  and  $r'_C$  entail the event  $\mathcal{E}_2$ , the view of the adversary is identical in both cases: the challenger did not abort and received a well-formed key.

Now assume that the view of the adversary where Ch uses  $r_C$  is identical to the view of the adversary where Ch uses  $r'_C$  for every query before  $i$ . Because the adversary has the same view in both cases and we fixed the random tape  $r_A$ , the  $i$ -th query is  $C_i$  in either case. Now, if  $C_i$  is malformed, the challenger will return  $\perp$  (regardless of any randomness). On the other hand, if  $C_i$  passes the ciphertext sanity check, then  $\text{GOOD}_j$  guarantees that the policy is satisfied, and so by the soundness of the sanity check, the challenger will reply with the same answer whether  $r_C$  or  $r'_C$  was used.  $\square$

**Lemma 5.** Fix a random tape  $r_A$  of the adversary such that  $\mathcal{E}_1$  occurs. Then  $Pr_{r_C}[\neg\mathcal{E}_3|\mathcal{E}_2] \leq \frac{2q\nu_2}{\delta/2}$ . We define the negligible quantity on the right hand side to be  $\nu_3$ .

*Proof.* Let  $p_2$  be the probability that event  $\mathcal{E}_2$  occurs. Because we fixed an execution where  $\mathcal{E}_1$  occurs, we have that  $p_2 \geq \delta/2$ . We shall prove inductively that  $Pr[\mathcal{E}_2 \wedge \mathcal{F}_i] \geq p_2 - i\nu_2$ .

By Lemma 4, before the first ciphertext query the adversary has no information about  $r_C$  other than  $\mathcal{E}_2$  occurred. Hence the first ciphertext is independent of any  $r_C$  for which  $\mathcal{E}_2$  holds. Recall that for any ciphertext,  $\nu_2$  is the negligible fraction of user keys that cannot decrypt it. The probability that a uniformly selected  $r_C$  conditioned on  $\mathcal{E}_2$  will fail on the first ciphertext query is  $Pr[\neg\text{GOOD}_1|\mathcal{E}_2] \leq \frac{\nu_2}{p_2}$ . Thus

$Pr[\text{GOOD}_1|\mathcal{E}_2] \geq 1 - \frac{\nu_2}{p_2}$  and so there is at least a  $(1 - \frac{\nu_2}{p_2}) \cdot p_2 = p_2 - \nu_2$  fraction of the random tapes remaining that satisfy  $\mathcal{E}_2 \wedge \text{GOOD}_1$ .

On the  $i$ -th query, the queried ciphertext once again cannot be decrypted by a  $\nu_2$  fraction of all possible  $r_C$ 's. In the worst case, this fraction is disjoint from the ones excised by the first  $i - 1$  queries. By Lemma 4, the  $i$ -th query is independent of any  $r_C$  for which  $\mathcal{E}_2 \wedge \mathcal{F}_{i-1}$  hold. By induction, this accounts for at least a  $p_2 - (i - 1)\nu_2$  fraction of all possible  $r_C$ 's. The probability that a uniformly selected  $r_C$  conditioned on  $\mathcal{E}_2 \wedge \mathcal{F}_{i-1}$  will fail to decrypt the  $i$ -th ciphertext query is  $Pr[\neg\text{GOOD}_i|\mathcal{E}_2 \wedge \mathcal{F}_{i-1}] \leq \frac{\nu_2}{p_2 - (i-1)\nu_2}$ . Consequently, we calculate that  $Pr[\mathcal{E}_2 \wedge \mathcal{F}_i]$  is at least  $p_2 - i\nu_2$ .

Eventually, after  $q$  queries, we have  $Pr[\mathcal{E}_2 \wedge \mathcal{E}_3] = Pr[\mathcal{E}_2 \wedge \mathcal{F}_q]$  is at least  $p_2 - q\nu_2$ . So  $Pr[\mathcal{E}_3|\mathcal{E}_2] \geq 1 - \frac{q\nu_2}{p_2} \geq 1 - \frac{2q\nu_2}{\delta}$ , from which the lemma follows.  $\square$

Finally, the adversary must output a decoder box  $D$ . We show that *any* decoder box can implicate the user in only a negligible fraction of dummy attribute sets. We denote this negligible quantity by  $\nu_4$ . Our main lemma, which we prove later, is as follows:

**Lemma 6.** Fix the public parameters  $PK$  and an identity  $ID$ . Let  $\epsilon = \frac{1}{\text{poly}(\lambda)}$  and  $D$  be an  $\epsilon$ -useful decoder box. If  $\mathcal{I}$  is a dummy attribute set for the user, we consider the following experiment:

- Select a dummy attribute collection  $\mathcal{J}$  at random subject to the condition  $\mathcal{R}(\mathcal{I}, \mathcal{J}) = 0$ .
- Select a random message  $M$  and encrypt  $M$  using  $\mathcal{J}$  as the dummy attributes.
- The decoder box outputs some  $M' = D(C)$ .

Define the event  $D\text{Box}$  to hold when  $M' = M$ . The lemma states that for all but a negligible fraction,  $\nu_4$ , of choices for  $\mathcal{I}$  we have

$$Pr[D\text{Box}] > \frac{\epsilon}{24m}$$

In particular, the tracing algorithm will (with overwhelming probability) implicate the PKG for all but a negligible fraction of choices of dummy attributes.

Assuming the lemma above, we continue our proof by contradiction. By Lemma 4, the view of the adversary after events  $\mathcal{E}_2$  and  $\mathcal{E}_3$  will be identical for all of the remaining  $(\delta/2) - \nu_3$  fraction of  $r_C$ 's. Thus, the adversary creates this box independent of  $r_C$  other than the fact that  $\mathcal{E}_2 \wedge \mathcal{E}_3$  hold. Because any of these dummy attribute sets remain equally likely, the probability that D succeeds (i.e. the tracing algorithm on box D implicates the user) is at most  $\frac{\nu_4}{(\delta/2) - \nu_3}$ . We summarize this contradiction in the following equations:

$$\begin{aligned}
\delta &= Pr[\text{SUCC}] \\
&\leq Pr[\text{SUCC} | \mathcal{E}_1 \wedge \mathcal{E}_2] \\
&= Pr[\text{SUCC} | \mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \mathcal{E}_3] Pr[\mathcal{E}_3 | \mathcal{E}_1 \wedge \mathcal{E}_2] \\
&\quad + Pr[\text{SUCC} | \mathcal{E}_1 \wedge \mathcal{E}_2 \wedge \neg \mathcal{E}_3] Pr[\neg \mathcal{E}_3 | \mathcal{E}_1 \wedge \mathcal{E}_2] \\
&\leq \frac{\nu_4}{(\delta/2) - \nu_3} \cdot 1 \\
&\quad + 1 \cdot \nu_3 \\
&= \text{negl.}
\end{aligned}$$

This concludes the proof of Theorem 5.4.2. □

We now prove the main lemma (Lemma 6):

*Proof of Lemma 6:* For the purposes of this proof, we fix an identity ID and ignore all portions of the decryption key except for the dummy attributes contained in it:  $\mathcal{I} = \{\mathcal{I}_j\}_{j \in [m]}$ . Similarly, we ignore all portions of the ciphertext except which dummy attributes are contained in it:  $\mathcal{J} = \{\mathcal{J}_j\}_{j \in [m]}$ . We will refer to  $\mathcal{I}_j$  (resp.  $\mathcal{J}_j$ ) as the  $j$ -th component or index of the dummy attributes in a user key (resp. ciphertext). We can imagine both  $\mathcal{I}$  and  $\mathcal{J}$  as being subsets of the same universe  $\mathcal{K}$  which contains all  $m$ -

tuples of  $k$ -element sets. We simply refer to these dummy attribute collections as the “user set” and the “ciphertext set”.

Recall that a user can decrypt if and only if each component in the intersection between the user set and the ciphertext set contains at least  $\tau$  elements. Let  $\mathbf{D}$  be a decryption box which we fix to be  $\epsilon$ -useful. For each ciphertext set  $\mathcal{J}$  we can have some probability  $p_{\mathcal{J}}$  that the box will decrypt on it <sup>2</sup>.

Consider how one randomly samples ciphertexts which cannot be decrypted by the user. We may think of choosing ciphertext set that intersects (with the user set) on the  $j$ -th component  $\mathcal{I}_j$  by less than  $\tau$  attributes by first choosing a set of  $\beta = 1 + k - \tau$  attributes disjoint from  $\mathcal{I}_j$ , then selecting the remaining  $k - \beta$  attributes at random. Because these  $\beta$ -element sets of attributes will be important for us, it is useful to think of any arbitrary  $\beta$ -element subset (of  $\{1, \dots, n\}$ ) as an atomic object, which we will call a *bundle*. To clarify the description, every set of  $\beta$  attributes on the  $j$ -th component is a  $j$ -*bundle*. Let  $B_j$  be the set of all  $j$ -bundles. When sampling random ciphertexts which cannot be decrypted by the user, the idea is to select a bundle which avoids the user set on some component, then select a ciphertext set which contains that bundle. For each bundle  $b \in B_j$ , we can associate to it a set of ciphertexts whose attribute set contains it:  $\mathcal{K}_b^j := \{\mathcal{J} = (\mathcal{J}_1, \dots, \mathcal{J}_m) | b \subseteq \mathcal{J}_j\}$ . Conversely, for each ciphertext  $\mathcal{J} = (\mathcal{J}_1, \dots, \mathcal{J}_m)$  we can associate to it a set of  $j$ -bundles which it contains:  $\mathcal{B}_{\mathcal{J}}^j := \{b \in B_j | b \subseteq \mathcal{J}_j\}$ . We may similarly define sets for users:  $\mathcal{V}_b^j := \{\mathcal{I} = (\mathcal{I}_1, \dots, \mathcal{I}_m) | b \cap \mathcal{I}_j = \emptyset\}$  and  $\mathcal{A}_{\mathcal{I}}^j := \{b \in B_j | b \cap \mathcal{I}_j = \emptyset\}$  (users that avoid a bundle, and bundles that avoid a user, respectively).

We first make the observation that by symmetry, the size of the sets  $\mathcal{K}_b^j$  (as well as the sets  $\mathcal{B}_{\mathcal{J}}^j, \mathcal{V}_b^j, \mathcal{A}_{\mathcal{I}}^j$ ) are independent of  $b, j, \mathcal{J}$ , and  $\mathcal{I}$ . Thus, we may speak of the value  $|\mathcal{K}_b^j|$  even outside the scope of a defined  $b$  or  $j$ . We define the set of all bundles to

---

<sup>2</sup>Note that since we ignore the message, this is taken over the randomness used in the encryption except for the selection of the ciphertext set



be  $B := \bigcup_{j \in [m]} B_j$  and we can similarly define the set of all bundles contained in (resp. avoided by) a ciphertext (resp. a user) as  $\mathcal{B}_{\mathcal{J}} := \bigcup_{j \in [m]} B_{\mathcal{J}}^j$  (resp.  $\mathcal{A}_{\mathcal{I}} = \bigcup_{j \in [m]} A_{\mathcal{I}}^j$ ). By symmetry, selecting a ciphertext set  $\mathcal{J} \in \mathcal{K}$  uniformly at random then selecting a bundle it contains  $b \in \mathcal{B}_{\mathcal{J}}$  uniformly at random generates the same distribution on pairs  $(\mathcal{J}, b)$  as selecting a bundle  $b \in \mathcal{B}$  at random (say it is a  $j$ -bundle) followed by selecting a ciphertext set containing it  $\mathcal{J} \in \mathcal{K}_b^j$  uniformly at random. This can be combinatorially written as  $|\mathcal{K}| \cdot |\mathcal{B}_{\mathcal{J}}| = |\mathcal{B}| \cdot |\mathcal{K}_b^j|$ .

**Definition 2.** Let  $b \in B_j$  be a ( $j$ -)bundle. Define  $p_b^j := \frac{\sum_{\mathcal{J} \in \mathcal{K}_b^j} p_{\mathcal{J}}}{|\mathcal{K}_b^j|}$ , which is the average probability that a randomly selected ciphertext set containing this bundle is decrypted by D. A bundle  $b \in B_j$  is said to be *heavy on  $j$*  if  $p_b^j > \frac{\epsilon}{6m}$ . We define  $\mathcal{L}_j$  to be the set of bundles which are light (not heavy) on  $j$ , and  $\mathcal{L} = \bigcup_{j \in [m]} \mathcal{L}_j$ .

We first show a combinatorial lemma.

**Lemma 7 (Marking Lemma).** Consider an arbitrary marking on bundles where over  $\frac{1}{2}$  the bundles in each  $B_j$  for each  $j = 1, \dots, m/2$  are marked. Then with probability at least  $1 - \left(\frac{3}{4}\right)^{m/2}$  a randomly sampled ciphertext set  $\mathcal{J}$  will have the property that for some  $j$ , a  $\frac{1}{3}$  fraction of  $\mathcal{B}_{\mathcal{J}}^j$  will be marked.

*Proof.* Consider any fixed  $j$  between 1 and  $m/2$ . For a ciphertext set  $\mathcal{J}$  we define a predicate  $MARK_j(\mathcal{J})$  to hold true if at least a  $\frac{1}{3}$  fraction of  $\mathcal{B}_{\mathcal{J}}^j$  is marked. Let  $p$  be the probability over a randomly chosen  $\mathcal{J}$  that  $MARK_j(\mathcal{J}) = 1$ . Then we have

$$\begin{aligned}
\frac{1}{2} &\leq Pr_{\mathcal{J}, b \leftarrow \mathcal{B}_{\mathcal{J}}^j} [b \text{ is marked}] \\
&\leq Pr[b \text{ is marked} | MARK_j(\mathcal{J}) = 1] \cdot Pr[MARK_j(\mathcal{J}) = 1] \\
&\quad + Pr[b \text{ is marked} | MARK_j(\mathcal{J}) = 0] \cdot Pr[MARK_j(\mathcal{J}) = 0] \\
&\leq 1 \cdot p + \frac{1}{3} \cdot (1 - p)
\end{aligned}$$

Solving for  $p$  we obtain  $p \geq \frac{1}{4}$ . Since  $j$  was arbitrary, the probability that no  $j$  from 1 to  $m/2$  have  $MARK_j$  is  $(\frac{3}{4})^{m/2}$  which is negligible in  $n$  (since  $m$  was chosen to be super-logarithmic in  $n$ ).  $\square$

**Claim 1.** Let  $D$  be an  $\epsilon$ -useful decoder box. Either (1) on over half the components, more than half the bundles are heavy (on those components) or (2) on at least half the components, at least half the bundles are light. We claim that Case (1) implies Lemma 6, our main lemma. We further claim Case (2) will contradict the usefulness of  $D$ .

*Proof. Case (1):* One can view the process of sampling a random  $\mathcal{J}$  subject to  $\mathcal{R}(\mathcal{I}, \mathcal{J}) = 0$  as:

1. Selecting a random component  $j \in [m]$
2. Selecting a bundle  $b \in \mathcal{A}_T^j$  in avoiding the user on that component uniformly at random
3. Selecting a ciphertext set  $\mathcal{J} \in \mathcal{K}_b^j$  that contains that bundle uniformly at random

Then we see that there is a  $\frac{1}{2}$  probability of selecting a component with over half the bundles heavy in the first step, a  $\frac{1}{2}$  probability that the bundle selected in the second step is heavy, and finally by the definition of heavy, the ciphertext selected in the third step will be decrypted by  $D$  with at least a  $\frac{\epsilon}{6m}$  probability. Combined, this gives a probability of  $\epsilon/24m$  as claimed in Lemma 6. Since the tracing algorithm repeats this experiment  $\eta = \frac{24m\lambda}{\epsilon}$  times, the PKG will be implicated with all but an exponentially small probability.

**Case (2):** WLOG we may assume the first  $m/2$  components have more than half the bundles light. By the marking lemma, if we mark the light bundles, we know that with all but a negligible probability, a randomly sampled ciphertext will have at least  $\frac{1}{3}$  fraction marked (i.e. light) bundles on some component. Define the predicate

$LIGHT_j(\mathcal{J})$  to be true if there are at least  $\frac{1}{3}$  fraction of light bundles on the  $j$ -th component. We focus only on the ciphertext sets which has  $LIGHT_j(\mathcal{J}) = 1$  for some  $j$  (only negligibly many do not have this property). Consider the space of all pairs  $(\mathcal{J}, b)$  where  $\mathcal{J}$  is a ciphertext set which contains a light  $j$ -bundle  $b$  (for some  $j$ ). We will define two probability distributions  $\mathcal{D}_1$  and  $\mathcal{D}_2$  on this space. The first distribution is:

1. Select a random  $\mathcal{J}$ .
2. For every component of  $\mathcal{J}$ , place all light bundles it contains into a set  $S_{\mathcal{J}}$ . In other words, define  $S_{\mathcal{J}} := \bigcup_{j \in [m]} (\mathcal{B}_{\mathcal{J}}^j \cap \mathcal{L}_j)$ .
3. Select a random bundle  $b \in S_{\mathcal{J}}$ .

The second distribution is:

1. Select a random light bundle  $b \in \mathcal{L}$ .
2. Select a random  $\mathcal{J} \in K_b^j$ .

Observe that if we fix some  $(\mathcal{J}', b')$ , the probability that  $\mathcal{D}_1$  selects it is  $\frac{1}{|\mathcal{K}|} \cdot \frac{1}{|S_{\mathcal{J}'|}$ . As mentioned above, at least a third of all the bundles it contains (on some component) are light, and because each component contains the same number of bundles, at least  $\frac{1}{3m}$  total bundles it contains will be light. Thus, the probability that  $\mathcal{D}_1$  selects  $(\mathcal{J}', b')$  lies between  $\frac{1}{|\mathcal{K}|} \cdot \frac{3m}{|\mathcal{B}_{\mathcal{J}'|}$  and  $\frac{1}{|\mathcal{K}|} \cdot \frac{1}{|\mathcal{B}_{\mathcal{J}'|}$ .

For the distribution  $\mathcal{D}_2$ , it will select  $(\mathcal{J}', b')$  with probability  $\frac{1}{|\mathcal{L}|} \cdot \frac{1}{|\mathcal{K}_b^j|}$ . By assumption, there are at least half the bundles light on the first half of the components, so overall, the light bundles make up over a quarter fraction of all bundles. Thus this probability is between  $\frac{4}{|\mathcal{B}|} \cdot \frac{1}{|\mathcal{K}_b^j|}$  and  $\frac{1}{|\mathcal{B}|} \cdot \frac{1}{|\mathcal{K}_b^j|}$ .

By the earlier observation that  $|\mathcal{K}| \cdot |\mathcal{B}_{\mathcal{J}}| = |\mathcal{B}| \cdot |\mathcal{K}_b^j|$ , the probability that a ciphertext set is selected in  $\mathcal{D}_1$  is at most  $3m$  times as likely as that ciphertext set is selected in  $\mathcal{D}_2$ . However, the probability that  $\mathsf{D}$  decrypts a ciphertext containing a ciphertext set sampled from the first distribution is  $\epsilon$ , while by definition of “light”, the probability that  $\mathsf{D}$  decrypts a ciphertext containing a ciphertext set sampled from the second distribution is at most  $\frac{\epsilon}{6m}$ . Then we have

$$\begin{aligned}
\epsilon &= \sum_{(\mathcal{J},b)} ((\mathcal{J},b) \text{ is sampled by } \mathcal{D}_1) \cdot p_{\mathcal{J}} \\
&\leq \sum_{(\mathcal{J},b)} 3m \cdot ((\mathcal{J},b) \text{ is sampled by } \mathcal{D}_2) \cdot p_{\mathcal{J}} \\
&\leq 3m \cdot \sum_{(\mathcal{J},b)} ((\mathcal{J},b) \text{ is sampled by } \mathcal{D}_2) \cdot p_{\mathcal{J}} \\
&\leq 3m \cdot \frac{\epsilon}{6m}
\end{aligned}$$

which is a contradiction. □

## 5.4.2 Dishonest User Game

We now prove  $\mathcal{AI}\mathcal{BE}\text{-}\mathcal{G}\mathcal{EN}$  is secure relative to the Selective-ID DishonestUser game assuming the underlying ABE scheme is Selective-Set secure. To accomplish this, we construct a security reduction.

**Theorem 5.4.3.** *Assuming that the underlying OT is fully simulatable (secure as per the ideal/real world security definition [Can00]) and the underlying ABE scheme is Selective-Set secure, the advantage of any adversary in the Selective-ID DishonestUser game is negligible for  $\mathcal{AI}\mathcal{BE}\text{-}\mathcal{G}\mathcal{EN}$ .*

**Conceptual Argument:** As before, we focus on the hybrid-OT model where the real OT protocol in the key generation protocol is replaced by an ideal functionality. Again,

we use the composition theorem of Canetti [Can00] to convert an adversary in the real world into this model. Given an adversary for the Selective-ID DishonestUser game, we provide some intuition as to how we construct a simulator to use this adversary to assist us in winning the Selective-Set game with the underlying ABE scheme. We first discuss the difference in the initial selection the adversary must make in each game.

In the Selective-ID DishonestUser game, the adversary must first announce the identity  $ID^*$  which it will attack. Note that this implicitly defines a set of  $\ell$  attributes corresponding to this identity. Note that the adversary *does not* have to mention which dummy attributes will be used in this attack. To account for this, we select a random “target” dummy attributes  $\hat{\mathcal{I}} = \{\hat{\mathcal{I}}_j\}_{j \in [m]}$  which we will eventually “force” the adversary to choose. In the Selective-Set game, the ciphertext attributes must be announced in advance. In our reduction, after the adversary announces  $ID^*$  in the Selective-ID DishonestUser game, our simulator chooses dummy attributes by choosing  $\hat{\mathcal{J}} = \{\hat{\mathcal{J}}_j\}_{j \in [m]}$  at random subject to  $\mathcal{R}(\hat{\mathcal{I}}, \hat{\mathcal{J}}) = 0$  (outside the view of the adversary). This ensures that when we later force the adversary to receive the dummy attributes  $\hat{\mathcal{I}}$  in the decryption key  $d_{ID^*}$ , this key cannot decrypt a ciphertext encrypted with the dummy attributes  $\hat{\mathcal{J}}$ . Then we define  $\gamma$  as the set of attributes  $\{u_{i, ID_i^*}\} \cup \bigcup_{j=1}^m \hat{\mathcal{J}}_j$ . Our simulator announces  $\gamma$  as the selected ciphertext attribute set in the Selective-Set game.

In the next phase of the Selective-ID DishonestUser game, our simulator handles the key generation queries by rephrasing these as ABE key queries in Phase 1 of the Selective-Set game. In the A-IBE key generation protocol, we also simulate the ideal OT functionality. Since the dummy attribute set is determined jointly by the inputs of the PKG (our simulator) and the user (the adversary), we can arbitrarily control the output if we “rush” the adversary to give its inputs first. Since we can control the dummy attributes, we can query for a precise policy rather than one which contains all

the dummy attributes (as prescribed in the honest protocol construction). We mention that extra caution needs to be taken in the special case where the queried key is for  $ID^*$ . In the proof, we describe how the simulator handles this case by forcing the dummy attributes to be exactly the previously selected  $\hat{\mathcal{I}}$ .

It is mostly intuitive that this simulation is indistinguishable from a real execution. After all, the simulator is mostly used to transport queries from the Selective-ID DishonestUser game to the Selective-Set game. However, we must argue the tweaks used to cover the discrepancies between the two games does not affect the accuracy of the simulation. We prove this in Lemma 8.

At the end of the Selective-ID DishonestUser game, the adversary outputs a decryption key  $d_{ID^*}$  and a decoder box  $D$ . Even though in the key generation phase it received a decryption key for  $ID^*$  under the dummy attributes  $\hat{\mathcal{I}}$ , it is possible that somehow the adversary was able to come up with a decryption key with a different dummy attribute set. Let  $\mathcal{I}^* = \{\mathcal{I}_j^*\}_{j \in [m]}$  be the dummy attributes in this decryption key. In our proof, we split our analysis into two types of attacks, which can be intuitively thought of as this key being “close” to  $\hat{\mathcal{I}}$  or not. We refer to these two cases as Type-I and Type-II attacks. Lemmas 9 and 10 show that in each case, the simulator has a noticeable chance of decrypting the ciphertext, which will allow us to win the Selective-Set game.

**Proof Organization:** We describe the organization of our proof. We assume toward a contradiction that there exists an adversary  $\mathcal{A}_0$  that wins the Selective-ID DishonestUser game with non-negligible success probability. As before, we use the composition theorem of Canetti [Can00] (Lemma 2) to argue there exists an adversary  $\mathcal{A}$  that succeeds in the hybrid-OT world with non-negligible probability  $\delta$ .

We spend the first half of the proof constructing a simulator  $\mathcal{S}$  that plays the

Selective-Set game against a challenger Ch by internally using  $\mathcal{A}$  as a black-box.  $\mathcal{S}$  behaves by simulating the role of the PKG in the Selective-ID DishonestUser game as well the ideal OT functionality to play against  $\mathcal{A}$ . In Lemma 8, we show that this simulation is perfect from  $\mathcal{A}$ 's point of view. The success probability of this internal  $\mathcal{A}$  must also be  $\delta$ .

Conditioning on  $\mathcal{A}$ 's success, we classify the adversarial strategy of  $\mathcal{A}$  into two cases: Type-I and Type-II attacks. In a Type-I attack, Lemma 9 shows that the simulator has a  $\frac{\delta}{8\eta}$  advantage in winning the Selective-Set game. In a Type-II attack, Lemma 10 shows that the simulator has a  $\frac{\delta^2}{16\eta^2}$  advantage in winning the Selective-Set game. Since the probability of  $\mathcal{A}$  succeeding is  $\delta$ ,  $\mathcal{S}$  has a  $\frac{\delta^3}{16\eta^2}$  overall advantage in the Selective-Set game.

*Proof of Theorem 5.4.3:* Assume there is an adversary  $\mathcal{A}_0$  that wins the DishonestUser game with advantage  $\delta_0$ . As in the previous proof, by Lemma 2 (the composition theorem of Canetti [Can00]), there exists an adversary  $\mathcal{A}$  that has advantage  $\delta$  in the ideal world where the OT protocol in the key generation is replaced by an ideal functionality. By the same lemma, this new advantage  $\delta$  differs from  $\delta_0$  only by a negligible quantity.

We create a simulator  $\mathcal{S}$  that uses  $\mathcal{A}$  to play against a Selective-Set ABE challenger Ch.  $\mathcal{S}$  also simulates the ideal OT functionality which  $\mathcal{A}$  will interact with during its execution. We label the interaction between the parties so it is clear whether we are referring to a phase in the DishonestUser game or the Selective-Set game. We assume that  $\lambda, \epsilon, k, n, m, \tau, \ell$  are known to all parties and fixed in advance. The algorithm for  $\mathcal{S}$  follows:

- **A-IBE Select ID ( $\mathcal{A} \rightarrow \mathcal{S}$ ):** The adversary  $\mathcal{A}$  selects an identity  $ID^*$  as the challenge identity.  $\mathcal{S}$  designates the user attributes  $\{u_{i,j}\}$  and dummy attributes  $\{t_{i,j}\}_{i \in [n], j \in [m]}$ .  $\mathcal{S}$  then selects  $m$  sets of  $k$  elements at random, which we denote

as  $\hat{\mathcal{I}} = \{\hat{\mathcal{I}}_j\}_{j \in [m]}$ .  $\mathcal{S}$  randomly selects a  $\hat{\mathcal{J}}$  subject to the constraint  $\mathcal{R}(\hat{\mathcal{I}}, \hat{\mathcal{J}}) = 0$ .

- **ABE Init ( $\mathcal{S} \rightarrow \mathbf{Ch}$ ):**  $\mathcal{S}$  sets  $\gamma = \{u_{i, \text{ID}_i^*}\} \cup \bigcup_{j=1}^m \hat{\mathcal{J}}_j$  as the challenge set and sends this to  $\mathbf{Ch}$ .
- **ABE Setup ( $\mathbf{Ch} \rightarrow \mathcal{S}$ ):**  $\mathbf{Ch}$  sets up the ABE scheme and sends  $\mathcal{S}$  the public parameters  $\text{PK}_{ABE}$ .
- **A-IBE Setup ( $\mathcal{S} \rightarrow \mathcal{A}$ ):**  $\mathcal{S}$  sets  $\text{PK} = (\text{PK}_{ABE}, \{u_{i,j}\}, \{t_{i,j}\})$  (recall that we are suppressing the IBE portion) and sends this to  $\mathcal{A}$ .
- **A-IBE Key Generation Queries:** For the  $r$ -th query,  $\mathcal{A}$  interacts with  $\mathcal{S}$  to generate a decryption key for  $\text{ID}^{(r)}$ . First, we describe the strategy for  $\mathcal{S}$  in the case that  $\text{ID}^{(r)} \neq \text{ID}^*$ . We break this down into the individual steps of the key generation protocol.

1.  $\mathcal{A}$  does not abort in this step as the attributes are all distinct.
2. The decryption key for the IBE portion is sent from  $\mathcal{S}$  to  $\mathcal{A}$  in this step.
3.  $\mathcal{S}$  (simulating the PKG) deviates from the honest protocol this step. It sets  $u_i = u_{i, \text{ID}_i^{(r)}}$ , and then it selects sets of random dummy attributes  $\mathcal{I}$ . It constructs the *specific* policy “The ciphertext contains all  $u_i$  and at least  $\tau$  elements from each  $\mathcal{I}_j$ ”. Observe that  $\gamma$  does not satisfy this policy as it does not contain some  $u_i$  (due to the fact that  $\text{ID}^{(r)} \neq \text{ID}^*$ ).  $\mathcal{B}$  delegates the computation of this decryption key to  $\mathbf{Ch}$ :

- **ABE Phase 1 query ( $\mathcal{S} \rightarrow \mathbf{Ch}$ ):**  $\mathcal{S}$  asks  $\mathbf{Ch}$  for a decryption key corresponding to the above policy.
- **ABE Phase 1 reply ( $\mathbf{Ch} \rightarrow \mathcal{S}$ ):**  $\mathbf{Ch}$  replies with a decryption key which we decompose as  $(d_0, \{d_{i,j}\}_{i \in \mathcal{I}_j, j \in [m]})$ .

4.  $\mathcal{S}$  waits until the next step to choose the random permutations.



5.  $\mathcal{S}$  and  $\mathcal{A}$  then engage in  $m$  executions of a  $k$ -out-of- $n$  ideal OT functionality where  $\mathcal{S}$  acts as the sender and  $\mathcal{A}$  acts as the receiver. On the  $j$ -th execution, because  $\mathcal{S}$  is also simulating the OT functionality, it can simply wait for the indexing set  $U_j$  to arrive from  $\mathcal{A}$ , then  $\mathcal{S}$  chooses  $\pi_j$  to be a permutation which takes  $U_j$  to  $\mathcal{I}_j$ .  $\mathcal{S}$  then finishes the OT simulation by sending the correct output  $\{d_{i,j}\}_{i \in \mathcal{I}_j}\}_{j \in [m]}$  to  $\mathcal{A}$ . This results in  $\mathcal{A}$  receiving exactly the key components that  $\mathcal{S}$  possesses.
6. ( $\mathcal{S} \rightarrow \mathcal{A}$ ):  $\mathcal{S}$  sends the permutation list  $\pi$  and  $d_0$  to  $\mathcal{A}$ .
7.  $\mathcal{A}$  sets the decryption key  $d = (d_0, \{(\mathcal{I}_j, \{d_{i,j}\}_{i \in \mathcal{I}_j})\}_{j \in [m]})$ . Finally,  $\mathcal{A}$  performs a key sanity check and succeeds.

In the special case where  $\text{ID}^{(\tau)} = \text{ID}^*$ , we modify Step 3 to use  $\mathcal{I} = \hat{\mathcal{I}}$ . Note that by choice,  $\gamma$  still does not pass the policy “The ciphertext contains all  $u_i$  and at least  $\tau$  elements from each  $\mathcal{I}_j$ ” by the construction of  $\hat{\mathcal{I}}$  and  $\hat{\mathcal{J}}$ . Continuing with the construction,  $\mathcal{S}$  now enters the challenge phase of the Selective-Set game.

- **ABE Challenge** ( $\mathcal{S} \rightarrow \text{Ch} \rightarrow \mathcal{S}$ ):  $\mathcal{S}$  chooses two equal-length messages  $M_0$  and  $M_1$  uniformly at random and sends them to **Ch**. The challenger flips a random coin  $b$  and encrypts  $M_b$  with attributes  $\gamma$ . The ciphertext  $\hat{C}$  is sent back to  $\mathcal{S}$ .
- **ABE Phase 2** ( $\mathcal{S} \rightarrow \text{Ch}$ ):  $\mathcal{S}$  skips this phase.
- **A-IBE Create Decoder Box** ( $\mathcal{A} \rightarrow \mathcal{S}$ ):  $\mathcal{A}$  outputs a decryption key  $d_{\text{ID}^*}$  (with dummy attributes  $\mathcal{I}^*$ ) and a decoder box  $D$ . If the decryption key is invalid,  $\mathcal{S}$  aborts and outputs a random value as the guess. Otherwise, let  $\mathcal{U}^*$  be the uniform distribution on dummy attribute collections  $\mathcal{J}$  such that  $\mathcal{R}(\mathcal{I}^*, \mathcal{J}) = 0$ , i.e. ones which cannot be decrypted by  $d_{\text{ID}^*}$ . Similarly, we define  $\hat{\mathcal{U}}$  to be the uniform distribution on  $\{\mathcal{J} | \mathcal{R}(\hat{\mathcal{I}}, \mathcal{J}) = 0\}$ . If the statistical distance between  $\mathcal{U}^*$  and  $\hat{\mathcal{U}}$  more than  $\frac{\delta}{4\eta}$  then we call this a **Type-I**, otherwise we call it a **Type-II** attack.

We simulate the tracing algorithm by having  $\mathcal{S}$  sample  $\eta = \frac{24m\lambda}{\epsilon}$  ciphertexts from  $\mathcal{U}^*$  and attempting to decrypt them with  $D$ . If  $D$  ever succeeds in correctly decrypting one, then we say  $\mathcal{A}$  has succeeded in the simulation.

- **ABE Guess** ( $\mathcal{S} \rightarrow \mathbf{Ch}$ ): Finally,  $\mathcal{S}$  attempts to decrypt  $\hat{C}$  with  $d_{ID^*}$  and  $D$ . In either case, if the decryption is successful and returns  $M_{b^*}$  then  $\mathcal{S}$  outputs the guess  $b^*$ , otherwise it will make a random guess.

This completes the construction of our simulator  $\mathcal{S}$ . We define a few events associated to this simulation that will be used later. Let  $\text{SUCC}_{\mathcal{A}}$  be the event that  $\mathcal{A}$  succeeds in our simulation (and  $\text{Fail}_{\mathcal{A}}$  that  $\mathcal{A}$  does not succeed), and let  $\text{SUCC}_{\mathcal{S}}$  be the event that  $\mathcal{S}$  succeeds in guessing  $b$ . Let  $\mathcal{E}_1$  be the event that  $\mathcal{A}$  successfully used a **Type-I** attack and similarly define  $\mathcal{E}_2$  as the event that  $\mathcal{A}$  used a **Type-II** attack. We first make a claim about the distribution of  $\hat{\mathcal{J}}$  given a view of the adversary, and subsequently show in Lemma 8 that  $\mathcal{A}$  succeeds with the same probability in our simulation as it would in a real execution.

**Claim 2.**  $\hat{\mathcal{J}}$  is independent of  $\mathcal{A}$ 's view except that  $\mathcal{R}(\hat{\mathcal{I}}, \hat{\mathcal{J}}) = 0$ . Indeed, it is equally likely that any of the  $\hat{\mathcal{J}}$  sets (subject to the condition above) was chosen by the simulator.

The claim is true by the observation that  $\mathcal{S}$  does not make use of any  $\hat{\mathcal{J}}_j$  in any of the intermediate steps while communicating with  $\mathcal{A}$ .

**Lemma 8.** The distribution of the information  $\mathcal{A}$  receives in the above protocol is identical to the distribution  $\mathcal{A}$  would have received in a real execution of the Selective-ID DishonestUser game. In particular,  $\Pr[\text{SUCC}_{\mathcal{A}}] = \delta$ .

*Proof.* By observation, this claim holds up to the key query phase. We analyze the difference between the simulation and a real execution in this phase on the  $r$ -th query.

First, we consider the case where  $ID^{(r)} \neq ID^*$ .  $\mathcal{A}$  selects random indices  $U_1, \dots, U_j$  and uses these as private input to the ideal OT functionality. In a real execution, the PKG would have to sample  $\pi_1, \dots, \pi_j$  at random. However, the simulator selected  $\mathcal{I}$  at random and “solved” for  $\pi$ , but this still results in the same distribution. The key components that  $\mathcal{S}$  obtains from  $\text{Ch}$  will be distributed identically to the ones a real execution when restricted to  $\mathcal{I}$  (by requirement, cf. Section 5.3.1). Hence, the distribution of replies received by  $\mathcal{A}$  in the simulation will be identical to that of a real execution.

In the other case where  $ID^{(r)} = ID^*$ ,  $\mathcal{S}$  sets  $\mathcal{I} = \hat{\mathcal{I}}$ . Because each identity may only be queried on once, this set has not been used before nor will it be used again in the simulation. Although this set has been implicitly used to define  $\gamma$ , this was done outside the view of  $\mathcal{A}$ , and thus we are back in the previous case where  $\mathcal{I}$  was selected at random.

After the key generation phase, the only interaction between  $\mathcal{A}$  and  $\mathcal{S}$  is the Create Decoder Box phase, but  $\mathcal{S}$  does not send anything to  $\mathcal{A}$  in this phase. Therefore, the view of  $\mathcal{A}$  in the simulation is identical to that of the real game, and thus, the probability of  $\mathcal{A}$  succeeding in the simulation is identical to it succeeding in a real execution:  $Pr[\text{SUCC}_{\mathcal{A}}] = \delta$ .  $\square$

It remains to show that  $\mathcal{S}$  has a non-negligible advantage in the Selective-Set game. We handle the cases of Type-I attacks and Type-II attacks separately. Define  $\delta_i = Pr[\mathcal{E}_i]$  for  $i = 1, 2$  to be the probability that a Type-I attack was used. By the previous lemma,  $Pr[\text{SUCC}_{\mathcal{A}}] = \delta$ , and clearly  $\delta_1 + \delta_2 = Pr[\text{SUCC}_{\mathcal{A}}] = \delta$ . We state two lemmas that calculate the probability that  $\mathcal{S}$  succeeds under each such event:

**Lemma 9.** Conditioned on event  $\mathcal{E}_1$ ,  $\mathcal{S}$  will correctly decrypt the message using  $d_{ID^*}$  with probability at least  $\frac{\delta}{4\eta}$ .

Consequently,  $\mathcal{S}$  will correctly guess  $b$  with probability at least  $\frac{1}{2} + \frac{\delta}{8\eta}$ , i.e.  $Pr[\text{SUCC}_{\mathcal{S}}|\mathcal{E}_1] \geq \frac{1}{2} + \frac{\delta}{8\eta}$

*Proof.* Recall that in a Type-I attack, we have that the statistical distance between  $\mathcal{U}^*$  and  $\hat{\mathcal{U}}$  is greater than  $\frac{\delta}{4\eta}$ . Then  $\hat{\mathcal{J}}$ , which can be viewed as being sampled randomly from  $\hat{\mathcal{U}}$ , lies outside of the set  $\{\mathcal{J}|\mathcal{R}(\mathcal{I}^*, \mathcal{J}) = 0\}$  with probability greater than  $\frac{\delta}{4\eta}$ . By definition, this means  $\mathcal{R}(\mathcal{I}^*, \hat{\mathcal{J}}) = 1$ , and thus  $d_{\mathbb{D}^*}$  can successfully decrypt it. In such a case,  $\mathcal{S}$  correctly guesses  $b$ . On the other hand if  $\mathcal{R}(\mathcal{I}^*, \hat{\mathcal{J}}) = 0$  then at worst,  $\mathcal{S}$  makes a random guess, which is correct with probability  $\frac{1}{2}$ . We conclude the following:

$$Pr[\text{SUCC}_{\mathcal{S}}|\mathcal{E}_1] \geq \frac{1}{2} + \frac{\delta}{8\eta}$$

□

**Lemma 10.** Conditioned on the event  $\mathcal{E}_2$ ,  $\mathcal{S}$  will correctly decrypt the message (using  $\mathbb{D}$ ) with probability at least  $\frac{\delta}{4\eta}$ .

Consequently,  $\mathcal{S}$  will correctly guess  $b$  with probability at least  $\frac{1}{2} + \frac{\delta^2}{16\eta^2}$ .

Let  $p$  be the probability that  $\mathbb{D}$  decrypts a ciphertext sampled from the distribution  $\mathcal{U}^*$ . Recall that our simulated tracing algorithm makes  $\eta$  attempts to decrypt a  $C$  chosen randomly from  $\mathcal{U}^*$ . For  $\mathcal{A}$  to succeed, it must decrypt at least once, and since the success probability of  $\mathcal{A}$  is  $\delta$  the union bound and Markov's inequality gives us,  $Pr[p > \frac{\delta}{2\eta}] > \frac{\delta}{2}$ .

Recall that in a Type-II attack, we have that the statistical distance between  $\mathcal{U}^*$  and  $\hat{\mathcal{U}}$  is at most  $\frac{\delta}{4\eta}$ . We view  $\hat{C}$  as being sampled from  $\hat{\mathcal{U}}$ , so if  $p > \frac{\delta}{2\eta}$ ,  $\mathbb{D}$  will decrypt  $\hat{C}$  with probability at least  $p - \frac{\delta}{4\eta} > \frac{\delta}{4\eta}$ . Thus overall, with probability at least  $\frac{\delta^2}{8\eta^2}$ ,  $\mathbb{D}$  will decrypt and  $\mathcal{S}$  will correctly guess  $b^* = b$ . Otherwise,  $\mathcal{S}$  correctly guesses  $b$  with probability (at worst)  $\frac{1}{2}$ .

We conclude the following:

$$Pr[\text{SUCC}_{\mathcal{S}}|\mathcal{E}_2] \geq \frac{1}{2} + \frac{\delta^2}{16\eta^2}$$

□

Combining Lemmas 9 and 10 we compute the success probability of  $\mathcal{S}$  as:

$$\begin{aligned} Pr[\text{SUCC}_{\mathcal{S}}] &= Pr[\text{SUCC}_{\mathcal{S}}|\mathcal{E}_1]Pr[\mathcal{E}_1] + Pr[\text{SUCC}_{\mathcal{S}}|\mathcal{E}_2]Pr[\mathcal{E}_2] \\ &\quad + Pr[\text{SUCC}_{\mathcal{S}}|\text{Fail}_{\mathcal{A}}]Pr[\text{Fail}_{\mathcal{A}}] \\ &\geq \left(\frac{1}{2} + \frac{\delta}{8\eta}\right)\delta_1 + \left(\frac{1}{2} + \frac{\delta^2}{16\eta^2}\right)\delta_2 + \frac{1}{2} \cdot (1 - \delta) \\ &\geq \frac{1}{2} + \frac{\delta_1\delta}{8\eta} + \frac{\delta_2\delta^2}{16\eta^2} \end{aligned}$$

As  $\delta_1 + \delta_2 = \delta$ ,  $\mathcal{S}$ 's advantage is at least  $\frac{\delta^3}{16\eta^2}$ . This concludes the proof of Theorem 5.4.3. □

## 5.5 Concrete Construction Based on the DBDH Assumption

In this section, we give a construction of a secure A-IBE scheme  $\mathcal{AIBE}\text{-}1$  based on the decisional BDH assumption. We instantiate the  $\mathcal{AIBE}\text{-GEN}$  scheme using the attribute-based encryption scheme of Goyal et al. [GPS06] which relies on the DBDH assumption. We show how this ABE scheme satisfies the requirements stated in Section 5.3.1. Because the Waters IBE scheme [Wat05] also relies on the decisional BDH assumption, we may use it as our underlying IBE scheme.

For completeness, we show the construction of the compiled protocol below (the IBE portion is implicit and suppressed for clarity).

### 5.5.1 The Construction

We work in a bilinear group  $(p, \mathbb{G}, \mathbb{G}_T, e, P)$ . We define the Lagrange coefficient  $\Delta_{i,S}$  for  $i \in \mathbb{Z}/p\mathbb{Z}$  and some set  $S \subset \mathbb{Z}/p\mathbb{Z}$  to be

$$\Delta_{i,S}(x) := \prod_{j \in S \setminus \{i\}} \frac{x - j}{i - j}.$$

We represent the identities as strings of length  $\ell$  (since an identity  $\text{ID} \in \mathbb{Z}/p\mathbb{Z}$ ,  $\ell$  is the number of bits required to represent an element in  $\mathbb{Z}/p\mathbb{Z}$ ). Let  $n, m, k$  be chosen as usual.

**Setup** For each  $i \in [\ell]$ , choose two numbers  $u_{i,0}$  and  $u_{i,1}$  uniformly at random from  $\mathbb{Z}/p\mathbb{Z}$  such that all  $2\ell$  numbers are different. In addition, for each  $i \in [n]$  and  $j \in [m]$  choose a  $t_{i,j}$  uniformly at random from  $\mathbb{Z}/p\mathbb{Z}$ . Also choose a number  $y$  uniformly at random in  $\mathbb{Z}/p\mathbb{Z}$ .

The published public parameters are:

$$\text{PK} = \left[ \begin{array}{l} \{(U_{i,j} = u_{i,j}P) : i \in [\ell], j \in \{0, 1\}\}, \\ \{(T_{i,j} = t_{i,j}P) : i \in [n], j \in [m]\}, Y = e(P, P)^y \end{array} \right]$$

The master key is:

$$\text{MK} = \left[ \{u_{i,j} : i \in [\ell], j \in \{0, 1\}\}, \{t_{i,j} : i \in [n], j \in [m]\}, y \right]$$

#### Key Generation Protocol

1.  $U$  aborts if the published values in the public key are not all different.
2. PKG runs IBE-KeyGen with ID and  $\text{MK}_{\text{IBE}}$  to obtain a decryption key  $d_{\text{IBE}}$ . This key is sent to  $U$ .
3. In this step, the PKG generates the ABE key. This is performed as follows:

- (a) PKG generates  $m + 1$  random numbers  $y_0, \dots, y_m$  from  $\mathbb{Z}/p\mathbb{Z}$  such that  $y_0 + \dots + y_m = y$ . We will use  $y_0$  to tie in the identity and  $y_1, \dots, y_m$  for the dummy attribute sets.
  - (b) PKG generates  $\ell$  random numbers  $r_1, \dots, r_\ell$  from  $\mathbb{Z}/p\mathbb{Z}$  such that  $r_1 + \dots + r_\ell = y_0$ .
  - (c) PKG generates  $m$  random polynomials (of degree  $\tau - 1$ )  $q_1, \dots, q_m$  with  $q_j(0) = y_j$ .
  - (d) PKG computes the key components  $d_i = r_i/u_{i,\text{ID}_i}P$  for all  $i \in [\ell]$  and sends them to  $U$ . It also computes key components  $d_{i,j} = q_j(i)/t_{i,j}P$  for all  $i \in [n], j \in [m]$  and stores them.
4. PKG chooses random permutations  $\pi_1, \dots, \pi_m \in S_n$ .
  5. PKG and  $U$  then engage in  $m$  executions of a  $k$ -out-of- $n$  oblivious transfer protocol where PKG acts as the sender and  $U$  acts as the receiver. In the  $j$ -th execution, the private input of PKG is the key components  $\{d_{\pi_j(i),j}\}_{i=1}^n$  and the private input of  $U$  is a set  $\mathcal{I}_j$  of  $k$  randomly selected dummy attributes. The private output of  $U$  is the key component  $\{\pi_j(i), d_{\pi_j(i),j}\}_{i \in \mathcal{I}_j}$ .
  6. PKG sends  $U$  the permutation list  $\pi$ .  $U$  checks if he got the right key components as per  $\pi$  (and aborts if the check fails).
  7.  $U$  sets  $d = (\{d_i\}_{i \in [\ell]}, \{(\mathcal{I}_j, \{d_{i,j}\}_{i \in \mathcal{I}_j})\}_{j \in [m]})$  and runs a key sanity check on  $d$ , which we will define.  $U$  aborts if the check fails. Finally,  $U$  sets the decryption key  $d_{\text{ID}} = d$ .

**Key Sanity Check** Here, we show how to define an appropriate key sanity check for the GPSW ABE scheme. Given a decryption key

$$d_{\text{ID}} = (\{d_i\}_{i \in [\ell]}, \{(\mathcal{I}_j, \{d_{i,j}\}_{i \in \mathcal{I}_j})\}_{j \in [m]})$$

for an identity ID, we define a (deterministic) algorithm to check the well-formedness of this key.

1. For each  $j \in [m]$ , let  $S$  be the first  $\tau$  elements of  $\mathcal{I}_j$ . Verify that every point  $x \in \mathcal{I}_j$  lies on the polynomial interpolated by the points in  $S$ :

$$e(d_{x,j}, T_{x,j}) \stackrel{?}{=} \prod_{i \in S} e(d_{i,j}, T_{i,j})^{\Delta_{i,S}(x)}$$

2. Set  $Y_j = \prod_{i \in S} e(d_{i,j}, T_{i,j})^{\Delta_{i,S}(0)}$ .

3. Finally, check that

$$Y \stackrel{?}{=} \prod_{i \in [\ell]} e(U_{i, \text{ID}_i}, d_i) \prod_{j \in [m]} Y_j$$

If all of the above are verified, then the key sanity check passes, otherwise it fails.

**Encryption** To encrypt a message  $M \in \mathbb{G}_T$  under an identity ID, choose a random value  $s \in \mathbb{Z}/p\mathbb{Z}$  and a subset  $\mathcal{J}_j \subset [n]$  of size  $k$  for each  $j \in [m]$ . Compute the ciphertext  $C$  as follows.

$$C = (\{\mathcal{J}_j\}_{j \in [m]}, c = M \cdot Y^s, \{(C_i = sU_{i, \text{ID}_i}) : i \in [\ell]\}, \\ \{(C_{i,j} = sT_{i,j}) : j \in [m], i \in \mathcal{J}_j\})$$

The key generation for ID was set up so that if on each component  $j \in [m]$  the user's dummy attributes ( $\mathcal{I}_j$ ) intersect the ciphertext's dummy attributes ( $\mathcal{J}_j$ ) by more than  $\tau$  then the user can decrypt the message.

**Decryption** To decrypt the ciphertext

$$C = (\{\mathcal{J}_j\}_{j \in [m]}, c, \{C_i\}, \{C_{i,j}\})$$

using  $d_{\text{ID}} = (\{d_i\}_{i \in [\ell]}, \{(\mathcal{I}_j, \{d_{i,j}\}_{i \in \mathcal{I}_j})\}_{j \in [m]})$ , first run a ciphertext sanity check on  $C$ , which we will define.



If the check fails, output  $\perp$ . Otherwise, recover the message  $M$  by selecting (for each  $j \in [m]$ ) a set  $S_j \subset \mathcal{I}_j \cap \mathcal{J}_j$  of threshold size  $\tau$  and performing the following computations:

$$\begin{aligned}
& c / \prod_{i \in [\ell]} e(C_i, d_i) \prod_{j \in [m]} \prod_{i \in S_j} (e(C_{i,j}, d_{i,j}))^{\Delta_{i,S_j}(0)} \\
&= M \cdot e(P, P)^{sy} / \prod_{i \in [\ell]} e(su_{i, \text{ID}_i} P, r_i / u_{i, \text{ID}_i} P) \\
& \prod_{j \in [m]} \prod_{i \in S_j} (e(st_{i,j} P, q_j(i) / t_{i,j} P))^{\Delta_{i,S_j}(0)} \\
&= M \cdot e(P, P)^{sy} / e(P, P)^{sy_0} \prod_{j \in [m]} e(P, P)^{sq_j(0)} \\
&= M \cdot e(P, P)^{sy} / e(P, P)^{sy_0} \prod_{j \in [m]} e(P, P)^{sy_j} \\
&= M
\end{aligned}$$

The decryption algorithm outputs  $\perp$  if  $\mathcal{R}(\mathcal{I}, \mathcal{J}) = 0$ .

**Ciphertext Sanity Check** Our ciphertext sanity check is similar to that in [Goy07]. Given a ciphertext  $C = (\{\mathcal{J}_j\}_{j \in [m]}, c, \{C_i\}, \{C_{i,j}\})$  for an identity  $\text{ID}$ , we define a (deterministic) algorithm to check the well-formedness of this ciphertext. Verify that

$$\begin{aligned}
& e(C_i, U_{1, \text{ID}_1}) \stackrel{?}{=} e(U_{i, \text{ID}_i}, C_1), \quad i \in [\ell], \quad \text{and} \\
& e(C_{i,j}, U_{1, \text{ID}_1}) \stackrel{?}{=} e(T_{i,j}, C_1), \quad j \in [m], i \in \mathcal{J}_j
\end{aligned}$$

If all of the above are verified, then the ciphertext sanity check passes, otherwise it fails.

**Trace** This algorithm takes an identity  $\text{ID}$ , a well-formed decryption key  $d_{\text{ID}}$  (with dummy attributes  $\mathcal{I}$ ) and a decoder box  $\text{D}$  which is  $\epsilon$ -useful. Our tracing algorithm

will run in time polynomial in  $\lambda$  and  $\frac{1}{\epsilon}$ . The tracing algorithm will repeat the following experiment  $\eta = \frac{24m\lambda}{\epsilon}$  times:

1. Choose a random message  $M$  and random  $\mathcal{J}$  subject to the constraint that  $\mathcal{R}(\mathcal{I}, \mathcal{J}) = 0$ .
2. Encrypt  $M$  using the attributes  $\mathcal{J}$  to obtain a ciphertext  $C$ .
3. Attempt to decrypt  $C$  using the decoder box.

If  $D$  ever correctly decrypted a ciphertext, then the algorithm implicates the PKG by returning PKG, otherwise it returns User.

### 5.5.2 Security Proofs

The security of  $\mathcal{ATBE}$ -1 follows directly from that of the generic construction. If we choose a fully simulatable  $k$ -out-of- $n$  OT and an IBE which are secure under the decisional BDH assumption (e.g. Lindell [Lin08] and Waters [Wat05], respectively), we have that:

**Theorem 5.5.1.** *Under the DBDH assumption, the advantages of any adversary in the IND-ID-CPA, DishonestPKG, and Selective-ID DishonestUser games are negligible for  $\mathcal{ATBE}$ -1.*

It remains to show that the GPSW ABE scheme satisfies the requirements stated in Section 5.3.1. By observation, we see that it satisfies the key decomposition requirement. Also, the key sanity check and ciphertext sanity check are clearly correct. In the notation of the construction above, we now show that the sanity checks are sound.

Let  $d$  and  $d'$  be distinct keys for an identity ID that have passed the key sanity check, and let  $C = (\{\mathcal{J}_j\}_{j \in [m]}, c, \{C_i\}, \{C_{i,j}\})$  be a ciphertext that has passed

the ciphertext sanity check. Write  $d = (\{d_i\}_{i \in [\ell]}, \{(\mathcal{I}_j, \{d_{i,j}\}_{i \in \mathcal{I}_j})\}_{j \in [m]})$  and  $d' = (\{d'_i\}_{i \in [\ell]}, \{(\mathcal{I}'_j, \{d'_{i,j}\}_{i \in \mathcal{I}'_j})\}_{j \in [m]})$ .

Since  $d$  and  $d'$  passed the key sanity check and  $C$  passed the ciphertext sanity check, we have that

1. For each fixed  $j$ , each  $d_{i,j}$  and  $d'_{i,j}$  lie on unique polynomials  $q_j$  and  $q'_j$ , respectively.
2. If we write  $Y_j = \prod_{i \in S} e(d_{i,j}, T_{i,j})^{\Delta_{i,S}^{(0)}}$ ,  $Y'_j = \prod_{i \in S'} e(d'_{i,j}, T_{i,j})^{\Delta_{i,S'}^{(0)}}$ , then

$$Y = \prod_{i \in [\ell]} e(U_{i, \text{ID}_i}, d_i) \prod_{j \in [m]} Y_j = \prod_{i \in [\ell]} e(U_{i, \text{ID}_i}, d_i) \prod_{j \in [m]} Y'_j.$$

3.  $C_i = rU_{i, \text{ID}_i}$  and  $C_{i,j} = rT_{i,j}$ , where  $r = \log_{U_{1, \text{ID}_1}}(C_1)$ .

Now because each  $d_{i,j}$  lies on a unique polynomial, the decryption using  $d$  results in

$$\begin{aligned} M &= c / \prod_{i \in [\ell]} e(C_i, d_i) \prod_{j \in [m]} \prod_{i \in S_j} (e(C_{i,j}, d_{i,j}))^{\Delta_{i,S_j}^{(0)}} \\ &= c / \prod_{i \in [\ell]} e(rU_{i, \text{ID}_i}, d_i) \prod_{j \in [m]} \prod_{i \in S_j} (e(rT_{i,j}, d_{i,j}))^{\Delta_{i,S_j}^{(0)}} \\ &= c / \left( \prod_{i \in [\ell]} e(U_{i, \text{ID}_i}, d_i) \prod_{j \in [m]} \prod_{i \in S_j} (e(T_{i,j}, d_{i,j}))^{\Delta_{i,S_j}^{(0)}} \right)^r \\ &= c / Y^r \end{aligned}$$

regardless of how  $S$  is chosen. Similarly, because each  $d'_{i,j}$  lies on a unique polynomial,

the decryption using  $d'$  results in

$$\begin{aligned}
M' &= c / \prod_{i \in [\ell]} e(C_i, d'_i) \prod_{j \in [m]} \prod_{i \in S'_j} (e(C_{i,j}, d'_{i,j}))^{\Delta_{i,S'_j}(0)} \\
&= c / \prod_{i \in [\ell]} e(rU_{i, \text{ID}_i}, d'_i) \prod_{j \in [m]} \prod_{i \in S'_j} (e(rT_{i,j}, d'_{i,j}))^{\Delta_{i,S'_j}(0)} \\
&= c / \left( \prod_{i \in [\ell]} e(U_{i, \text{ID}_i}, d'_i) \prod_{j \in [m]} \prod_{i \in S'_j} (e(T_{i,j}, d'_{i,j}))^{\Delta_{i,S'_j}(0)} \right)^r \\
&= c / Y^r
\end{aligned}$$

regardless of how  $S'$  is chosen.

Thus  $M = M'$ , which shows that the sanity checks are sound.

## 5.6 Conclusion and Open Problems

In this chapter, we proposed a model of a secure accountable authority identity-based encryption scheme which handles black-box decoders. This model is a critical improvement over the original Goyal [Goy07] model. We gave a generic construction of an A-IBE scheme in this enhanced model from any IBE, OT, and KP-ABE scheme. We also gave a concrete construction of an A-IBE scheme under the decisional BDH assumption where the security was respect to the IND-ID-CPA, DishonestPKG, and Selective-ID DishonestUser games. We also mention a manuscript [GLS] that is in preparation which describes achieving perfect completeness for our scheme, as well as a slightly different (and more efficient) combinatorial construction using the non-monotonic property of the OSW KP-ABE scheme.

There are several interesting open problems to be explored. We prove our construction to be secure in the Selective-ID DishonestUser game. This is seemingly due to the underlying connection to the Goyal et al. [GPS06] scheme which is only provably select-set secure. Even if there is some inherent difficulty in proving the full

security of attribute-based encryption schemes such as Sahai-Waters [SW05] or Goyal et al. [GPS06], there may be other tricks that can be done for our construction.

Important questions arise when dealing with the users' decryption keys. The security in both Goyal [Goy07] and our construction only hold when a one decryption key is generated per user (with an explicit break if more than one is made available). This means that if the user loses his key, the user needs to get a new identity  $ID'$  to request a new key. Can we make a A-IBE scheme that allows a single ID to generate polynomially many keys?

Our tracing algorithm takes as input a user's decryption key. If a user lost the key or is deliberately uncooperative in court, then we cannot implicate the PKG or the user. One interesting open problem is to consider the possibility of tracing a box using only a public tracing key, or with the assistance of a tracing authority. What would be the proper additional modifications to the model of accountable authority IBE to account for this?

Finally, we mention the issue of efficiency in our scheme. We view this in terms of the added cost of turning an IBE scheme into an A-IBE scheme by secret sharing the message as in our construction. Each ciphertext and decryption key will now have an additional  $\ell + mk$  group elements and an additional  $mk$  elements to represent the attributes. In our construction, there was a single global parameter  $\lambda$  which governed these parameters (of accountability) as well as the security of the scheme. One can imagine having a second parameter  $\lambda'$  which will determine the *accountability* rather than the security of the scheme which will allow us to adjust the level of accountability in the scheme. The creation of an A-IBE scheme with only a logarithmic or constant sized decryption key and ciphertext remains as a broad open question.

## REFERENCES

- [Abe99] Masayuki Abe. “Mix-Networks on Permutation Networks.” In Kwok-Yan Lam, Eiji Okamoto, and Chaoping Xing, editors, *Advances in Cryptology – ASIACRYPT’99*, volume 1716 of *Lecture Notes in Computer Science*, pp. 258–273. Springer Verlag, 1999.
- [AF07] Masayuki Abe and Serge Fehr. “Perfect NIZK with Adaptive Soundness.” In Salil Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pp. 118–136. Springer Verlag, 2007.
- [AH01] Masayuki Abe and Fumitaka Hoshino. “Remarks on Mix-Network Based on Permutation Networks.” In Kwangjo Kim, editor, *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pp. 317–324. Springer Verlag, 2001.
- [AHL08] Man Ho Au, Qiong Huang, Joseph K. Liu, Willy Susilo, Duncan S. Wong, and Guomin Yang. “Traceable and Retrievable Identity-Based Encryption.” In Steven Bellovin, Rosario Gennaro, Angelos Keromytis, and Moti Yung, editors, *ACNS 08: 6th International Conference on Applied Cryptography and Network Security*, volume 5037 of *Lecture Notes in Computer Science*, pp. 94–110. Springer Verlag, 2008.
- [AP03] Sattam S. Al-Riyami and Kenneth G. Paterson. “Certificateless Public Key Cryptography.” In Chi-Sung Lai, editor, *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pp. 452–473. Springer Verlag, 2003.
- [ASW00] N. Asokan, Victor Shoup, and Michael Waidner. “Optimistic Fair Exchange of Digital Signatures.” *IEEE J. Selected Areas in Comm.*, **18**(4):593–610, April 2000.
- [Bar06] Paulo Barreto. “The Pairing-Based Crypto Lounge.”, 2006. Available at <http://www.larc.usp.br/~pbarreto/pblounge.html>.
- [BB04a] Dan Boneh and Xavier Boyen. “Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles.” In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pp. 223–238. Springer Verlag, 2004.

- [BB04b] Dan Boneh and Xavier Boyen. “Secure Identity Based Encryption Without Random Oracles.” In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pp. 443–459. Springer Verlag, 2004.
- [BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. “Hierarchical Identity Based Encryption with Constant Size Ciphertext.” In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pp. 440–456. Springer Verlag, 2005.
- [BBP04] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. “An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem.” In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pp. 171–188. Springer Verlag, 2004.
- [BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. “Short Group Signatures.” In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pp. 41–55. Springer Verlag, 2004.
- [BDM98] Feng Bao, Robert Deng, and Wenbo Mao. “Efficient and Practical Fair Exchange Protocols with Offline TTP.” In Paul Karger and Li Gong, editors, *Proceedings of IEEE Security & Privacy*, pp. 77–85, May 1998.
- [BF01] Dan Boneh and Matthew K. Franklin. “Identity-Based Encryption from the Weil Pairing.” In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pp. 213–229. Springer Verlag, 2001.
- [BF03] Dan Boneh and Matthew K. Franklin. “Identity Based Encryption from the Weil Pairing.” *SIAM Journal on Computing*, **32**(3):586–615, 2003.
- [BGh07] Paulo S. L. M. Barreto, Steven D. Galbraith, Colm Ó hÉigeartaigh, and Michael Scott. “Efficient pairing computation on supersingular Abelian varieties.” *Designs, Codes and Cryptography*, **42**(3):239–271, 2007.
- [BGL03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. “Aggregate and Verifiably Encrypted Signatures from Bilinear Maps.” In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pp. 416–432. Springer Verlag, 2003.

- [BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. “Evaluating 2-DNF Formulas on Ciphertexts.” In Joe Kilian, editor, *TCC 2005: 2nd Theory of Cryptography Conference*, volume 3378 of *Lecture Notes in Computer Science*, pp. 325–341. Springer Verlag, 2005.
- [BKM06] Adam Bender, Jonathan Katz, and Ruggero Morselli. “Ring Signatures: Stronger Definitions, and Constructions Without Random Oracles.” In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pp. 60–79. Springer Verlag, 2006.
- [BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. “Short Signatures from the Weil Pairing.” *Journal of Cryptology*, 17(4):297–319, September 2004.
- [BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. “Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions.” In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pp. 614–629. Springer Verlag, 2003.
- [BN05] Paulo S. L. M. Barreto and Michael Naehrig. “Pairing-Friendly Elliptic Curves of Prime Order.” In Bart Preneel and Stafford Tavares, editors, *SAC 2005: 12th Annual International Workshop on Selected Areas in Cryptography*, Lecture Notes in Computer Science, pp. 319–331. Springer Verlag, 2005.
- [Bol03] Alexandra Boldyreva. “Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme.” In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pp. 31–46. Springer Verlag, 2003.
- [BPW03] Alexandra Boldyreva, Adriana Palacio, and Bogdan Warinschi. “Secure Proxy Signature Schemes for Delegation of Signing Rights.” *Cryptology ePrint Archive*, Report 2003/096, 2003. <http://eprint.iacr.org/>.
- [BR93] Mihir Bellare and Phillip Rogaway. “Random Oracles are Practical: A Paradigm for Designing Efficient Protocols.” In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pp. 62–73. ACM Press, 1993.
- [Can00] Ran Canetti. “Security and Composition of Multiparty Cryptographic Protocols.” *Journal of Cryptology*, 13(1):143–202, 2000.



- [CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. “The random oracle methodology, revisited.” In *30th Annual ACM Symposium on Theory of Computing*, pp. 209–218. ACM Press, 1998.
- [CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. “On the Random-Oracle Methodology as Applied to Length-Restricted Signature Schemes.” In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pp. 40–57. Springer Verlag, 2004.
- [Cha81] David Chaum. “Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms.” *Communications of the ACM*, **24**(2):84–88, 1981.
- [CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. “A Forward-Secure Public-Key Encryption Scheme.” In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pp. 255–271. Springer Verlag, 2003.
- [CN03] Jean-Sébastien Coron and David Naccache. “Boneh et al.’s  $k$ -Element Aggregate Extraction Assumption Is Equivalent to the Diffie-Hellman Assumption.” In Chi-Sung Lai, editor, *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pp. 392–397. Springer Verlag, 2003.
- [CNS07] Jan Camenisch, Gregory Neven, and Abhi Shelat. “Simulatable Adaptive Oblivious Transfer.” In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pp. 573–590. Springer Verlag, 2007.
- [Coc01] Clifford Cocks. “An Identity Based Encryption Scheme Based on Quadratic Residues.” In Bahram Honary, editor, *Cryptography and Coding, 8th IMA International Conference*, volume 2260 of *Lecture Notes in Computer Science*, pp. 360–363, Cirencester, UK, December 17–19, 2001. Springer Verlag.
- [CS05] Sanjit Chatterjee and Palash Sarkar. “Trading Time for Space: Towards an Efficient IBE Scheme with Short(er) Public Parameters in the Standard Model.” In Dongho Won and Seungjoo Kim, editors, *ICISC 05: 8th International Conference on Information Security and Cryptology*, volume 3935 of *Lecture Notes in Computer Science*, pp. 424–440. Springer Verlag, 2005.
- [EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. “A Randomized Protocol for Signing Contracts.” *Commun. ACM*, **28**(6):637–647, 1985.

- [FR94] Gerhard Frey and Hans-Georg Rück. “A Remark Concerning  $m$ -divisibility and the Discrete Logarithm in the Divisor Class Group of Curves.” *Math. Comput.*, **62**(206):865–874, April 1994.
- [FS01] Jun Furukawa and Kazue Sako. “An Efficient Scheme for Proving a Shuffle.” In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pp. 368–387. Springer Verlag, 2001.
- [FST06] David Freeman, Michael Scott, and Edlyn Teske. “A taxonomy of pairing-friendly elliptic curves.” Cryptology ePrint Archive, Report 2006/372, 2006. <http://eprint.iacr.org/>.
- [Fur05] Jun Furukawa. “Efficient and Verifiable Shuffling and Shuffle-Decryption.” *IEICE Transactions*, **88-A**(1):172–188, 2005.
- [Gal05] Steven Galbraith. “Pairings.” In Ian F. Blake, Gadiel Seroussi, and Nigel Smart, editors, *Advances in Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Notes*, chapter IX, pp. 183–213. Cambridge University Press, 2005.
- [Gen03] Craig Gentry. “Certificate-Based Encryption and the Certificate Revocation Problem.” In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pp. 272–293. Springer Verlag, 2003.
- [Gen06] Craig Gentry. “Practical Identity-Based Encryption Without Random Oracles.” In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pp. 445–464. Springer Verlag, 2006.
- [GH07] Matthew Green and Susan Hohenberger. “Blind Identity-Based Encryption and Simulatable Oblivious Transfer.” In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pp. 265–282. Springer Verlag, 2007.
- [GI08] Jens Groth and Yuval Ishai. “Sub-linear Zero-Knowledge Argument for Correctness of a Shuffle.” In Nigel Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pp. 379–396. Springer Verlag, 2008.
- [GK03] Shafi Goldwasser and Yael Tauman Kalai. “On the (In)security of the Fiat-Shamir Paradigm.” In *44th Annual Symposium on Foundations of Computer Science*, pp. 102–115. IEEE Computer Society Press, 2003.

- [GL07a] Jens Groth and Steve Lu. “A Non-interactive Shuffle with Pairing Based Verifiability.” In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pp. 51–67. Springer Verlag, 2007.
- [GL07b] Jens Groth and Steve Lu. “Verifiable Shuffle of Large Size Ciphertexts.” In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007: 10th International Conference on Theory and Practice of Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pp. 377–392. Springer Verlag, 2007.
- [GLS] Vipul Goyal, Steve Lu, Amit Sahai, and Brent Waters. “Black-Box Accountable Authority Identity-Based Encryption.” Manuscript.
- [GLS08] Vipul Goyal, Steve Lu, Amit Sahai, and Brent Waters. “Black-box accountable authority identity-based encryption.” In Peng Ning, Paul Syverson, and Somesh Jha, editors, *ACM CCS 08: 15th Conference on Computer and Communications Security*, pp. 427–436. ACM Press, 2008.
- [GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. “A Digital Signature Scheme Secure Against Adaptive Chosen-message Attacks.” *SIAM Journal on Computing*, 17(2):281–308, April 1988.
- [GOS06a] Jens Groth, Rafail Ostrovsky, and Amit Sahai. “Non-interactive Zaps and New Techniques for NIZK.” In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pp. 97–111. Springer Verlag, 2006.
- [GOS06b] Jens Groth, Rafail Ostrovsky, and Amit Sahai. “Perfect Non-interactive Zero Knowledge for NP.” In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pp. 339–358. Springer Verlag, 2006.
- [Goy07] Vipul Goyal. “Reducing Trust in the PKG in Identity Based Cryptosystems.” In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pp. 430–447. Springer Verlag, 2007.
- [GPS06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. “Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data.” In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06: 13th Conference on Computer and Communications Security*, pp. 89–98. ACM Press, 2006. Available as Cryptology ePrint Archive Report 2006/309.

- [Gro03] Jens Groth. “A Verifiable Secret Shuffle of Homomorphic Encryptions.” In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pp. 145–160. Springer Verlag, 2003.
- [Gro06] Jens Groth. “Simulation-Sound NIZK Proofs for a Practical Language and Constant Size Group Signatures.” In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pp. 444–459. Springer Verlag, 2006.
- [GS08] Jens Groth and Amit Sahai. “Efficient Non-interactive Proof Systems for Bilinear Groups.” In Nigel Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pp. 415–432. Springer Verlag, 2008.
- [HOT04] Ryotaro Hayashi, Tatsuaki Okamoto, and Keisuke Tanaka. “An RSA Family of Trap-Door Permutations with a Common Domain and Its Applications.” In Feng Bao, Robert Deng, and Jianying Zhou, editors, *PKC 2004: 7th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pp. 291–304. Springer Verlag, 2004.
- [HSV06] Florian Hess, Nigel Smart, and Frederik Vercauteren. “The Eta-Pairing Revisited.” *IEEE Transactions on Information Theory*, **52**(10):4595–4602, 2006.
- [IN83] K. Itakura and K. Nakamura. “A Public-Key Cryptosystem Suitable for Digital Multisignatures.” *NEC J. Res. & Dev.*, **71**:1–8, October 1983.
- [Jou04] Antoine Joux. “A One Round Protocol for Tripartite Diffie-Hellman.” *Journal of Cryptology*, **17**(4):263–276, September 2004.
- [KLS00] Stephen Kent, Charles Lynn, and Karen Seo. “Secure Border Gateway Protocol (Secure-BGP).” *IEEE J. Selected Areas in Comm.*, **18**(4):582–592, April 2000.
- [KM05] Neal Koblitz and Alfred Menezes. “Pairing-Based Cryptography at High Security Levels.” In Nigel Smart, editor, *Proceedings of Cryptography and Coding 2005*, volume 3796 of *LNCS*, pp. 13–36. Springer-Verlag, December 2005.
- [Kob87] Neal Koblitz. “Elliptic Curve Cryptosystems.” *Mathematics of Computation*, **48**:203–209, 1987.

- [LBD04] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. “Secure Key Issuing in ID-based Cryptography.” In *ACSW Frontiers*, volume 32 of *CRPIT*, pp. 69–74. Australian Computer Society, 2004.
- [Len87] H. W. Lenstra. “Factoring Integers with Elliptic Curves.” *The Annals of Mathematics*, **126**:649–673, 1987.
- [Lin08] Andrew Y. Lindell. “Efficient Fully-Simulatable Oblivious Transfer.” In Tal Malkin, editor, *Topics in Cryptology – CT-RSA 2008*, volume 4964 of *Lecture Notes in Computer Science*, pp. 52–70. Springer Verlag, 2008.
- [LMR04] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. “Sequential Aggregate Signatures from Trapdoor Permutations.” In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pp. 74–90. Springer Verlag, 2004.
- [LOS06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. “Sequential Aggregate Signatures and Multisignatures Without Random Oracles.” In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pp. 465–485. Springer Verlag, 2006.
- [Mil86a] Victor S. Miller. “Short Programs for Functions on Curves.”, 1986.
- [Mil86b] Victor S. Miller. “Use of Elliptic Curves in Cryptography.” In Hugh C. Williams, editor, *Advances in Cryptology – CRYPTO’85*, volume 218 of *Lecture Notes in Computer Science*, pp. 417–426. Springer Verlag, 1986.
- [MOR01] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. “Accountable-Subgroup Multisignatures: Extended Abstract.” In *ACM CCS 01: 8th Conference on Computer and Communications Security*, pp. 245–254. ACM Press, 2001.
- [MSK02] Shigeo Mitsunari, Ryuichi Saka, and Masao Kasahara. “A New Traitor Tracing.” *IEICE Transactions*, **E85-A(2)**:481–484, February 2002.
- [MUO96] Masahiro Mambo, Keisuke Usuda, and Eiji Okamoto. “Proxy Signatures for Delegating Signing Operation.” In *ACM CCS 96: 3rd Conference on Computer and Communications Security*, pp. 48–57. ACM Press, 1996.
- [MVO91] Alfred Menezes, Scott Vanstone, and Tatsuaki Okamoto. “Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field.” In *STOC ’91: Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pp. 80–89. ACM, 1991.

- [Nac07] David Naccache. “Secure and Practical Identity-Based Encryption.”, June 2007.
- [Nef01] C. Andrew Neff. “A Verifiable Secret Shuffle and Its Application to e-Voting.” In *ACM CCS 01: 8th Conference on Computer and Communications Security*, pp. 116–125. ACM Press, 2001.
- [Nie02] Jesper Buus Nielsen. “Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case.” In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pp. 111–126. Springer Verlag, 2002.
- [NSK04] Lan Nguyen, Reihaneh Safavi-Naini, and Kaoru Kurosawa. “Verifiable Shuffles: A Formal Model and a Paillier-Based Efficient Construction with Provable Security.” In Markus Jakobsson, Moti Yung, and Jianying Zhou, editors, *ACNS 04: 2nd International Conference on Applied Cryptography and Network Security*, volume 3089 of *Lecture Notes in Computer Science*, pp. 61–75. Springer Verlag, 2004.
- [NSK05] Lan Nguyen, Reihaneh Safavi-Naini, and Kaoru Kurosawa. “A Provably Secure and Efficient Verifiable Shuffle based on a Variant of the Paillier Cryptosystem.” *Journal of Universal Computer Science*, **11**(6):986–1010, 2005.
- [NSZ04] David Nicol, Sean Smith, and Meiyuan Zhao. “Evaluation of Efficient Security for BGP Route Announcements using Parallel Simulation.” *Simulation Modelling Practice and Theory*, **12**:187–216, 2004.
- [Oka88] Tatsuaki Okamoto. “A Digital Multisignature Scheme Using Bijective Public-Key Cryptosystems.” *ACM Trans. Computer Systems*, **6**(4):432–441, November 1988.
- [OO99] Kazuo Ohta and Tatsuaki Okamoto. “Multisignature Schemes Secure Against Active Insider Attacks.” *IEICE Trans. Fundamentals*, **E82-A**(1):21–31, 1999.
- [OSW07] Rafail Ostrovsky, Amit Sahai, and Brent Waters. “Attribute-based encryption with non-monotonic access structures.” In Peng Ning, Sabrina De Capitani di Vimercati, and Paul Syverson, editors, *ACM CCS 07: 14th Conference on Computer and Communications Security*, pp. 195–203. ACM Press, 2007.
- [Pas03] Rafael Pass. “On Deniability in the Common Reference String and Random Oracle Model.” In Dan Boneh, editor, *Advances in Cryptology –*

- CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pp. 316–337. Springer Verlag, 2003.
- [Pat05] Kenneth Paterson. “Cryptography from Pairings.” In Ian F. Blake, Gadiel Seroussi, and Nigel Smart, editors, *Advances in Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Notes*, chapter X, pp. 215–51. Cambridge University Press, 2005.
- [Sha85] Adi Shamir. “Identity-Based Cryptosystems and Signature Schemes.” In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO’84*, volume 196 of *Lecture Notes in Computer Science*, pp. 47–53. Springer Verlag, 1985.
- [Sho97] Victor Shoup. “Lower Bounds for Discrete Logarithms and Related Problems.” In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT’97*, volume 1233 of *Lecture Notes in Computer Science*, pp. 256–266. Springer Verlag, 1997.
- [SK95] Kazue Sako and Joe Kilian. “Receipt-Free Mix-Type Voting Scheme - A Practical Solution to the Implementation of a Voting Booth.” In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology – EUROCRYPT’95*, volume 921 of *Lecture Notes in Computer Science*, pp. 393–403. Springer Verlag, 1995.
- [SOK00] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. “Cryptosystems based on Pairing.” In *SCIS 2000*, Okinawa, Japan, January 2000.
- [SW05] Amit Sahai and Brent R. Waters. “Fuzzy Identity-Based Encryption.” In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pp. 457–473. Springer Verlag, 2005.
- [Wat05] Brent R. Waters. “Efficient Identity-Based Encryption Without Random Oracles.” In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pp. 114–127. Springer Verlag, 2005.
- [Wei40] André Weil. “Sur les fonctions algébriques corps de constantes finis.” *C.R. Academie des Sciences Paris*, **210**:592–594, 1940.
- [Wik05] Douglas Wikström. “A Sender Verifiable Mix-Net and a New Proof of a Shuffle.” In Bimal K. Roy, editor, *Advances in Cryptology – ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pp. 273–292. Springer Verlag, 2005.