Cryptographic Schemes based on Elliptic Curve Pairings

Sattam S. Al-Riyami

Technical Report
RHUL–MA–2005–2
2 February 2005

Department of Mathematics
Royal Holloway, University of London
Egham, Surrey TW20 0EX, England
http://www.rhul.ac.uk/mathematics/techreports

Cryptographic Schemes based on Elliptic Curve Pairings

Sattam S. Al-Riyami

Thesis submitted to the University of London
for the degree of Doctor of Philosophy

Information Security Group
Department of Mathematics
Royal Holloway, University of London
2004

Declaration

These doctoral studies were conducted under the supervision of Dr. Kenneth G. Paterson and Prof. Chris Mitchell.

The work presented in this thesis is the result of original research carried out by myself, in collaboration with others, whilst enrolled in the Department of Mathematics as a candidate for the degree of Doctor of Philosophy. This work has not been submitted for any other degree or award in any other university or educational establishment.

Sattam S. Al-Riyami

Acknowledgements

I'm privileged to have crossed path with Kenny Paterson, who offered me many hours of his precious time every month. From his inspirational assistance and criticism I learned *a lot* about doing research and presenting it and will remain forever grateful. Also I'd like to thank Chris Mitchell for his continued guidance and for his supervision during the first year of my Ph.D..

I'm deeply grateful to all the lecturers, secretaries and students in the maths department, they provided me with an excellent research environment during my doctoral studies. In particular Alex Dent, Geraint Price, Paula Valenca and Simos Xenitellis for insightful discussions. I'd like to express my gratitude to Priya Gopalan, Jane Klemen, Caroline Kudla and Andreas Pashalidis for proof reading content of this thesis. Finally, and most importantly, I owe thanks to my government for sponsoring me, and my family and god for self-evident reasons.

Abstract

Cryptographic Schemes based on Elliptic Curve Pairings:
Contributions to Public Key Cryptography and Key Agreement Protocols

This thesis introduces the concept of *certificateless public key cryptography* (CL-PKC). Elliptic curve pairings are then used to make concrete CL-PKC schemes and are also used to make other efficient key agreement protocols.

CL-PKC can be viewed as a model for the use of public key cryptography that is intermediate between traditional certificated PKC and ID-PKC. This is because, in contrast to traditional public key cryptographic systems, CL-PKC does not require the use of certificates to guarantee the authenticity of public keys. It does rely on the use of a trusted authority (TA) who is in possession of a master key. In this respect, CL-PKC is similar to identity-based public key cryptography (ID-PKC). On the other hand, CL-PKC does not suffer from the key escrow property that is inherent in ID-PKC. Applications for the new infrastructure are discussed.

We exemplify how CL-PKC schemes can be constructed by constructing several certificateless public key encryption schemes and modifying other existing ID based schemes. The lack of certificates and the desire to prove the schemes secure in the presence of an adversary who has access to the master key or has the ability to replace public keys, requires the careful development of new security models. We prove that some of our schemes are secure, provided that the Bilinear Diffie-Hellman Problem is hard.

We then examine Joux's protocol [90], which is a one round, tripartite key agreement protocol that is more bandwidth-efficient than any previous three-party key agreement protocol, however, Joux's protocol is insecure, suffering from a simple man-in-the-middle attack. We shows how to make Joux's protocol secure, presenting several tripartite, authenticated key agreement protocols that still require only one round of communication. The security properties of the new protocols are studied. Applications for the protocols are also discussed.

Contents

List of Figures

List of Tables

Abbreviations

AK:	Authenticated Key Agreement
AKC:	Authenticated Key Agreement with Confirmation
BDH:	Bilinear Diffie-Hellman
BDHP:	Bilinear Diffie-Hellman Problem
BF:	Boneh and Franklin
BLS:	Boneh, Lynn and Shacham
DH:	Diffie-Hellman
DLP:	Discrete Logarithm Problem
CA:	Certification Authority
CCA:	Chosen Ciphertext Attack
CBE:	Gentry's Certificate-based Encryption
CDHP:	Computational Diffie-Hellman Problem
CL-:	Certificateless
CPA:	Chosen Plaintext Attack
CRHF:	Collision Resistant Hash Function
CRL:	Certificate Revocation List
DBDHP:	Decisional Bilinear Diffie-Hellman Problem
DDHP:	Decisional Diffie-Hellman Problem
FFS:	Function Field Sieve
GDHP:	Generalised Diffie-Hellman Problem
HCL-:	Hierarchical Certificateless
ID:	Identifier
ID-:	Identifier-based
IND-:	Indistinguishability of Encryptions
IP:	Internet Protocol
KDC:	Key Distribution Center
KGC:	Key Generating Center
KTC:	Key Translation Center
MAC:	Message Authentication Code
MOV:	Menezes, Okamoto and Vanstone
MTI:	Matsumoto, Takashima and Imai
OCSP:	Online Certificate Status Protocol
OWE:	One-Way Encryption
OWHF:	One-Way Hash Function
PKC:	Public Key Cryptography
PKE:	Public Key Encryption
PKG:	Private Key Generator
PKI:	Public Key Infrastructure
PKS:	Public Key Signature
PoP:	Proof of Possession
SEM:	Security Mediator
STS:	Station-to-Station
TA:	Trusted Authority
TAK:	Tripartite Authenticated Key Agreement
TAKC:	Tripartite Authenticated Key Agreement with Confirmation

Chapter 1

Introduction

Contents

The aim of this chapter is to provide an introduction and present the overall structure of the thesis. This chapter also describes the main contributions of the thesis to public key cryptography and key agreement protocols.

1.1 Motivation

This thesis explores cryptographic schemes based on elliptic curve pairings. Part I is an analysis of cryptographic elliptic curve pairings which will form the basis for the concrete solutions to the cryptographic problems addressed in both Part II and Part III of this thesis. Next, we will briefly outline the main motivations behind these concrete solutions.

The space occupied between two topics or concepts provides fertile ground for scientific research. Part of this thesis finds its roots in the space beteen the two approaches for developing trust in PKC, these are, firstly certificate-based PKC and secondly identity-based public key cryptography (ID-PKC). Both the ID-PKC and certificate-based PKC model are well studied in cryptography. These two established approaches clearly raise a range of questions such as: Are they any other ways of establishing trust using PKC?, for example, does any desirable model lie in between

ID-PKC and certificate-based PKC?, can public keys be managed and used without certificates? The pursuit for the answers to these questions led to the discovery of a new notion in public key cryptography called certificateless public key cryptography (CL-PKC), which is set out in Part II of this thesis.

A recent and important development in cryptographic protocols which uses elliptic curve pairings is Joux's protocol [90]. While Joux's protocol is a very efficient three party key agreement protocol, it is not suitable for open networks because it does not provide authentication. This characteristic of Joux's protocol hampers many of its practical applications; Part III attempts to remedy this clear weakness in the Joux protocol. As we shall see, achieving high levels of security using Joux's protocol is no simple task. We develop and provide analysis of authenticated versions of Joux's tripartite protocols for certificate-based infrastructures in Part III of this thesis.

1.2 Overall Structure and Summary of Contributions

We briefly outline the structure of this thesis and highlight its main contributions:

Part I: In this part, we explain and present relevant background material. In Chapters 2 and 3, we cover the nomenclature and definitions necessary for understanding the remainder of the thesis. We cover elliptic curve pairings, provable security and some computational and decisional problems. We provide expositions for certificate-based and identity-based cryptography and cryptographic workflows. These are required for understanding Part II of this thesis. We explain key agreement protocols and authenticated key agreements protocol goals and attributes. These are prerequisites for understanding Part III of this thesis. We end Part I with a survey of the development of pairing-based cryptography, which also shows how this thesis relates to the wider body of research. We also highlight some cryptographic applications.

Note, that after covering Part I, the reader can move on to either Part II or Part III

of this thesis, as they are independent parts that tackle different aspects of pairing-based cryptography.

Part II: In this part, we introduce and develop CL-PKC. In Chapter 4, we start by describing CL-PKC, comparing it to ID-PKC and standard certificate-based PKC. As the reader shall appreciate, CL-PKC is competitive and has the potential to be used in real world applications. In Chapters 5 and 6, we exemplify the certificateless concept by presenting some very efficient certificateless public key encryption (CL-PKE) schemes and prove their security in an appropriate model. In Chapter 7, we show how arbitrary identity-based public key encryption (ID-PKE) schemes and arbitrary standard public key encryption (PKE) schemes can be combined to construct CL-PKE schemes. Also in Chapter 7, we examine in detail Gentry's certificate-based encryption (CBE) model [76] and identify some of its shortcomings. We describe a simple modification to the CBE model and show how CL-PKE schemes can be transformed into CBE schemes in this modified model. Finally, Chapter 8 builds on existing ID-PKC results to provide some certificateless schemes that can be easily bootstrapped from existing identity-based schemes.

Part III: In Part III, we address the problem with Joux's protocol, namely its lack of authentication. In Chapter 9, we construct some tripartite authenticated key agreement protocols that preserve the communication advantages of Joux's protocol. We then analyze their security properties. We also explain how to transform some authenticated two party Diffie-Hellman based protocols, and tripartite pairing-based protocols into tripartite authenticated protocols, having one party offline. We provide a brief analysis of Shim's protocol [138], showing that it does not make mathematical sense. Finally, we examine the communication advantages of authenticated versions of Joux's protocol in different network settings, providing a pass-optimal authenticated and key confirmed tripartite protocol that generalises the station-to-station protocol [59].

Discussions summarising the merits of this research and pointing towards future research ideas are presented throughout the thesis. As our pairings survey will demon-

strate, it has emerged that pairings from elliptic curves are a very powerful primitive and can be used to build novel cryptographic schemes with interesting properties. This thesis provides further evidence for this, by presenting one round tripartite authenticated key agreement protocols and schemes for certificateless public key cryptography.

1.3 Publications and Origins of Contributions

This thesis contains some previous research published with K.G. Paterson [4, 5, 6, 7]. The CL-PKC encryption model of Section 5.2, security model of Section 6.2 and much of the content of Chapter 8 were originally described in [6, 7]. Some contents of Chapter 9 first appeared in [4] and was later improved on and published in [5]. Finally, the ideas developed in Section 7.2 are a direct result of communication with D. Boneh.

Part I

Definitions and Preliminary Topics

Chapter 2

Definitions

Contents

The primary aims of this chapter are to define elliptic curve pairings and to establish some notational conventions which are used throughout this thesis. Furthermore, we introduce basic ideas of complexity theory and explore the relationships between several different cryptographic definitions.

2.1 Abstract Algebra and the Main Groups

We will make free use of basic concepts about groups, rings and fields [114, Chap.2]. The notation $\mathbb{G}$ is used to denote a group which is a set with some binary operation. We let $\mathbb{G}^*$ denote the set of non-identity elements of the group.

Definition 2.1 The number of elements in $\mathbb{G}$, denoted $|\mathbb{G}|$, is called the *order* of $\mathbb{G}$. A group $\mathbb{G}$ is *finite* if $|\mathbb{G}|$ is finite.

Definition 2.2 A group $\mathbb{G}$ is *cyclic* if there is an element $g \in \mathbb{G}$ such that for each $a \in \mathbb{G}$ there is an integer i with $a = g^i$. Such an element g is called a *generator* of $\mathbb{G}$.

A field is denoted $\mathbb{F}$ and $\overline{\mathbb{F}}$ denotes the algebraic closure of $\mathbb{F}$. A finite field of order t is denoted $\mathbb{F}_t$.

The main groups used in this thesis are $\mathbb{Z}_n$, $\mathbb{G}_1$ and $\mathbb{G}_2$. The group $\mathbb{Z}_n$ denotes the set of integers under addition modulo n. We will use $\{0, 1, \ldots, n-1\}$ to denote the elements of $\mathbb{Z}_n$ and $\{1, 2, \ldots, n-1\}$ to denote the elements of $\mathbb{Z}_n^*$. The two other groups are an additive group $\mathbb{G}_1$ and a related multiplicative group $\mathbb{G}_2$. Both are cyclic groups of large prime order related to elliptic curves over finite fields. In the sequel, the foundation used to describe $\mathbb{G}_1$ is covered and in Section 2.3, the bilinear maps which links $\mathbb{G}_1$ and $\mathbb{G}_2$ are described.

2.2 Elliptic Curves

We will very briefly introduce some basic theory of elliptic curves. Most of the results in this introduction comes from [25, 97]. The aim of this section is to introduce elliptic curves, point compression and point representation.

Elliptic curves can be defined using several different equations. An elliptic curve E over a field $\mathbb{F}$ is commonly given by an *affine Weierstrass* equation of the form

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \tag{2.1}$$

where $a_i \in \mathbb{F}$ for $i = 1, 2, 3, 4, 6$. The elliptic curve $E(\mathbb{F})$ is defined to be the set of points $(x, y) \in \mathbb{F} \times \mathbb{F}$ that satisfy this equation, along with a point at infinity denoted as ∞. In order for this cubic curve to be an elliptic curve it must be smooth. This

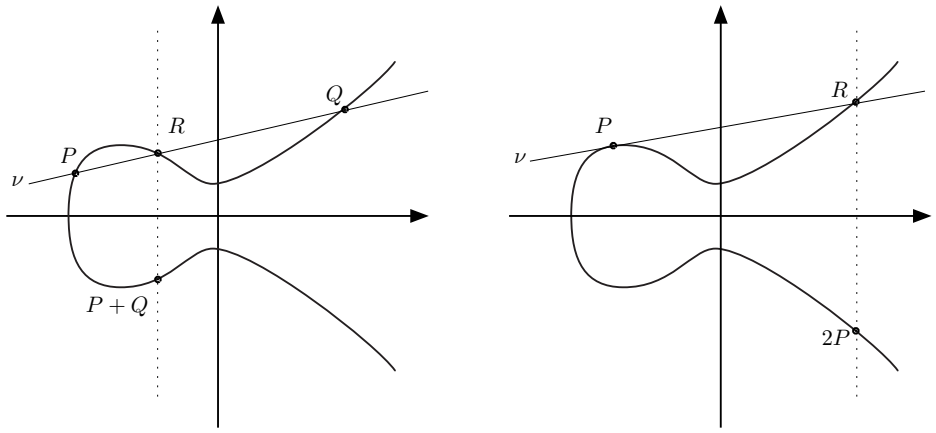

Figure 2.1: Elliptic curve operations defined over the real number field $\mathbb{R}$: addition of two points and doubling of a point.

means there is no point of $E(\overline{\mathbb{F}})$ where both partial derivatives vanish. Thus, for any $(x, y) \in E(\overline{\mathbb{F}})$, both the conditions

$$a_1 y - 3x^2 - 2a_2 x - a_4 = 0 \tag{2.2}$$

and

$$2y + a_1 x + a_3 = 0 \tag{2.3}$$

cannot be simultaneously satisfied.

Without loss of generality, the elliptic curve equation can be simplified and given in the *short Weierstrass* form if the characteristic of $\mathbb{F}$ is neither 2 or 3. This form of equation is:

$$y^2 = x^3 + ax + b \tag{2.4}$$

where $a, b \in \mathbb{F}$. The elliptic curve is still required to be smooth.

If $\mathbb{K}$ is any extension field of $\mathbb{F}$, the set $\{(x, y) \in \mathbb{K} \times \mathbb{K} : E(\mathbb{K})\} \cup \{\infty\}$ with some group operation, $+$, can be used to form a group, denoted $(E(\mathbb{K}), +)$, known as an elliptic curve group.

We look at the elliptic curve over the real number field in Figure 2.1, to help illustrate

how the group operation is defined in the general case. To define the operation on points called *point addition on* $E(\mathbb{K})$, the following rules are followed:

- Let $P \in E(\mathbb{K})$. Then $P + \infty = P$ and $\infty + P = P$. So ∞ serves as the additive *identity* for the group. If $P = \infty$ then we define $-P = \infty$. In what follows, we will let the notation $E(\mathbb{K})^*$ denote the group $E(\mathbb{K})$ excluding the identity element ∞.
- Let $P = (x, y) \in E(\mathbb{K})^*$. Then $-P = -(x, y) = (x, -y)$ and $P + (-P) = \infty$. So the *inverse* of P is $-P$.
- Let $P = (x, y) \in E(\mathbb{K})^*$ and $Q = (x', y') \in E(\mathbb{K})^*$. If $x \neq x'$, then $P + Q = -R$, where $-R$ is a reflection of R in the x-axis and R is the point of intersection of the line joining P and Q with E. This geometric construction can be visualised using the left diagram of Figure 2.1.
- Let $P \in E(\mathbb{K})^*$. Then $P + P = -R$, here $-R$ is a reflection of R in the x-axis. R is the point of intersection for the tangent ν at P with E. This *point doubling* can be visualised using the right diagram of Figure 2.1.

It can be shown that $E(\mathbb{K})$ is *commutative* and *associative* under addition, that is, $P + Q = Q + P$ and $(P + Q) + R = P + (Q + R)$ for all $P, Q, R \in E(\mathbb{K})$ [141, §2]. Thus, the composition rules yield an abelian group $(E(\mathbb{K}), +)$ with identity element ∞. From here on the symbol $+$ will be omitted from the representation and $E(\mathbb{K})$ will be used instead. We will write $P = (x, y)$ for a point in the group $E(\mathbb{K})$. The geometric definitions presented here lead to algebraic formulae for the group law which are valid for any characteristic of the underlying field. These can be found in [25, p. 33].

The notation mP denotes the scalar multiplication of $P \in E(\mathbb{K})$ by $m \in \mathbb{Z}$. The value of mP is the following: for $m = 0$ it is equal to ∞; for $m \leq -1$ it is equal to $(-m)(-P)$; and for $m \geq 1$ it is equal to $\underbrace{P + \ldots + P}_{m \text{ points}}$.

In general, scalar multiplication on elliptic curves is believed to be hard to invert. In Section 2.4.2, this inversion problem is examined in more detail.

A point compression technique is possible within an affine coordinate system, where elliptic curve points are written in the form (x, y), because for every x coordinate a maximum of two possible y coordinates exist. This allows the x coordinate to be sent along with a bit, denoted $\tilde{y}$. For example, in $\mathbb{F}_t$ if t is an odd prime, then we can take $\tilde{y} = y \bmod 2$. Specifications of compression algorithms for the x and y coordinates in $\mathbb{F}_t$ and $\mathbb{F}_{2^n}$ are described in [87, Annex A]. If a compressed point is used by a scheme, extra computation will be required to decompress it to the standard affine representation, (x, y).

Alternative trade-offs between computational complexity (in point addition and point doubling) and storage/communicational bandwidth can be achieved by switching to a different coordinate system to represent the elliptic curve. A popular coordinate systems for elliptic curve cryptography is the projective coordinate system. In the projective coordinates (X, Y, Z) over $\mathbb{F}_t$, the defining equation of the curve in Weierstrass form can be taken as

$$Y^2Z = X^3 + aXZ^2 + bZ^3. \tag{2.5}$$

The affine coordinates (x, y) of points resulting from equation 2.4 are related to the projective coordinates (X, Y, Z) of points resulting from equation 2.5 by the equations: $x = X/Z$ and $y = Y/Z$. The point ∞ in this coordinate system is defined to be the triple $(0, 1, 0)$.

In this thesis, extensive use will be made of pairings on elliptic curves. For cryptographic applications, our focus will be on elliptic curves defined over finite fields. The preferred finite fields are $\mathbb{F}_t$, $\mathbb{F}_{2^n}$ and $\mathbb{F}_{3^n}$, where t is a large prime and $n \in \mathbb{Z}^*$. Curves defined over $\mathbb{F}_{2^n}$ allow more efficient bit computations than those defined over $\mathbb{F}_{3^n}$. Generally, curves defined over $\mathbb{F}_{3^n}$ require smaller key sizes than those defined over $\mathbb{F}_{2^n}$ for equivalent security. However, this advantage may be eliminated if efficient algorithms for fields of characteristic three can be found, we will return to this issue in Section 3.1.

2.3 Bilinear Maps from Elliptic Curve Pairings

We let P denote a generator of $\mathbb{G}_1$, where $\mathbb{G}_1$ is an additive group of some large prime order q. Let $\mathbb{G}_2$ be a related multiplicative group with $|\mathbb{G}_2| = |\mathbb{G}_1|$. A pairing is a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ with the following properties:

1. The map $\hat{e}$ is bilinear: Given $Q, W, Z \in \mathbb{G}_1$, we have

   $$\hat{e}(Q, W + Z) = \hat{e}(Q, W) \cdot \hat{e}(Q, Z) \text{ and } \hat{e}(Q + W, Z) = \hat{e}(Q, Z) \cdot \hat{e}(W, Z).$$

   Consequently, for any $a, b \in \mathbb{Z}_q$:

   $$\hat{e}(aQ, bW) = \hat{e}(Q, W)^{ab} = \hat{e}(abQ, W) = \hat{e}(Q, abW) = \hat{e}(bQ, W)^a.$$

2. The map $\hat{e}$ is non-degenerate: $\hat{e}(P, P) \neq 1_{\mathbb{G}_2}$, where $1_{\mathbb{G}_2}$ is the identity element of $\mathbb{G}_2$.

3. The map $\hat{e}$ is efficiently computable.

This pairing map $\hat{e}$ is sometimes called an admissible pairing. We will show that since $\hat{e}$ is bilinear, the map $\hat{e}$ is also symmetric.

Proof. Being symmetric means that for any $Q, W \in \mathbb{G}_1$, the equality $\hat{e}(Q, W) = \hat{e}(W, Q)$ holds. Both $Q, W \in \mathbb{G}_1$ can be represented using some generator P and some $a, b \in \mathbb{Z}_q$: let $Q = aP$ and $W = bP$. Then we have $\hat{e}(Q, W) = \hat{e}(aP, bP)$ and by bilinearity of $\hat{e}$ we have $\hat{e}(Q, W) = \hat{e}(aP, bP) = \hat{e}(P, P)^{ab} = \hat{e}(bP, aP) = \hat{e}(W, Q)$.

□

Typically, $\mathbb{G}_1$ is a subgroup of the group of points on an elliptic curve over a finite field, i.e. $E(\mathbb{F}_t)$. $\mathbb{G}_2$ is then a subgroup of the multiplicative group of a related finite field. Currently, the parameters are chosen so that $\mathbb{G}_1$ has around 2^{160} elements and $\mathbb{G}_2$ is a subgroup of $\mathbb{F}_{t^r}$ where r is known as the security multiplier (or embedding degree) and t^r has roughly 1024 bits. With the increase in value of r, the pairing computation efficiency decreases.

The map $\hat{e}$ is derived by modifying either the Weil pairing [112] (with both inputs in the same cyclic group) or Tate pairing [69] (with related inputs in the left hand side of the pairing map) on an elliptic curve over $\mathbb{F}_t$. The computational complexity of the Tate pairing is less than that of the Weil pairing. The Weil and Tate pairing need to be modified because the pairings may always output $1_{\mathbb{G}_2}$ in the right hand side of the pairing map[2.1]. Verheul [147] introduced a valuable tool for modifying these pairings called distortion maps. Distortion maps are applicable to a special class of curves called supersingular curves. A distortion map, Φ, is an efficiently computable group endomorphism from $E(\mathbb{F}_t)$ to $E(\mathbb{F}_{t^r})$. Applying Φ to one of the inputs to a pairing ensures the two inputs are linearly independent, therefore, one obtains a non-trivial pairing result. An alternative modification to eliminate trivial pairing results uses trace maps [36, §4.1] (these are group isomorphisms from $E(\mathbb{F}_{t^r})$ to $E(\mathbb{F}_t)$); this technique works on all curves.

Note that many of our schemes can be adapted to situations in which two different groups, denoted as $\mathbb{G}_1'$ and $\mathbb{G}_1''$, are used instead of $\mathbb{G}_1$. The pairing map becomes $e' : \mathbb{G}_1' \times \mathbb{G}_1'' \rightarrow \mathbb{G}_2$ with similar properties to the $\hat{e}$ map. However, the map e' is generally not symmetric. The use of unmodified pairings can increase the range of curves for which our cryptographic schemes can be realised.

Some discussions on security of these elliptic curve pairings and its relation to underlying computation problems is covered next in Section 2.4. However, more comprehensive descriptions of how these groups, pairings and other parameters should be selected in practice for efficiency and security are beyond the scope of this thesis and are described elsewhere. See, for example, [12, 13, 32, 33, 36, 66, 73, 74] for implementation of pairings and selection of curves with suitable properties.

We simply assume throughout the remainder of this thesis that suitable groups $\mathbb{G}_1$ and $\mathbb{G}_2$, a map $\hat{e}$ and an element $P \in \mathbb{G}_1$ can be chosen, and that elements of $\mathbb{G}_1$ and $\mathbb{G}_2$ can be represented by bit strings of the appropriate lengths.

[2.1] The self pairing of any point with itself in an unmodified Weil pairing always returns the trivial result $1_{\mathbb{G}_2}$.

2.4 The Bilinear Diffie-Hellman Problem and Related Problems

We introduce here the computational problems that will form the basis of security for many of our schemes. Many cryptographic primitives are based on number theoretic problems. These cryptographic problems and assumptions exist within the framework of complexity theory. The definitions for the two frequently used complexity theory terms, *negligible function* and *polynomial time algorithm*, are as follows:

Definition 2.3 A function $\epsilon(k)$ is called *negligible* (in the parameter k) if for every $c \geq 0$ there exists an integer $k_c > 0$ such that for all $k > k_c$, $\epsilon(k) < k^{-c}$.

Negligibility is usually used to formalise the hardness of a problem. Since usually we do not know the exact running time of an algorithm (which is the number of bit operations executed by the algorithm) on an input, the big-$\mathcal{O}$ notation is used to represent the order of the asymptotic upper bound. Many definitions presented here are adapted from [114].

Definition 2.4 Let f and g be functions of parameter k. We write $f(k) = \mathcal{O}(g(k))$ if there exists a positive constant c and a positive integer k_0 such that $0 \leq f(k) \leq cg(k)$ for all $k \geq k_0$.

The term 'worst-case running time' of an algorithm is used in Definition 2.5 to represent an upper bound on the execution time for any input, expressed as a function of the input size.

Definition 2.5 A *polynomial time algorithm* is an algorithm whose worst-case running time function is of the form $\mathcal{O}(k^c)$, where k is the input size and c is a constant.

Informally, we regard a polynomial time algorithm as being efficient. A polynomial time algorithm uses resources, such as memory and computation power, which are bounded by a polynomial function in k. It is assumed that processes such as the initialisation of such an algorithm is also performed in polynomial time. Hence, the

running time refers to the whole process and not just the adversary's actions. Subexponential algorithms possess asymptotically slower running times than polynomial time algorithms and asymptotically faster running times than that of exponential time algorithms. If a deterministic algorithm, $\mathcal{A}$, has a random variable as input which affects the output of $\mathcal{A}$, then $\mathcal{A}$ will be viewed as a *probabilistic* algorithm with an internal random variable instead of a random input. In this thesis, the algorithms are always *non-uniform*, which means that they can behave differently for inputs of different sizes [17, §B.2.3].

Negligibility of functions and complexity of algorithms are parametised by values k. In cryptographic algorithms, the value of k is important. This is because it can 'tune' many parameters, such as the size of cryptographic groups and key lengths, within those algorithms[2.2]. The larger k is, the more computation is required to run an algorithm and in Section 3.5 the reader will see that this is precisely what we want to achieve when bounding an adversary's probability of success. Hence, k from here on will be called the *security parameter*. When dealing with the security parameter as input, many authors choose to represent it as 1^k, that is, the string of 1's of length k. Here we use the notation of k directly as an input to our algorithms. However, it should be understood that an input of length k is actually being used.

2.4.1 The Bilinear Diffie-Hellman Generator, Problem and Assumption

The bilinear Diffie-Hellman definitions presented here were first formally given in [32].

Definition 2.6 We say that a randomized algorithm $\mathcal{IG}$ is a *bilinear Diffie-Hellman (BDH) parameter generator*, if:

1. $\mathcal{IG}$ takes a security parameter k as input, for integer $k \geq 1$,

[2.2] Not all security related parameters are scalable, for example, the entropy and length of user chosen passwords does not scale with computation power.

2. $\mathcal{IG}$ runs in polynomial time in k, and

3. $\mathcal{IG}$ outputs a prime number q, the description of groups $\mathbb{G}_1$, $\mathbb{G}_2$ of prime order q and a pairing map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.

Formally, the output of the algorithm $\mathcal{IG}(k)$ is $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$[2.3]. The output q is contained in the description of groups $\mathbb{G}_1$, $\mathbb{G}_2$. Polynomial time (in k) algorithms for computing both $\hat{e}$ and group action in groups $\mathbb{G}_1$, $\mathbb{G}_2$ are also included in the description of $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$.

Definition 2.7 Let $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ be output by algorithm $\mathcal{IG}(k)$ and let P be a generator of $\mathbb{G}_1$. The *bilinear Diffie-Hellman problem (BDHP)* in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ is as follows: Given $\langle P, aP, bP, cP \rangle$ with uniformly random choices of $a, b, c \in \mathbb{Z}_q^*$, compute $\hat{e}(P, P)^{abc} \in \mathbb{G}_2$. An algorithm $\mathcal{A}$ has advantage ϵ in solving the BDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ if

$$\Pr\left[\mathcal{A}(P, aP, bP, cP) = \hat{e}(P, P)^{abc}\right] \geq \epsilon.$$

Here the probability is measured over the random choices of $a, b, c \in \mathbb{Z}_q^*$, $P \in \mathbb{G}_1^*$ and the random bits of $\mathcal{A}$.

The BDH assumption states that no probabilistic polynomial time algorithm has non-negligible advantage (in k) in solving the BDHP for $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ generated by $\mathcal{IG}$ on input k.

2.4.2 Related Problems and Assumptions

The hardness of the BDHP forms the security foundation for many pairing-based cryptographic schemes. Other important computational problems related to pairing-based schemes exist. Some of these are covered here.

[2.3] Throughout this thesis the notation $\langle x_1, x_2, \ldots, x_n \rangle$ denotes a n-tuple formed of the n objects $x_1, x_2 \ldots, x_n$. Often, as is the case here, all the x_is in the n-tuple do not come from the same set.

Definition 2.8 Let $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ be output by algorithm $\mathcal{IG}(k)$ and let P be a generator of $\mathbb{G}_1$. The *generalized bilinear Diffie-Hellman problem (GBDHP)* in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ is as follows: Given $\langle P, aP, bP, cP \rangle$ with uniformly random choices of $a, b, c \in \mathbb{Z}_q^*$, output a pair $\langle Q \in \mathbb{G}_1^*, \hat{e}(P,Q)^{abc} \in \mathbb{G}_2 \rangle$. An algorithm $\mathcal{A}$ has advantage ϵ in solving the GBDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ if

$$\Pr\left[\mathcal{A}(P, aP, bP, cP) = \langle Q, \hat{e}(P,Q)^{abc} \rangle\right] \geq \epsilon.$$

Here the probability is measured over the random choices of $a, b, c \in \mathbb{Z}_q^*$, $P \in \mathbb{G}_1^*$ and the random bits of $\mathcal{A}$.

Similarly, the GBDH assumption states that no probabilistic polynomial time algorithm has non-negligible advantage (in k) in solving the GBDHP for $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ generated by $\mathcal{IG}$ on input k. Notice that an algorithm used to solve the BDHP can be used to solve the GBDHP in which the algorithm outputs the choice $Q = P$. While the GBDHP may appear to be in general easier to solve than the BDHP because the algorithm gets to choose Q, no polynomial-time algorithm is known for solving either the GBDHP or the BDHP when the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ and the pairing $\hat{e}$ are appropriately selected. If the GBDHP algorithm also outputs $s \in \mathbb{Z}_q^*$ such that $Q = sP$, then the problems are equivalent. The GBDHP is the foundation of security for the schemes in Chapter 8.

Dupont and Enge [65] defined a version of the generalised BDHP in the context of unmodified pairings, which is a natural generalisation of the BDHP to that setting. In that version [65] of the generalised BDHP, the problem instance is $\langle P, Q, aP, bP, cP, cQ \rangle$ with uniformly random choices of $a, b, c \in \mathbb{Z}_q^*$, $P \in \mathbb{G}_1'^*$ and $Q \in \mathbb{G}_1''^*$, and the problem objective is to output $e'(P,Q)^{abc} \in \mathbb{G}_2$. The setting is in context of parameters of the form $\langle e', \mathbb{G}_1', \mathbb{G}_1'', \mathbb{G}_2 \rangle$ where the pairing map is $e' : \mathbb{G}_1' \times \mathbb{G}_1'' \to \mathbb{G}_2$. This problem is believed to be hard and if $\mathbb{G}_1' = \mathbb{G}_1'' = \mathbb{G}_1$, $Q = P$, and the input is modified then the above problem is the BDHP. Notice that in the general case when $Q \neq P$, the value of cQ is also provided, whilst the pair $\langle aQ, bQ \rangle$ is not provided. Also notice that the problem instance (not the solving algorithm, as in Definition 2.8) contains Q as input. This problem and the GBDHP

we describe in Definition 2.8 are incomparable because of the differences in settings, the nature of element Q and the information provided (i.e., group elements) to the solving algorithm.

Definition 2.9 We define a number of computational and decisional problems that are related to the BDHP:

- Let $\mathbb{G}$ be a finite cyclic group and let g be a generator of $\mathbb{G}$. The *discrete logarithm problem (DLP)* in $\mathbb{G}$ is as follows: Given $\langle g, g^a \rangle$ with uniformly random choice of $a \in \mathbb{Z}^*_{|\mathbb{G}|}$, find $a \in \mathbb{G}$.
- Let $\mathbb{G}$ be a finite cyclic group and let g be a generator of $\mathbb{G}$. The *computational Diffie-Hellman problem (CDHP)* in $\mathbb{G}$ is as follows: Given $\langle g, g^a, g^b \rangle$ with uniformly random choices of $a, b \in \mathbb{Z}^*_{|\mathbb{G}|}$, find $g^{ab} \in \mathbb{G}$.
- Let $\mathbb{G}$ be a finite cyclic group, and let g be a generator of $\mathbb{G}$. The *decisional Diffie-Hellman problem (DDHP)* in $\mathbb{G}$ is as follows: Given $\langle g, g^a, g^b, g^c \rangle$ with uniformly random choices of $a, b, c \in \mathbb{Z}^*_{|\mathbb{G}|}$, determine if $g^{ab} = g^c$.

Note here that $\mathbb{G}$ is a multiplicative group and that computational assumptions related to above problems can be stated for a system specified parameter generator in an obvious manner. The CDHP can be easily solved if one can compute a given g and g^a, which is precisely the DLP. In fact, solving the DLP is the only known method to solve the CDHP. For the lower bound on the hardness of the CDHP for various elliptic curves, see [117].

The GBDHP is related to a genalisation of the CDHP in $\mathbb{G}_1$ and $\mathbb{G}_2$ in the same way that the BDHP is related to the standard CDHP in those groups [32, 73]. The generalisation of the CDHP is the following: Given g, a generator of a finite cyclic group $\mathbb{G}$, and $\langle g^a, g^b \rangle$ with uniformly random choices of $a, b \in \mathbb{Z}^*_{|\mathbb{G}|}$, find a generator $v \in \mathbb{G}$ and $v^{ab} \in \mathbb{G}$. An associated decisional problem can also be defined for the BDHP.

Definition 2.10 Let $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ be output by algorithm $\mathcal{IG}$ and let P be a generator of $\mathbb{G}_1$. The *decisional bilinear Diffie-Hellman problem (DBDHP)* in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ is as follows: Given $\langle P, aP, bP, cP \rangle \in \mathbb{G}_1$ and a random element $Q \in \mathbb{G}_2^*$ with uniformly random choices of $a, b, c \in \mathbb{Z}_q^*$, determine whether $Q = \hat{e}(P,P)^{abc}$ or not. A distinguisher $\mathcal{A}$ for the DBDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ has advantage ϵ in solving the DBDHP if

$$\begin{aligned} &|\Pr[\mathcal{A}(P, aP, bP, cP, Q) = 1] - \\ &\Pr\left[\mathcal{A}(P, aP, bP, cP, \hat{e}(P,P)^{abc}) = 1\right]| \quad \geq \epsilon \end{aligned}$$

Here the probability is measured over the random choices of $a, b, c \in \mathbb{Z}_q^*$, $Q \in \mathbb{G}_1^*$, $P \in \mathbb{G}_1^*$ and the random bits of $\mathcal{A}$.

2.4.3 Implications of Bilinear Maps

Some relationships between the computational and decisional problems described in the previous section will be explored next.

As a consequence of bilinearity we will show that the BDHP for parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ is no more difficult to solve than the CDHP in either $\mathbb{G}_1$ or $\mathbb{G}_2$.

Proof. Given $\langle P, aP, bP, cP \rangle \in \mathbb{G}_1^4$, let $\gamma = \hat{e}(P,P)^{abc}$ be the solution to the BDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$. We have two ways of solving the BDHP by using CDHP oracles: (i) By solving the CDHP on input $\langle P, aP, bP \rangle$ in additive group $\mathbb{G}_1$, we can find $abP \in \mathbb{G}_1$. Given abP one can compute $\hat{e}(abP, cP) = \gamma \in \mathbb{G}_2$, which is the solution to the BDHP.
(ii) By solving the CDHP on input $\langle \hat{e}(P,P), \hat{e}(aP,P), \hat{e}(bP,cP) \rangle$ in multiplicative group $\mathbb{G}_2$, we can find $\hat{e}(P,P)^{abc} = \gamma \in \mathbb{G}_2$, which is the solution to the BDHP. □

The reverse relationship is not known.

Another consequence of bilinearity is that if we are operating in the context of the tuple $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$, the DDHP in $\mathbb{G}_1$ can be solved in polynomial time. Suppose that an entity wants to confirm that $cP = abP$ for the tuple $\langle P, aP, bP, cP \rangle$ where a, b

and c are elements of $\mathbb{Z}_q^*$. Then that entity simply needs to verify that the equality $\hat{e}(aP, bP) = \hat{e}(P, cP)$ holds; if it does, then in the tuple $\langle P, aP, bP, cP\rangle$ we must have $cP = abP$. This valuable insight was pointed out by Joux and Nguyen [92]. The groups where the DDHP is easy and the CDHP is hard are called gap groups.

If an efficient map $\hat{e}$ is obtained from the Weil or Tate Pairing, then this may lead to a sub-exponential algorithm for the DLP on elliptic curves. To illustrate this using a modified pairing, suppose we let $P, Q \in \mathbb{G}_1$ and suppose the pair $\langle P, Q\rangle$ is an instance of the DLP in $\mathbb{G}_1$. The goal of the solving algorithm is to determine $a \in \mathbb{Z}_q$ such that $Q = aP$. Let $\mu = \hat{e}(P, P), \rho = \hat{e}(Q, P) \in \mathbb{G}_2$ and consider the pair $\langle \mu, \rho\rangle$ as an instance of the DLP in $\mathbb{G}_2$. By bilinearity we know that $\rho = \mu^a$. So the ability to solve the DLP in $\mathbb{G}_2$ gives the ability to solve it in $\mathbb{G}_1$, if there is a sub-exponential algorithm for the former and r is small. However, even if there is a sub-exponential algorithm in $\mathbb{G}_2$, it may be no better than existing algorithms in $\mathbb{G}_1$ because r may be large. Hence, if we want the DLP to be hard in $\mathbb{G}_1$, elliptic curves which can be used by this map must be chosen carefully so that the DLP remains hard in $\mathbb{G}_2$.

The first application of the elliptic curve pairings to cryptography was for solving the DLP on elliptic curves. An unmodified Weil pairing was used by Menezes, Okamoto and Vanstone [112] to mount an attack on supersingular curves (which have $r \leq 6$). Another bilinear map, called the Tate pairing was used by Frey and Rück [69] in a similar attack. As before, these attacks really only work when r is small.

2.5 Other Notation

Throughout this thesis, a string x is a member of $\{0,1\}^*$ where the superscript '*' symbolises an unspecified bit length. When x and y are strings, $x\|y$ denotes their concatenation. The symbol $\oplus$, denotes a bit-wise exclusive OR (XOR) operator. For protocol messages 'Sends to' is denoted by '$\rightarrow$'.

Chapter 3

Preliminary Topics

Contents

The aim of this chapter is to give an overview of cryptographic infrastructures and protocols, and the foundations of provable security. A literature review of pairing-based cryptography is also presented, with the aim of putting into perspective the contributions of this thesis.

3.1 Efficiency

Before introducing the concept of efficiency, we will introduce some basic terminology. An *entity* (or party) is someone or something which sends, receives, or manipulates information. It may be a human being or a computer terminal. This entity could be runing a *scheme*, which is a general term referring to a set of algorithms used to provide specific services. For example, an encryption scheme provides a confidentiality service. Formal descriptions of specific schemes and services are given in Section 3.2 and Section 3.5.1. A *protocol* is an algorithm involving multiple communicating parties, defined by a sequence of steps precisely specifying the actions required of the parties in order to achieve a specified objective. Technically, since a protocol's objective could be to provide a specific service, all schemes can be called protocols. For example, an encryption scheme can be considered a protocol in which only one message is sent between two entities. In this thesis, however, we will make a distinction and reserve the term protocol for interactions which achieve the class of objectives introduced in Section 3.3. Moreover, schemes can be used as tools (or primitives) within a protocol.

The security parameter, number of entities and nature of the communication and computer system all affect efficiency. Given that the efficiency of schemes and protocols is a principal factor shaping research in real world applications, it is important to define relevant efficiency criteria.

Computational complexity: To measure the amount of computation, two computational attributes are defined: *computational overhead* and the ability to perform *precomputation*.

- Computational overhead: This property refers to the cost of all arithmetic computations. Sometimes this is quantified in terms of time for specific algorithms on specific processors. In most schemes and protocols in this thesis, elliptic curve pairings are usually the dominant calculation. With recent advances in efficient implementation of pairings [12, 74], however, the complexity of a pairing computation is now of a similar order of magnitude to that of an elliptic curve point multiplication.
- Precomputation: This property refers to the potential for entities to precompute part of a protocol or scheme in their spare time. This precomputation, which is typically performed in an offline stage, may facilitate a faster response in an online stage.

It may also be desirable that one or more entities (perhaps with limited computational environments) perform fewer computations than other entities (with more powerful computational environments).

Communication complexity: It is an advantage when a protocol has low *communication overhead*. Designing protocols with a minimal number of *passes*, *rounds* and *broadcasts* is also usually advantageous. We now define these important communication criteria and discuss the relevant scenarios for using rounds, broadcasts or passes.

- Communication overhead: This property measures the number of bits transmitted by each participating entity. In many protocols or schemes, keys are communicated. When keys are communicated, protocols and schemes based on elliptic curves can have a low communication overhead. This is because, for comparable cryptographic strength, the elliptic curve key size (denoted N_{ec}) grows only slightly faster than that of the cube root of the corresponding conventional key size (denoted N_{cn}). More precisely, with current algorithmic knowledge [25, §I.3],

$$N_{\text{ec}} = \beta N_{\text{cn}}^{1/3} (\log(N_{\text{cn}} \log 2))^{2/3} \tag{3.1}$$

where $\beta = \frac{2(64/9)^{1/3}}{(\log 2)^{2/3}} \approx 4.91$. Equation 3.1 may not hold for elliptic curve schemes which are pairing-based because they use special curves with ad-

ditional properties. Special curves and curves which have been proven insecure are usually avoided in traditional elliptic curve cryptography. Furthermore, Coppersmith's algorithm [55] which was originally described for fields of characteristic two, may be generalised to fields with small characteristic and the performance of the Function Field Sieve (FFS) [2, 91] (designed for fields with small characteristic) is only well known for fields of characteristic two. Note, however, that recently the first implementation of the FFS in characteristic three has been carried out by Granger *et al.* in [80]. These algorithms may allow for better attacks on some curves – recall Section 2.4.3. Hence, the size of the elliptic curve key for pairings (denoted $N_{\text{pair}-\text{ec}}$) is at least N_{ec} for the same level of security.

- Passes: The number of passes is the total number of messages exchanged in the protocol.
- Broadcast: A broadcast message is a message that is sent to every party in a protocol.
- Rounds: A round consists of all the messages that can be sent and received in parallel within one time unit (synchronised communication is assumed). It is desirable to minimise the number of rounds so as to tolerate network delays.

These notions of communication complexity are each more or less appropriate depending on the particular network architecture that is used. For example, wireless systems operate usually in broadcast mode, as do Ethernet systems in non-switched environments. Therefore, in these systems every packet propagates to (and is available to) all nodes. Thus, the number of rounds and broadcasts are a more natural way of measuring a protocol's communication complexity for such systems. On the contrary, the Internet Protocol running over a public network like the Internet is based on the concept of point-to-point communication, where the number of passes is the right measure.

Storage complexity: This is primarily a measure of memory needed for storage (for example, keys, certificates and algorithms) and working memory required to run a cryptographic scheme or protocol. The notion of storage complexity

also extends to the memory required to store and run a scheme or a protocol's actual algorithms. Storage complexity can be difficult to quantify as it is implementation dependent.

System complexity: We believe that the notions of communication, computational and storage complexities are insufficient to describe a scheme or a protocol. This is because in most circumstances, a cryptographic scheme or protocol does not exist in isolation.

Thus, we extend the scope of complexity by introducing the notion of *system complexity*. Entities include servers, trusted authorities (TAs) and participating parties. Every protocol or scheme contains a set of algorithms. A set of entities and algorithms form an integrated system. Thus, an integrated system encapsulates the set of elements that collectively work as a unit.

We introduce this qualitative measure because it gathers some practical efficiency characteristics of a scheme or a protocol. Some of these characteristics are subtle, but cannot be taken for granted. System complexity captures the way entities participate in schemes or protocols. Moreover, the relationships between different schemes and protocols are also captured. We define the terms *flexibility*, *interactivity* and *interoperability* next.

- Flexibility: This property relates to how an entity uses the algorithms of a specific scheme or protocol. A flexible scheme or protocol is one which is adaptable in deployment. This might be achieved by reducing the degree of temporal ordering required for executing different algorithms within the scheme or protocol. For example, a flexible scheme can render practical efficiency gains in the system because processes such as registration need not be performed in a particular order. This can create efficient applications which use novel cryptographic workflows. See Sections 3.2.5 and 4.6.3 for examples. A flexible protocol is one where the messages transmitted in a protocol are *independent* of each other, in the sense that they can be sent and received in any order. A protocol with this property is labelled as a *message independent* protocol.
- Interactivity: This property relates to how the entities interact in a scheme

or protocol. The number of interacting entities, nature of entity interaction and number of interactions may be examined. Decreasing the involvement of online servers and TAs usually reduces the number of rounds when looking at the communication complexity of any algorithm. When the scheme or protocol relies on less entities, it usually has less points of failure. Thus, the instantiation of a low-interaction protocol or scheme within the system makes it simpler and more fault tolerant. Additional improvement in efficiency is achieved if the interaction does not require an authentic or confidential channel to be set up. Further benefits occur when the number of interactions is reduced. *Non-interactive* is a term indicating that no interaction at all is required between entities. It is most useful in describing certain types of encryption schemes and key distribution schemes. If an encryption scheme is non-interactive, it does not require an interaction with any entity prior to an encryption taking place. Similarly, a non-interactive key distribution scheme does not require any interaction between entities to set up a shared key. The reader is referred to the survey in Section 3.6 for examples of such schemes.

- Interoperability: This property relates a protocol's or a scheme's interaction with other protocols and schemes. The ability to share algorithms and keys across multiple protocols and schemes is always desirable. This generally allows the system to be optimised and decreases the system's inherent complexity and inefficiencies, for then not every algorithm or key need to be set up independently. For example, code (algorithms) within the system can be reused thereby reducing storage. Chapter 8 contain examples of interoperable algorithms.

Of course, a flexible and low-interaction scheme or protocol which is interoperable is generally considered very desirable.

3.2 Public Key Cryptography

Cryptography is about the prevention and detection of malicious activities. The four fundamental goals of cryptography are [114]: (*i*) *confidentiality*: keeps data secret from all but those authorized to access it; (*ii*) *data integrity*: ensures data has not been altered by unauthorized or unknown means; (*iii*) *authentication*: corroboration of the identity of an entity – subdivided into two classes: *entity authentication* when it relates to entities and *data origin authentication* when it relates to information with corroboration of the identity (by definition this provides data integrity); (*iv*) *non-repudiation*: prevents an entity from denying previous commitments or actions.

We distinguish between asymmetric and symmetric cryptography. Asymmetric cryptography is often called public key cryptography (PKC). PKC involves two distinct keys, K_{pub} and K_{priv}. The public key, K_{pub}, can be widely distributed without compromising its corresponding private key, K_{priv}. In some systems, K_{priv} remains only known to the entity that generated it, whilst in other systems K_{priv} is given to an user by another entity. We will return to this issue in Section 3.2.4.

Symmetric cryptography involves only secret keys. The secret key must remain only known to the entities who use it. Block ciphers, stream ciphers, and message authentication codes (MACs) are all examples of symmetric primitives.

Symmetric cryptography requires the secret keys to be securely distributed between the entities. The distribution of secret keys requires prior communication of shared secret keys or secure channels. In practice, for symmetric cryptography, a secure channel is very difficult to achieve in the absence of an online TA which acts as either a key distribution center (KDC) or a key translation center (KTC). More innovative and effective ways of key management and achieving a wider set of cryptographic goals can be obtained by using PKC. To use PKC in practice, a TA is required. This TA is called a certification authority (CA) if certified public keys are used. The CA need not be online. The focus of this thesis is on PKC and particular attention will be paied to CAs in Section 3.2.3.

3.2.1 Cryptographic Primitives

Cryptographic primitives can be used when communicating in the presence of an adversary. Modern cryptographic primitives are developed after defining security through classifying possible attacks and modelling an adversary's capabilities to mount those attacks. A primitive's security is then shown to be directly related to the hardness of some well-defined and widely studied computational problem, like the computational problems in Section 2.4.

The meaning of the term *secure* for public key encryption (PKE) and public key signature (PKS[3.1]) schemes is the subject of Section 3.5. Next, we define PKE schemes which provide confidentiality and PKS schemes which provide non-repudiation, data integrity and data origin authentication.

3.2.1.1 Public Key Encryption

A PKE scheme, Π_{PK}, is usually specified by three algorithms; Key-Generation, Encrypt and Decrypt. These three algorithms do not include an algorithm which only deals with setting up of the scheme. Our definition of Π_{PK} includes a Setup algorithm to, for example, explicitly define the groups in which we are operating. This separation is useful in developing future concepts. For example, we construct schemes where the description of the key generation algorithm can be altered without affecting the system's parameters – see Section 7.2. In many practical applications of PKE, the output of the Setup algorithm is separated from the output of Key-Generation algorithm. Hence, we specify Π_{PK} by four algorithms: Setup, Key-Generation, Encrypt and Decrypt, where:

Setup: is a probabilistic polynomial time algorithm, which takes as input a security

[3.1] This abbreviation can be avoided, since signature schemes are practically always assumed to be in the public key cryptography setting. However we use it for consistency, since the abbreviation PKE is in use.

parameter k and returns system-wide parameters 'params'. The finite message (or plaintext) space, $\mathcal{M}$ and finite ciphertext space, $\mathcal{C}$ are included in params. Both $\mathcal{M}$ and $\mathcal{C}$ are defined by the security parameter k.

Key-Generation ($\mathcal{K}$): is a probabilistic polynomial time algorithm, which takes as input params and returns two keys; a public key K_{pub} and a private key K_{priv}. We write $\langle K_{priv}, K_{pub} \rangle \leftarrow \mathcal{K}(\mathsf{params})$.

Encrypt ($\mathcal{E}$): is a probabilistic polynomial time algorithm, which takes as input a message $M \in \mathcal{M}$, params, and the public key K_{pub}. It returns a ciphertext $C \in \mathcal{C}$. When clear, we write $C \leftarrow \mathcal{E}(M, K_{pub})$ as shorthand. We do not include params as input to simplify presentation.

Decrypt ($\mathcal{D}$): is a polynomial time algorithm, which takes as input a ciphertext $C \in \mathcal{C}$, params, and a private key K_{priv}. It returns a message $M \in \mathcal{M}$ or a message $\perp$ indicating an invalid ciphertext C. When clear, we write $M \leftarrow \mathcal{D}(C, K_{priv})$, as before, we do not include params as input to simplify presentation.

The output M should result from applying algorithm $\mathcal{D}$ with inputs k and K_{priv} on a ciphertext C generated by using algorithm $\mathcal{E}$ with inputs k and K_{pub} on message M. We say that a PKE scheme is sound if M is the output of $\mathcal{D}(\mathcal{E}(M, K_{pub}), K_{priv})$ for all $\langle K_{priv}, K_{pub} \rangle \leftarrow \mathcal{K}(\mathsf{params})$. Algorithm $\mathcal{E}$ is probabilistic to avoid an undesirable property where the output of the encryption scheme does not change for a fixed message input. With this property, the scheme cannot acheive semantic security – see Section 3.5. Algorithm $\mathcal{D}$ is usually deterministic, although, probabilistic decryption algorithms also exist.

3.2.1.2 Signature

A PKS scheme is specified by four algorithms: Setup, Key-Generation, Sign and Verify, where:

Setup: is a probabilistic polynomial time algorithm, which takes as input a security parameter k and returns system-wide parameters 'params'. The finite message space, $\mathcal{M}$ and finite signature space, $\mathcal{S}$ are included in params. Both $\mathcal{M}$ and $\mathcal{S}$ are defined by the security parameter k.

Key-Generation ($\mathcal{K}$): is a probabilistic polynomial time algorithm, which takes as input params and returns two keys; a verification key K_{pub} and a signing key K_{priv}. We write $\langle K_{priv}, K_{pub} \rangle \leftarrow \mathcal{K}(\mathsf{params})$.

Sign (Σ): is a probabilistic polynomial time algorithm which takes as inputs params, a message $M \in \mathcal{M}$ to be signed and a signing key K_{priv}. It outputs a signature $Sig \in \mathcal{S}$. When clear, we write $Sig \leftarrow \Sigma(M, K_{priv})$ to simplify presentation, as before, we omit params.

Verify ($\mathcal{V}$): is a polynomial time algorithm which takes as inputs params, a message $M \in \mathcal{M}$ and verification key K_{pub}, and $Sig \in \mathcal{S}$ as the signature to be verified. It returns valid or invalid. When clear, we write $\{\mathsf{valid}, \mathsf{invalid}\} \leftarrow \mathcal{V}(M, Sig, K_{pub})$.

The output valid should result from applying algorithm $\mathcal{V}$ with inputs k and K_{pub} on a signature Sig generated by using algorithm Σ with inputs k and K_{priv} on message M. We say that the PKS scheme is sound if valid is the output of $\mathcal{V}(M, \Sigma(M, K_{priv}), K_{pub})$ for all $\langle K_{priv}, K_{pub} \rangle \leftarrow \mathcal{K}(\mathsf{params})$. This definition does not encapsulate all types of digital signatures, such as for example, digital signatures with message recovery. Algorithm $\mathcal{V}$ can be probabilistic, in which case it should output valid or invalid for valid or invalid signatures with high probabilities.

3.2.2 The Lack of Authenticity

Figure 3.1 illustrates how a PKE scheme adversary, who is between an encryptor B of a ciphertext, and its decryptor A, can impersonate a honest decryptor A. The adversary achieves this, by replacing A's public key K_{pub} with a false public key

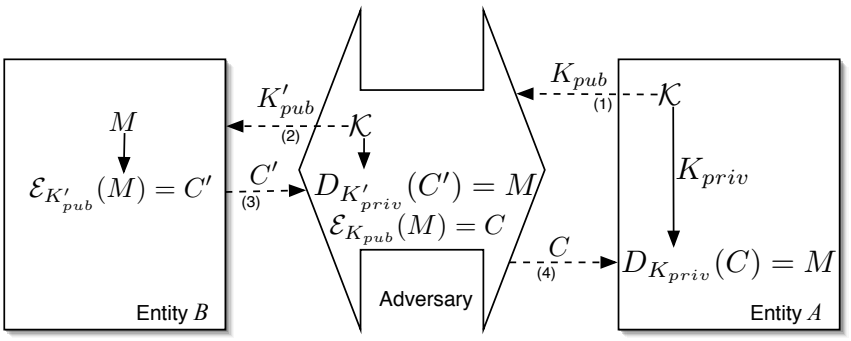

Figure 3.1: A PKE adversary impersonating entity A to B.

K'_{pub}, which is then acquired by B. Similar impersonation settings exist between the signer and verifier in signature schemes.

The following question arises from the need to prevent these kinds of attacks: how does B know (that is, authenticate) which particular public key is A's? To answer this question *data origin authentication* is required. Authenticating public keys means providing assurance (through supportive evidence) to the entity which receives a public key of the entity's identity to which the public key refers. Many data origin authentication methods exist; the usual method for providing authentication of public keys is by using certificates. An alternative method of providing this authentication is achieved by using identity-based public key cryptography (ID-PKC). Next, an explanation of both methods is provided in Sections 3.2.3 and 3.2.4.

3.2.3 PKC with Authenticity of Public Keys from Certificates

The usual way to guarantee the identity and/or identifiers (for example, age, sex or address of the entity) of the public key holder is based on a CA. The CA's digital signature binds entity A's identity and/or identifier ID_A to the corresponding public key, for generality, ID_A will henceforth only be called an identifier. The CA's signature, when sent along with the identifier and public key, forms a digital certificate which can be verified by any entity in possession of the CA's public key. This certi-

ficate provides a binding, assured by the CA, between the identifier and the public key. Digital certificates can contain further information, such as cryptographic algorithms to be used in conjunction with the public key in the certificate. The most widely adopted certificate format is the X.509 standard [149]; it specifies the other fields included (and bound into the certificate by the CA) in the certificates. The CA is the crucial entity for supporting digital certificates in a traditional public key infrastructure (PKI). A PKI is a security infrastructure whose services are implemented to deploy and manage the use of public key cryptography. Basic elements which make up a PKI include services, technology, processes and policies. For a more comprehensive description of PKI consult [1]. A basic certificate issued by a CA for entity A is of the form:

$$\mathrm{Cert}_A = (\mathsf{ID}_A \| K_{pub,A} \| \Sigma(\mathsf{ID}_A \| K_{pub,A}, K_{priv,\mathrm{CA}})).$$

Here, $\Sigma(\cdot, K_{priv,\mathrm{CA}})$ denotes the CA's signature. Entity A's public key is $K_{pub,A}$.

Even though we can achieve secure (that is, confidential and authentic) communication using PKE and PKS schemes alone, asymmetric key agreement protocols play a very important role in PKC and modern applications. This is because the session keys produced by key agreement protocols can be used in symmetric encryption schemes. The latter are generally more efficient than PKE schemes. Furthermore, it is an advantage to have a unique shared session key, because if it is compromised, it does not necessarily affect the security of long term keys. A compromise is more likely for session keys because they are exposed to various applications and machines. The long term key remains safe since it is usually securely stored and accessed through a special interface. Furthermore, the compromise of a session key does not necessarily affect the security of other session keys. In Section 3.3 we discuss some key agreement protocols, and in Section 3.4 we study the properties of key agreement protocols in more detail.

3.2.3.1 An Example: A Version of the ElGamal PKE Scheme with Authenticity of Public Keys from Certificates

Here we will build on the original ElGamal PKE scheme [68], whose security is based on the CDHP. The group used in the original ElGamal scheme is $\mathbb{G} = \mathbb{Z}_p^*$, but we present a scheme in the context of an arbitrary abelian group, $\mathbb{G}$, which may be derived from an elliptic curve. Additionally, we will be presenting the scheme in a standard certificate setting and will be using a hash function, denoted H_5. This non-standard example will familiarise the reader with a building block for the scheme in Chapter 5 and will make concrete some PKE concepts.

The PKE scheme with a certified public key is constructed using four algorithms: Setup, Key-Generation, Encrypt and Decrypt. The functions for all four algorithms are described below.

Setup: This algorithm has input k and runs as follows: (i) generate output $\mathbb{G}$, and additive group of some large prime order q; (ii) choose an arbitrary generator $P \in \mathbb{G}$; (iii) select a CA-key s_{CA} uniformly at random from $\mathbb{Z}_q^*$ and set $P_{\mathrm{CA}} = s_{\mathrm{CA}}P$; (iv) choose a cryptographic hash function $H_5 : \mathbb{G} \to \{0,1\}^n$. Here n will be the bit-length of plaintexts. Properties of cryptographic hash functions are covered in Section 3.5.1.

The system parameters params define the collection of publicly known parameters which are specific to a cryptographic scheme. For this scheme the system parameters are $\mathsf{params} = \langle \mathbb{G}, n, P, P_{\mathrm{CA}}, H_5 \rangle$. Included in the parameters for this certificate setting is P_{CA}, where the CA-key is $s_{\mathrm{CA}} \in \mathbb{Z}_q^*$. For the time being, we simply assume that params is authentically available to all parties, later on we will indicate how. The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = \mathbb{G} \times \{0,1\}^n$.

Key-Generation: For entity A, this algorithm runs as follows: (i) Choose a random $x \in \mathbb{Z}_q^*$; (ii) set the private key to be $K_{priv,A} = x$; (ii) set the public key to be $K_{pub,A} = P_A = xP \in \mathbb{G}^*$.

Note that to use a standard certificate-based PKE scheme, entity A must now be issued a certificate which has to be acquired and verified by entity B before encryption. This is an integral part of any standard certificate-based scheme. The certificate for this scheme will shown in detail later.

Encrypt ($\mathcal{E}$): To encrypt $M \in \mathcal{M}$ for entity A with public key $P_A \in \mathbb{G}^*$, perform the following steps: (i) choose a random value $r \in \mathbb{Z}_q^*$; (ii) compute and output the ciphertext: $C = \langle rP, M \oplus H_5(rP_A) \rangle$.

Decrypt ($\mathcal{D}$): Suppose $C = \langle U, V \rangle \in \mathcal{C}$. To decrypt this ciphertext using the private key $K_{priv,A}$, compute and output: $V \oplus H_5(xU)$.

It is easy to see that if $C = \langle U = rP, V \rangle$ is equal to $\mathcal{E}(M, P_A)$, then $\mathcal{D}(C, x) = M$. The security of this scheme will be discussed in Chapter 5.

Before using the encryption algorithm entity A must register and be granted by the CA a certificate of the form:

$$\mathrm{Cert}_A = \mathsf{ID}_A \| P_A \| \Sigma(\mathsf{ID}_A \| P_A, s_{\mathrm{CA}}).$$

This certificate is granted once the CA checks that entity A knows the private key of P_A or checks that A is the legitimate owner of the P_A. We assume that all entities have an authentic copy of P_{CA}, for example, a root CA is embedded in their system. Either certification hierarchies (for example, certificate chains) or cross certificates may be used in practice to facilitate interoperability. This is so that certificates issued by one CA can be verified by entities certified by other CAs. For a detailed discussion on certification topologies and certificate trust models see [114, §13.6] or [1, Chap.9]. Entity B will acquire Cert_A from either entity A or a public directory. The certificate Cert_A is then used in order to verify the authenticity of the public key P_A. The verification algorithm run by entity B is of the form $\mathcal{V}(\mathrm{Cert}_A, P_{\mathrm{CA}})$. Entity B uses P_A for encryption only if the output of $\mathcal{V}(\mathrm{Cert}_A, P_{\mathrm{CA}})$ is valid.

3.2.4 Identity-based PKC

Shamir [135] was the first to show that the authenticity problem in public key cryptography can be solved without the use of certification. The concept was named identity-based public key cryptography (ID-PKC) by Shamir and has subsequently also became known as identifier-based public key cryptography in some circles. In ID-PKC, entity A's public key K_{pub} is not delivered to entity B. This eliminates the attack presented in Figure 3.1. Rather in ID-PKC entity B encrypts a message for entity A or verifies a signature from entity A using a public key which is derived from only entity A's identifier $\mathsf{ID}_A \in \{0,1\}^*$. The TA has a new role in ID-PKC, and is renamed the Private Key Generator (PKG) to reflect this. The role of the PKG is to issue the private key corresponding to the public key (derived from the identifier ID_A) to entity A. This issuing only occurs after entity A is authenticated by the PKG. To generate private keys the PKG makes use of a master-key which must be kept secret. The requirement to have an authentic CA public key is replaced by the requirement to have authentic PKG parameters. Notice that both the PKG and the entity know the private key.

In his 1984 paper [135], Shamir was only able to construct a concrete identifier-based public key signature (ID-PKS) scheme. Developing a concrete satisfactory identifier-based public key encryption (ID-PKE) scheme remained an open problem. Of note were the solutions designed to solve this problem which required: tamper resistant hardware [57]; non-colluding users [145]; a public directory [121]; high computation complexity for each encryption [146]; very high computation cost by the PKG for each private key generation [86, 111].

The year 2001 witnessed the publication of two different ID-PKE schemes. An ID-PKE scheme was put forward by Cocks [54][3.2]. The security of Cock's scheme is based on the Quadratic Residuosity problem. Providentially, the Quadratic Residuosity problem is a well studied number theoretic problem. Due to message expansion, however, the scheme has a high communication overhead. Another ID-PKE scheme

[3.2] Cocks' ID-PKE scheme in [54] is said to have been discovered four years earlier.

was proposed by Boneh and Franklin [32]. The Boneh-Franklin encryption scheme (abbreviated to BF ID-PKE scheme) used the pairing techniques described in Section 2.3 and is very efficient. This scheme is described next.

3.2.4.1 An Example: The BF ID-PKE Scheme

The BF ID-PKE scheme was the first fully practical and secure ID-PKE scheme. It has much in common with the version of the ElGamal encryption scheme in Section 3.2.3.1. The security of BF ID-PKE [32] was proven by Boneh and Franklin to be based on the hardness of the BDHP. The scheme is constructed using four algorithms: Setup, Extract, Encrypt and Decrypt. The functions for all four algorithms are described in [32]. Here, we will provide a basic exposition of the scheme.

Setup: This algorithm runs as follows: (i) run $\mathcal{IG}$ on input k to generate output $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ as described in Section 2.3; (ii) choose an arbitrary generator $P \in \mathbb{G}_1$; (iii) select a master-key s uniformly at random from $\mathbb{Z}_q^*$ and set $P_0 = sP$; (iv) choose cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1^*$ and $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^n$. Here, n is the bit-length of plaintexts.

The system parameters are params$= \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_0, H_1, H_2 \rangle$. The master-key is $s \in \mathbb{Z}_q^*$. The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^n$.

To use the scheme, entity B creates entity A's public key from A's identifier ID_A. This is done by computing $H_1(\mathsf{ID}_A) \in \mathbb{G}_1^*$ as we will see in the encryption algorithm next.

Encrypt ($\mathcal{E}$): To encrypt $M \in \mathcal{M}$ for entity A with identifier $\mathsf{ID}_A \in \{0,1\}^*$, perform the following steps: (i) compute $Q_A = H_1(\mathsf{ID}_A) \in \mathbb{G}_1^*$; (ii) choose a random value $r \in \mathbb{Z}_q^*$; (iii) compute and output the ciphertext: $C = \langle rP, M \oplus H_2(\hat{e}(Q_A, P_0)^r) \rangle$.

The group element Q_A when used with params is the public key K_{pub} for user A in this scheme. For entity A to decrypt the ciphertext C, it first needs to obtain his private key from the PKG. The PKG uses the algorithm Extract, described next, to generate A's private key.

Extract: For $\mathsf{ID}_A \in \{0,1\}^*$ this algorithm runs as follows: (i) compute $Q_A = H_1(\mathsf{ID}_A) \in \mathbb{G}_1^*$; (ii) set private key to be $d_A = sQ_A$.

Decrypt ($\mathcal{D}$): Suppose $C = \langle U, V \rangle \in \mathcal{C}$. To decrypt this ciphertext using the private key d_A, compute and output: $V \oplus H_2(\hat{e}(d_A, U))$.

The value $\hat{e}(Q_A, P_0)^r$ used in encryption is the same as the value $\hat{e}(d_A, U)$ used in encryption since, using bilinearity,

$$\hat{e}(Q_A, P_0)^r = \hat{e}(Q_A, sP)^r = \hat{e}(Q_A, P)^{rs} = \hat{e}(sQ_A, rP) = \hat{e}(d_A, U).$$

Therefore, if $C = \langle U = rP, V \rangle$ is output by $\mathcal{E}(M, Q_A)$ for entity A then $\mathcal{D}(C, d_A)$ outputs M.

The above ID-PKE scheme is labelled BasicIdent in [32] and is used as a building block for a more complicated scheme which is labelled FullIdent in [32]. The scheme FullIdent is the scheme used in practice because it is proven secure in a stronger model. We will be compare ID-PKC to standard certificate-based PKC (and certificateless PKC) and discuss properties of ID-PKC such as the PKG key escrowing capabilities in Chapter 4.

3.2.5 Cryptographic Workflow

ID-PKC schemes enjoy the property that an entity's private key can be determined *after* its public key has been generated and used. This is a useful feature. An entity B can encrypt a message for A using A's identifier string, ID_A, of B's choice. This identifier should contain A's identity, but might also contain conditions (attributes

or actions) that A must satisfy before the PKG will deliver the corresponding private key.

This condition, for example, could be that A has a valid driver's licence. The encrypted message then could be A's new insurance document. In this way, B can create a *cryptographic workflow* that A must carry out before being able to access some piece of information (for example, the insurance document). The cryptographic workflow is a sequence of operations (for example, authentications) that need to be performed by an entity in order to achieve a certain goal[3.3]. This kind of application cannot be easily supported using traditional certificate-based systems, as the temporal ordering of private key before public key and public key before certificate (which needs to be distributed) are fixed in those systems.

Forcing A to visit multiple TAs [49], or combining sets of private keys to satisfy a set of conditions using a single TA [143], are innovative applications of ID-PKC's cryptographic workflow. In the following, we will show a simple application of the workflow concept which does not use the more sophisticated techniques introduced in [49, 143]. These later techniques are discussed further in the survey in Section 3.6.

3.2.5.1 An Example Use of an ID-PKE Scheme

In the scenario below we will exploit the cryptographic workflow property of ID-PKE schemes.

Problem: Entity B, who is a software merchant (for example, company.com), wants a simple solution for distributing the serial-number of the company's software to a customer A. Entity B wants to keep the serial-number confidential, that is, entity B intends to mitigate its liability and risk of serial-number exposure. Furthermore, B

[3.3] Other published examples using ID-PKC's flexibility can be found in [32, §1.1.2] (that is, delegation of decryption keys) and [49, 143]. Furthermore, HP Laboratories' Trusted Systems Lab recently provided demonstrations using ID-PKE of data tagging (a way of securing data which is independent of software, where tags reflect policies) and role-based/time-based/group-based encryption.

wants to allow entities without credit cards to purchase and access the serial-number in an online fashion. This is so the serial-number needed to run the software is obtained as soon as payment is made.

Solution: Using an ID-PKE scheme, B simply encrypts the serial-number with the customer's identifier containing the condition that the customer pays amount \$X into B's account. That is, the identifier is set to '$\mathsf{ID}_A \| \texttt{paid}\ B\ \texttt{\$X}$'. The PKG (for example, a bank that is prepared to accept payments from A for B), will only provide A with the private key matching the above identifier d_A once A has provided evidence of being entity A and the transaction (or receipt of the transaction) has gone through. The private key is used to decrypt the ciphertext containing the serial-number.

Analysis: The ID-PKE scheme allows the temporal ordering of algorithms to be changed. This is used to realise a cryptographic workflow with two conditions to be satisfied: an activity; forcing A to perform a \$X transaction, and an attribute; A is who he claims to be. Entity B tailors a workflow for A to follow and the PKG acts as the workflow enforcer.

3.3 Cryptographic Key Agreement Protocols

Before presenting some key agreement protocols, which will be commonly referred to in thesis, we further explore the concept of protocols provided in Section 3.1, and specifically key agreement protocols as described in [114, Chap.12.2].

The class of protocols whereby a shared secret becomes available to two or more parties for subsequent cryptographic use are known as key establishment protocols. Key establishment is further subdivided into key transport and key agreement. In key transport, one party creates or obtains a secret value and securely transfers it to the other parties. For example, the PKG in the ID-PKE scheme presented in Section 3.2.4.1 acts as a key transport agent, securely transporting the private key d_A to entity A. Key transport is not considered here in any more detail.

Definition 3.1 ([114]) A *key agreement protocol* is a key establishment technique in which a shared secret is derived by two (or more) parties as a function of information contributed by, or associated with, each of the parties, (ideally) such that no party can pre-compute the resulting value.

3.3.1 General Attack Classifications

An attack occurs when the intended goals of a protocol are not met or the desired security attributes do not hold. A *passive* attack occurs when an adversary can prevent the protocol from accomplishing its goals by simply observing the protocol runs. In contrast, an *active* attack is one in which the adversary may delete, inject, alter or redirect messages, or interleave multiple instantiations of the same protocol and the like.

Next, both the original Diffie-Hellman Protocol [58] and a simplified version of Joux's protocol [90] are presented. The goal of these protocols is to provide *good keys*. This goal states that the key is selected uniformly at random from the key space, so that no adversary has an information-theoretic advantage when mounting a guessing strategy to determine the key.

3.3.2 The Diffie-Hellman Protocol

Diffie and Hellman [58] revolutionised cryptography by introducing the first key agreement protocol not based on shared secrets. In this section, we consider the original Diffie-Hellman protocol.

Let k be a security parameter that determines the size of a large prime p, and in what follows, g denotes a generator of $\mathbb{Z}_p^*$. In the Diffie-Hellman protocol, we assume that entities A and B share the common values g and p (embedded in the system or acquired via some other mechanism). Then integers a and b, where $1 \leq a, b \leq p-2$,

are selected uniformly at random by entities A and B respectively. The ordering of protocol messages is irrelevant and either entity can initiate the protocol. The message flows are given in Figure 3.2. We choose to present the original variant of this protocol because we will be using it as a building block for the protocols in Section 9.1.2.1 and Section 9.8.2.

Protocol Messages

1. $A \rightarrow B: \quad g^a \bmod p$
2. $B \rightarrow A: \quad g^b \bmod p$

Figure 3.2: The Diffie-Hellman protocol.

Protocol description: Entity B computes $K_B = (g^a)^b \bmod p$ after obtaining message *1* in Figure 3.2 and A computes $K_A = (g^b)^a \bmod p$ once the communication in Figure 3.2 is complete. The values of K_A and K_B are both equal to $K_{AB} = g^{ab} \bmod p$. This can serve as the secret key shared by A and B. The values a and b should be deleted at the end of the protocol run.

The protocol's achievement of agreeing a good key in the face of passive adversaries could be related to the hardness of either the CDHP or the DDHP in $\mathbb{Z}_p^*$, depending whether a hash function (modelled as a random oracle – see Section 3.5.2) is used to derive a key or not. To make this claim concrete requires the development of an appropriate security model, such models are the topic of Chapter 9.

The Diffie-Hellman protocol can be extended to three parties. For three parties it takes two rounds and six broadcasts to establish a key. The first three message broadcasts are transmitted in the first round and the rest of the protocol broadcasts are transmitted in the next round. As in the two party case, we assume all participants here agree on suitable parameters g and p in advance. The message flows of this protocol are given in Figure 3.3.

Protocol Messages

1. $A \rightarrow B, C: \quad g^a \bmod p$
2. $B \rightarrow A, C: \quad g^b \bmod p$
3. $C \rightarrow A, B: \quad g^c \bmod p$
4. $A \rightarrow B, C: \quad g^{ba} \bmod p \| g^{ca} \bmod p$
5. $B \rightarrow A, C: \quad g^{ab} \bmod p \| g^{cb} \bmod p$
6. $C \rightarrow A, B: \quad g^{ac} \bmod p \| g^{bc} \bmod p$

Figure 3.3: A three party Diffie-Hellman protocol.

Protocol description: After the first three broadcasts of Figure 3.3, entity A computes $(g^b)^a \bmod p$ and $(g^c)^a \bmod p$, B computes $(g^a)^b \bmod p$ and $(g^c)^b \bmod p$ and C computes $(g^a)^c \bmod p$ and $(g^b)^c \bmod p$. Once the protocol in Figure 3.3 is complete K_A, K_B and K_C are computed by A, B and C respectively where K_A, K_B and K_C are all equal to $K_{ABC} = g^{abc} \bmod p$. This value can serve as the secret key shared by A, B and C. The values a, b and c should be deleted at the end of the protocol run.

3.3.3 Joux's Protocol

Joux [90] introduced a very simple and elegant one-round protocol in which the secret session key for three parties could be created in a single round using three broadcasts. Joux's protocol simplifies the Diffie-Hellman protocol extension shown in Figure 3.3 and is used to establish a shared secret key with minimal communication complexity. The protocol makes use of pairings on elliptic curves and requires each entity to transmit only a single broadcast message containing some public value. This should be contrasted with the obvious extension of the Diffie-Hellman protocol to three parties in Figure 3.3, which requires six rounds and six broadcasts.

We assume that A, B and C share the common values $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$, which are determined by the security parameter k. In Joux's protocol, P is the generator of the group $\mathbb{G}_1$ of prime order q as specified in Section 2.3 and $a, b, c \in \mathbb{Z}_q^*$ are selected uniformly at random by A, B and C respectively. As in the Diffie-Hellman protocol in Figure 3.2, the ordering of protocol messages is irrelevant and any of the three entities can initiate the protocol. The message flows are given in Figure 3.4.

Protocol Messages

1. $A \rightarrow B, C: \quad aP$
2. $B \rightarrow A, C: \quad bP$
3. $C \rightarrow A, B: \quad cP$

Figure 3.4: Joux's one round protocol.

Protocol description: Once the communication in Figure 3.4 is complete, A computes $K_A = \hat{e}(bP, cP)^a$, B computes $K_B = \hat{e}(aP, cP)^b$ and C computes $K_C = \hat{e}(aP, bP)^c$. By bilinearity of $\hat{e}$, K_A K_B and K_C are all equal to $K_{ABC} = \hat{e}(P, P)^{abc}$. This can serve as the secret key shared by A, B and C.

Although not explicitly mentioned in [90], the success of this protocol in achieving its aim of agreeing a good key for the three entities in the face of passive adversaries can be related to the hardness of either the BDHP or the DBHDP. As is the case with the two party Diffie-Hellman protocol, depending on how the key is derived the protocol relies on either the computational or decisional problem.

The reader will notice that our version of Joux's protocol is simpler than the original. It uses a modified pairing which allows us to avoid sending two points per participant. This modification of Joux's protocol was first performed by Verheul [147, §5.1].

Returning to encryption schemes, the ElGamal encryption scheme [68] can be viewed

as a Diffie-Hellman protocol [58] in key transfer mode [114]. A more complicated informal argument illustrates why compromising the basic BF ID-PKE scheme is related to compromising Joux's protocol [90]. This informal argument forms the basis for a formal proof of security. The argument can be understood by considering an instance of Joux's protocol in which the three public values exchanged by A, B and C are sP, rP and $Q_A = \xi P$ respectively, and the session key agreed is $K_{ABC} = \hat{e}(P, P)^{\xi rs}$. Here, we can think of entity A as the decryptor, entity B as the encryptor and entity C as the trusted PKG. To obtain the Boneh and Franklin encryption of message M, the session key K_{ABC} is input to a hash function and the result is XORed with M. In order to decrypt M, the problem an attacker needs to solve is: given $\langle P, rP, sP, \xi P\rangle$ determine $K_{ABC} = \hat{e}(P, P)^{\xi rs}$. This is an instance of the BDHP. Thus, supposing the BDHP is hard, to determine $\hat{e}(P, P)^{\xi rs}$ either r, s or ξ must be known. The PKG, with the knowledge of s, can compute $K_C = \hat{e}(\xi P, rP)^s = K_{ABC}$. Entity B, with the knowledge of r, computes $K_B = \hat{e}(\xi P, sP)^r = K_{ABC}$ during encryption. For decryption, entity A cannot compute K_{ABC} on its own. This is because ξ is unknown to all entities because of the way it is computed by hashing ID_A; obtaining ξ from ξP is equivalent to solving the DLP in $\mathbb{G}_1$. The PKG, however, furnishes A with $d_A = s\xi P$. Notice that d_A contains both s and ξ. Hence, A unlike in Joux's protocol (which uses one private value and two public values) computes K_{ABC} using one private value, $s\xi P$ (A's private key), and one public value, rP, by computing $K_A = \hat{e}(s\xi P, rP) = K_{ABC}$. Compromising $d_A = sQ_A = s\xi P$ compromises only entity A's private key because every entity has a unique ξ based on its identifier ID.

3.3.3.1 Bit Security

More properly, in Joux's protocol the session key should be derived by applying a suitable key derivation function. The key derivation function, denoted KDF, should be used on the quantity $\hat{e}(P, P)^{abc}$, thus, $K_{ABC} = \mathsf{KDF}(\hat{e}(P, P)^{abc})$. For otherwise, an attacker might be able to get partial information about session keys even if the BDHP is hard. Note that knowing some bits of $\hat{e}(P, P)^{abc}$ does not necessarily

enable the attacker to find $\hat{e}(P,P)^{abc}$ completely, so we have not contradicted the BDH assumption.

Bit security results of [75] suggest that a KDF for Joux's protocol can be constructed by taking the most significant bits of the trace of $\hat{e}(P,P)^{abc}$. The resulting key is no less secure than using all the bits of $\hat{e}(P,P)^{abc}$ as the key. One might also consider using a one way function such as a hash function (defined in Section 3.5.1) as the KDF. The disadvantage of using hash functions, compared to using the trace, is that hash functions are generally less efficient and it is harder to establish security. A similar argument encouraging the use of a key derivation function can be made for the Diffie-Hellman protocol. For examples of key derivation functions see [9, 87].

3.3.3.2 Man-in-the-Middle Attacks

Unfortunately, just like the unauthenticated two-party Diffie-Hellman protocol, Joux's protocol is only secure in the face of a passive adversary. In many practical applications, this does not model well the capabilities of a real world adversary.

In a more realistic model, the adversary is active, making both the Diffie-Hellman protocol and Joux's protocol susceptible to powerful impersonation attacks, similar to those presented in Figure 3.1. This is a textbook man-in-the-middle attack on protocols which do not have a mechanism to authenticate their users. The attack allows an adversary to masquerade as any entity to any other entity in the network since it is assumed that all the network traffic goes via the adversary.

Here we present a man-in-the-middle attack on Joux's protocol. For protocols, let E denote an adversary who replaces the public values from A to B and B to A with $a'P, b'P \in \mathbb{G}_1$. Here $a', b' \in \mathbb{Z}_q^*$ are chosen by E. In what follows, E_A indicates that E is impersonating A by sending or receiving messages intended for or originating from A.

1. Entity A sends aP to $E_{B,C}$.
2. The adversary E_A initiates a run of Joux's protocol by sending $a'P$ to B and C.
3. Entity B sends bP to E_A and C; C sends cP to E_A and B.
4. The adversary E_B forwards $b'P$ instead of bP to A and E_C simply forwards cP to A.

Entities B and C (following the protocol) compute $K_{E_ABC} = \hat{e}(P,P)^{a'bc}$. Entity A (following the protocol) computes a key $K_{AE_BC} = \hat{e}(P,P)^{ab'c}$. Since E can compute K_{AE_BC} and K_{E_ABC}, E can read all the traffic and can masquerade as any of A, B or C to the other two entities, that is, E can also impersonate C to A. Impersonations are performed by simply decrypting/re-encrypting (to and from A), deleting, replacing, decrypting/re-encrypting (to and from C) and/or injecting (by encrypting) messages.

Solutions to this problem for the Diffie-Hellman protocol are well known. In Chapter 9 we consider how the security of Joux's protocol can be enhanced to prevent man-in-the-middle and other types of attacks. In preparation for this, we next provide definitions of protocol goals and protocol attributes.

3.4 Authenticated Key Agreement Protocol Goals and Attributes

Here we discuss the various security attributes and goals for key agreement protocols. A security goal is an essential property that a protocol should possess. Every protocol should be designed with specific security goals in mind. Based on application, however, the importance of a security attribute spans from an essential requirement on par with a security goal, to a dispensable property, inessential for the key agreement protocol to possess.

3.4.1 Extensional Security Goals

An *extensional* goal [37, 129] for a protocol is defined to be a design goal that is independent of the protocol details. Below, three desirable and widely-agreed extensional goals for key agreement protocols are listed. A further discussion of these can be found in [114, Chapter 12]. The first goal we try to achieve is *implicit key authentication*. This goal, if met, assures an entity that only the intended other entities *can* compute a particular key. This level of authentication results in what is known as an *authenticated key agreement* (AK) protocol. *Explicit key authentication* is the second desirable goal. This goal is met if each entity is also assured that the intended other entities *have* actually computed the key. The resulting protocol is called an *authenticated key agreement with confirmation* (AKC) protocol. The final goal is that the protocol provides a good key as we described in Section 3.3.1.

In the context of public key cryptography, *short-term public values* are generally only used once to establish a session and are sometimes called *ephemeral keys*. Conversely, *long-term public keys* are static keys used primarily to authenticate the protocol's participants.

3.4.2 Security Attributes

A number of desirable security attributes have been identified for key agreement protocols [26, 27, 99] and our definitions are borrowed from these sources. Depending on the application scenario, these attributes can be vital in excluding realistic attacks.

Known session key security: A protocol is *known session key secure* if it still achieves its goals in the face of an adversary who has learnt some previous session keys.

(Perfect) forward secrecy: A protocol enjoys *forward secrecy* if, when the long-term private keys of one or more entities are compromised, the secrecy of

previous session keys remains unaffected. *Perfect* forward secrecy refers to the scenario when the long term private keys of all the participating entities are compromised.

No key-compromise impersonation: Suppose A's long-term private key is disclosed. Then of course an adversary can impersonate A in any protocol in which A is identified by this key. We say that a protocol resists *key-compromise impersonation* when this loss does not enable an adversary to impersonate *other* entities to A as well and obtain the session key.

No unknown key-share: In an *unknown key-share attack*, an adversary convinces a group of entities that they share a key with the adversary, whereas in fact, the key is shared between the group and another party. This situation can be exploited in a number of ways by the adversary when the key is subsequently used to provide encryption or integrity [93].

No key control: It should not be possible for any of the participants (or an adversary) to force the session key to a preselected value or predict the value of the session key. For a discussion of how protocol participants can partially force the values of keys to particular values and how to prevent this using commitments at the expense of extra protocol rounds see Mitchell *et al.* [115].

Some of the properties presented here are formalised in the context of security models to be presented in Section 9.3.

3.4.3 Further Attributes

It is desirable to reduce the computational, communicational, storage and system complexities of any protocol. As will become evident from our examination of Joux's protocol in Chapter 9, communication advantages that a protocol apparently possesses can disappear when one considers either a different network architecture or more stringent security requirements. Additionally, timestamps although a crucial

part of a PKI for documentation and legal use [3, 40, 88], can in certain circumstances be undesirable in authentication protocols due to their implementation difficulties. Essentially the difficulties arise due to complexity in synchronisation and the inappropriateness of 'relative' time in a multi-clock setting, see [59, 125] for further details.

A protocol is *role symmetric* when messages transmitted and computations performed by all the entities have the same structure.

3.5 Provable Security Basics

By examining indistinguishability and semantic security, Goldwasser and Micali [79] introduced the provable security paradigm. Instead of using an information theoretic framework, provable security is based on complexity theory. This is because modern cryptography assumes that the adversary attacking the cryptographic scheme or protocol is furnished with limited resources. Theoreticians and standards bodies now view a 'provably secure' scheme very favourably; some consider it a crucial attribute for any scheme.

To bound the adversary's resources, semantic security makes use of security parameters, as introduced in Section 2.4. The adversary is modelled as an algorithm, interacting with a challenger and/or oracles simulating participants in the system. The schemes that are labelled as *secure* are only secure with respect to all polynomial time (in security parameter k) adversaries.

An outline of the process that is followed in order to obtain a proof of security for a scheme or protocol, is as follows: (*i*) Provide a formal definition the goal(s) of the scheme or protocol. (*ii*) Provide a formal adversarial model (the access model). (*iii*) Define what it means for the scheme or protocol to be secure (the attack goal(s) it should withstand). For examples of this see Section 3.5.3. (*iv*) Provide a proof of security for the scheme or protocol by 'reducing' to a known hard computational

problem. This reduction is explained in what follows.

The reduction shows that the adversary can be transformed to an algorithm that solves a computational problem that is known to be hard. It does so by simulating the adversary's attack environment. The success probability of solving the hard computational problem can be related to that of the adversary. The very assumption that the computational problem is hard (in the sense of there being no polynomial time algorithm in k which can solve it) shows that no adversaries with non-negligible probability of success can exist.

The security proofs in this thesis require the hash functions used in our schemes and protocols to be instantiated by random oracles. First formulated by Bellare and Rogaway [21], this approach allows for the scheme or protocol to be provably secure in the random oracle model. To understand the origins of this model we next turn to hash functions.

3.5.1 Cryptographic Hash Functions

Cryptographic hash functions are functions which have many uses in cryptography. Hash functions can play an important role in implementing and/or proving the security of PKS schemes, PKE schemes and key agreement protocols.

A hash function H is an efficiently computable algorithm that maps an input x of arbitrary finite bitlength, to an output $H(x)$ of fixed bitlength n. We next list the properties of a Collision Resistant Hash Function (CRHF), H, with inputs x, x' and outputs y, y' in the same manner which is described in [114, §9.2.2]:

1. *preimage resistance*: for essentially all pre-specified outputs, it is computationally infeasible to find any input which hashes to that output, i.e., to find any preimage x' such that $H(x') = y$ when given any y for which a corresponding input is not known.

2. *2nd-preimage resistance*: it is computationally infeasible to find any second input which has the same output as any specified input, i.e., given x, to find a 2nd-preimage $x' \neq x$ such that $H(x) = H(x')$.

3. *collision resistance*: it is computationally infeasible to find any two distinct inputs x, x' which hash to the same output, i.e., such that $H(x) = H(x')$. (Note that here there is free choice of both inputs.)

A CRHF is also sometimes called a strong one way hash function. If the last condition is not satisfied, the hash function is considered a weak one-way hash function or a One-Way Hash Function (OWHF). The above description of hash functions although informal suffices for this thesis. This is because we would not be using the above definition of a hash function directly in our security proofs, we will return to this topic in the next section. For relations and separations between formal definitions from a provable-security viewpoint see Rogaway and Shrimpton [128]. For a brief survey of the publications which discuss hash function security-notions see [128, Appendix A].

A cryptographic hash function $H : \mathcal{X} \rightarrow \mathcal{Z}$ is usually designed to act as a compression function, mapping elements in $\mathcal{X} = \{0,1\}^*$ to elements in $\mathcal{Z} = \{0,1\}^n$. The hash function can also be designed to map elements of one group to elements of another group. In practice, however, mapping between groups is difficult and may require mapping to some intermediate set and using some deterministic encoding operations to map to and from the groups. Such a construction was shown for a hash function mapping the set of binary strings $\{0,1\}^*$ into a group $\mathbb{G}_1^*$ in [33, §4.3] and [36, §3.2].

3.5.2 Random Oracles

To provide a better security assurance than a heuristic one, the concept of ideal hash functions was introduced. Ideal hash functions are functions whose outputs are computationally indistinguishable from a random output. The heuristic step in the random oracle methodology is replacing this ideal H with a member of the family of all truly random functions from $\mathcal{X}$ to $\mathcal{Z}$, chosen uniformly at random. Since all

queries to H are answered by selecting an output at random, H is now effectively selected uniformly at random from the family of all functions. When queried on the same input the oracle is defined to produce the same output, since H (its analogy) will behave this way in the real world. Hence, in the random oracle model, no adversary can make use of the underlying structure of the real hash function.

On one hand, provided the adversary has no insight into H, using this black box idealised approach to model hash functions clearly captures the security essence of the overall scheme or protocol. Moreover, the abstraction allows for simple efficient protocols and schemes to be designed and proved secure. On the other hand, critics of this abstraction argue that, in the real world, no single deterministic polynomial time function can provide a good implementation of the random oracle. In other words, they argue that the random oracle methodology is flawed. For more detailed expositions see [21, 42].

3.5.3 Security notions for PKE

In this section, we define notions of security for standard PKE schemes. First we define the notion of one-way encryption (OWE), which is a weak notion of security. In all the definitions, there are two parties, the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$.

One-way encryption security for PKE: We say that a PKE scheme is OWE secure if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in the following game:

Setup and Challenge: The challenger takes a security parameter k as input and runs both the Setup and Key-Generation algorithms. The challenger picks a random plaintext $M \in \mathcal{M}$ and computes C^*, the encryption of M under a public key, K_{pub}, output by the Key-Generation algorithm. It gives $\mathcal{A}$ the resulting parameters params and a public key K_{pub} and the ciphertext C^*.

Guess: After performing some computations, $\mathcal{A}$ outputs a guess $M' \in \mathcal{M}$. The adversary wins the game if $M = M'$. We define $\mathcal{A}$'s advantage in this game to be $\text{Adv}(\mathcal{A}) := \Pr[M = M']$.

The probability is measured over the random bits used by the challenger and the adversary. Next, we define a stronger notion than OWE security. The next definition of security for a PKE scheme involves indistinguishability of encryptions against a fully-adaptive chosen ciphertext attacker (the *goal-access model* pair which corresponds this security notion is IND-CCA[3.4]) [16, 64, 126].

Chosen ciphertext security for PKE: We say that a PKE scheme is semantically secure against an adaptive chosen ciphertext attack ("IND-CCA secure") if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in the following game:

Setup: The challenger takes a security parameter k as input and runs both the Setup and Key-Generation algorithms. It gives $\mathcal{A}$ the resulting system parameters params and a public key K_{pub} output by the Key-Generation algorithm.

Phase 1: Adversary $\mathcal{A}$ may make decryption queries on ciphertexts of its choice.

Challenge Phase: Once $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length plaintexts $M_0, M_1 \in \mathcal{M}$, where $M_0 \neq M_1$. The challenger now picks a random bit $b \in \{0, 1\}$ and computes C^*, the encryption of M_b under the public key K_{pub}. Ciphertext C^* is delivered to $\mathcal{A}$.

Phase 2: Now $\mathcal{A}$ may make further decryption queries as in Phase 1. However, no decryption query can be made on the challenge ciphertext C^* for public key K_{pub} that was used to encrypt M_b.

[3.4] Some authors label CCA as CCA2 (the notion of IND-CCA2 was introduced by [126]), to contrast it with the non-adaptive chosen ciphertext attack (CCA1) model (the notion of IND-CCA1 was introduced by [120]). The CCA1 model does not allow the adversary access to the decryption oracle after being offered a challenge ciphertext.

Guess: Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. The adversary wins the game if $b = b'$. We define $\mathcal{A}$'s advantage in this game to be $\mathrm{Adv}(\mathcal{A}) := 2|\Pr[b = b'] - \frac{1}{2}|$.

The probability is measured over the random bits used by the challenger and the adversary. The notion of indistinguishability of encryptions against chosen plaintext attack (IND-CPA) is defined in the same way as IND-CCA, except that the IND-CPA adversaries are not given any access to a decryption oracle.

3.6 Survey of Pairing-based Schemes

Public Key Cryptography emerged from the ideas of Diffie and Hellman's seminal paper [58]. These ideas have been extended in many ways to develop a number of cryptographic schemes based on the hardness of the CDHP or other hard problems such as the RSA inversion problem [114, §3.3]. A comparable but more rapidly moving trend has recently occurred in the field of elliptic curve pairings. This followed Boneh and Franklin's paper [32], which introduced to the wider research community a practical and provably secure ID-PKE scheme. This efficient and simple ID-PKE scheme allowed for the extension of scope in ID-PKC. Numerous identifier-based schemes were subsequently developed. A classification of this large body of research into pairings is represented in Figure 3.5, where identifier-based publications are represented by rectangular boxes. Non-identifier-based publications are represented by the curved boxes. Publications in dotted boxes are either superseded by newer, improved publications, or considered to have a major flaw. Surveys and publications on the implementation of elliptic curve pairings are not included.

It can be seen in Figure 3.5 that some publications occured before that of Boneh and Franklin [32]; most importantly the work of Sakai, Ohgishi and Kasahara [133], Joux [90] and Verheul [147]. Sakai *et al.* [133] presented a non-interactive identifier-based key agreement protocol and ID-PKS schemes. In the schemes of Sakai *et al.*, a hash function is used to map identifiers to elements of a group $\mathbb{G}_1^*$; this function and the way in which Sakai *et al.* use pairings is found in many subsequent publications. Joux

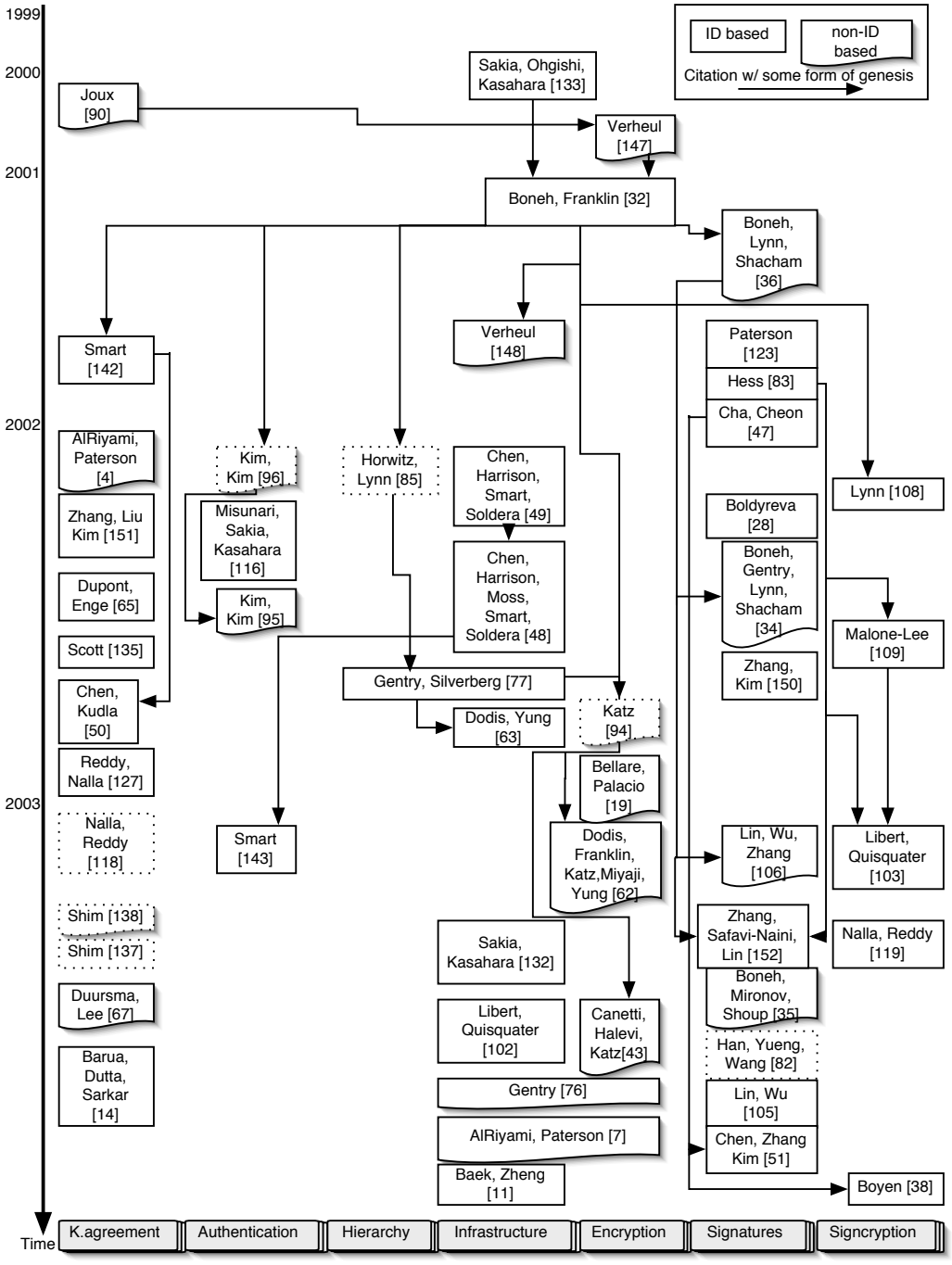

Figure 3.5: Overview of pairing-based publications.

[90] presented a tripartite protocol whose properties and relation to BF ID-PKE weref described in Section 3.3.3. Verheul [147] presented a distortion map which maps a point on the elliptic curve to an unrelated point. The main benefit of doing this was described in Section 2.3. A second benefit of distortion maps is that they ensure that both inputs to the pairing map are in a 'small' group with compact representation of elements. Another benefit is that it produces a simpler representation for the points. This is because only a single group $\mathbb{G}_1$ is required, rather than two different groups (one of which is large), on the left hand side of the pairing map. For example, this improvement is demonstrated in [147] on Joux's protocol where one point rather than two needs to be broadcasted by all three entities. A scheme using a variant of the ElGamal PKE scheme is also presented in [147] using this modified pairing map. This scheme allows for an escrow-able encryption service with only one public key. Sakai, Ohgishi and Kasahara [133] were the first to conceive of and explore the suitability of pairings to construct identifier-based cryptographic schemes. Boneh and Franklin [32], however, were the first to construct a provably secure and efficient ID-PKE scheme, which resulted in the proliferation of ID-PKC. Some concepts from previous publications were developed, modified and formalised to produce the BF ID-PKE scheme, whose basic version was presented in Section 3.2.4.1.

Figure 3.5 captures the relationship between various pairing publications (up to the end of 2003) in cryptography. The schemes in the publications are classified and placed in the most appropriate column category. The categories of the columns are: (*i*) key agreement protocols; (*ii*) authentication schemes; (*iii*) hierarchical schemes; (*iv*) infrastructure related schemes; (*v*) encryption schemes; (*vi*) signature schemes; and (*vii*) signcryption schemes. Arrows illustrate some relationships between publications. Publications at the base of an arrow are cited and thematically originated from the publications at the head of the arrow.

The following descriptions highlight the evolution of pairing-based schemes[3.5]:

[3.5] For an alternative pairing scheme survey see Paterson [122], for ID-PKC surveys which include many pairing schemes see Kudla [98] and Gagné [72].

Key agreement protocols: Several protocols have been proposed. These include identifier-based and group key agreement protocols. Only some of the proposals provide a comprehensive security treatment.

- Unauthenticated protocols: Joux provides an unauthenticated single round broadcast protocol [90] which was presented in Section 3.3.3. Duursma and Lee [67] show how to extend Joux's protocol [90] for use in an unauthenticated group with 3^n entities.
- Identifier-based protocols: Amongst other schemes, Sakai, Ohgishi and Kasahara [133] present a non-interactive identifier-based key agreement protocol. Dupont and Enge [65], unaware of Sakai *et al.*'s [133] work, proposed an analogy of the Sakai *et al.* protocol in the setting of unmodified pairings. Dupont and Enge's protocol [65] is proven secure provided that a certain generalisation of the BDHP is hard, see Section 2.4.2 for details. Smart [142] designed an identifier-based protocol which used two pairing computations and showed how to add key confirmation. This protocol and all of the remaining identifier-based protocols require short-term public keys to be exchanged. Smart's protocol [142] was modified by Chen and Kudla [50] who presented several protocols, each which provided at least one of the following improvements: (*i*) use of only a single pairing computation; (*ii*) forward security against the PKG; and/or (*iii*) interoperation of users with identities registered using different PKGs. Shim [137] provided an alternative protocol to Smart's protocol [142] and Chen and Kudla's protocols [50]. Shim's protocol [137] uses one pairing computation. It has been found, however, to be vulnerable to a man-in-the-middle attack by Sun and Hsieh [144].

  Zhang, Lui and Kim's protocol [151] and Nalla and Reddy's three protocols [118] are tripartite identifier-based key agreement protocols. We observe that these protocols combine ideas from the work of Smart [142] and Al-Riyami and Paterson [4] (see certificate-based protocols below and Chapter 9). The security of Nalla and Reddy's protocols [118], is undermined by a man-in-the-middle attack [139] on the first proposed protocol and more serious passive attacks [52] on the subsequent two protocols.

Binary [127] or ternary trees [14] are used to construct efficient conference identifier-based key agreement protocols. The security of the group protocols is based on the existence of secure identifier-based key agreement protocols. Barua, Dutta and Sarkar's protocol [14] uses two and three party identity key agreement protocols, while Reddy and Nalla's protocol [127] uses only a two party identifier-based key agreement protocol.

Scott [134] presents a different kind of authenticated key exchange whose properties are particularly suited for the Client-Server environment. An entity using Scott's protocol is assumed to be using a token in conjunction with a password.

- Certificate-based protocols: Al-Riyami and Paterson [4] present tripartite certificate-based key agreement protocols whose one round variants do not require signatures. One of the protocols was rendered insecure by Shim [136]; the full attack is covered in Section 9.5.1. In any case, Shim [138] presented an alternative one round tripartite protocol. Shim's protocol is vulnerable to an attack presented by Sun and Hsieh [144]. Moreover, the protocol cannot be implemented as it does not make mathematical sense for reasons that are described in Section 9.6.

Authentication schemes: A traitor tracing mechanism, which can trace authorized entities who give their keys to unauthorised entities in a broadcast encryption scheme, was introduced by Mitsunari, Sakai and Kasahara [116]. The scheme's security is based on the hardness of a problem dubbed the 'k-weak Diffie-Hellman problem', whose hardness remains an open problem.

Kim and Kim [95, 96] present, in two publications, interactive identification (or entity authentication) protocols. The protocols allow entity A to convince an entity B, of A's identifier by proving a private value corresponding to a public value. Proofs of security for the schemes in [95, 96] are also provided. Nevertheless, Zhang, Xu and Feng [154] showed that the identification scheme in [96] is actually insecure against a passive attacker. This attack is always possible because any entity can trivially impersonate the prover using only public information. The private value of the prover is not actually required to

run the protocol successfully. Although the protocol in [95] shares the same key generation method as in [96], the modified protocol actions in [95] between the prover and verifier ensures that the attack by Zhang *et al.* does not work.

Smart [143] shows how to use the BF ID-PKE scheme to construct an access control mechanism using key calculus techniques to broadcast encrypted data. The scheme extends the ideas of Boneh and Franklin [32] and Chen *et al.* [49] (see identifier-based infrastructure) to create a flexible scheme which is appropriate for use in access control structures because the workload is shifted to the decryptor. This work's contributions can also be categorised as being in the area of infrastructure related schemes.

Hierarchical schemes: The notion of an identifier-based hierarchy was introduced by Horwitz and Lynn [85]. The motivation for hierarchical schemes was to improve identifier-based infrastructures by spreading the workload of a PKG. The hierarchy introduced in [85] consisted of two levels: the upper level with total collusion resistance and the lower level with partial collusion resistance. In addition to the partial collusion resistance, another drawback of the scheme of [85] was that the efficiency of key generation and encryption decreased proportionally with the number of entities in the system, thus, it was not truly scalable. The open problem presented by Horwitz *et al.* was 'to construct a two-level hierarchical ID-PKE scheme that is totally collusion-resistant on the lower level and at least partially collusion-resistant on the upper level' [85, p.479]. Gentry and Silverberg [77] solved this problem, by presenting a totally collusion-resistant scheme supporting an arbitrary number of levels which scaled in a natural way. Further improvements were presented in [77] for two users who are close in the hierarchical tree. The improvement, which is an extension of the non-interactive identifier-based key agreement protocol by Sakai *et al.* [133], required the encryptor to use a dual identity form of the hierarchical ID-PKE scheme. Both publications [77, 85] present provably secure ID-PKE schemes and the work of Gentry and Silverberg [77] also extends hierarchy to ID-PKS schemes.

Infrastructure related schemes: The infrastructure surrounding ID-PKC has

been examined by a number of authors [11, 48, 49, 77, 132]. Applications of pairings to certificate-based infrastructures can be found in [76, 148]. We develop a new type of infrastructure in Part II of this thesis. Some work of that part appears in [7].

- Identifier-based infrastructure: The PKG is considered both the bottleneck and single point of failure in an identifier-based public key infrastructure (ID-PKI). By extending the identifier infrastructure created by the ID-PKC scheme of [32], Chen *et al.* [49] provide applications for ID-PKC and demonstrate how an entity can combine private keys obtained from multiple PKGs to form a single working private key. The applications make use of ID-PKC's cryptographic workflow property which was described in Section 3.2.5. Sakai and Kasahara [132] also examine the use of multiple PKGs and introduce an alternative ID-PKC infrastructure to [32]. Sakai and Kasahara [132], however, do not provide any security analysis for any of their efficient schemes. The advantages of using the ideas of [49, 132] instead of multiple ID-PKE (or ID-PKS) schemes using different PKGs one after the other, are that the schemes of [49, 132] offer: (*i*) computational efficiency; (*ii*) decryption (or verification) does not necessarily need to be applied in the opposite order to encryption (or signing); and (*iii*) as a consequence of *i*) and the flexibility of *ii*), cryptographic workflows between the PKGs are practical. We will be re-visiting cryptographic workflows in Section 4.6.3.

  The mechanisms by which ID-PKC's PKGs can operate on more than two levels has been demonstrated in two publications: a pure identifier-based infrastructure [77]; and a hybrid ID-PKC infrastructure using certificate chains as in Chen *et al.* [48] or identifier-based linked TAs. Complementing ID-PKI with traditional PKI at the higher levels of the hierarchy as in [48] is a very practical solution for building a general purpose keying infrastructure because key escrow is eliminated between intermediate TAs, who most probably: (*i*) do not need the flexibility of an ID-PKI; and (*ii*) are furnished with sufficient resources to manage keys using certificates. Novel applications presented in both infrastructures [48, 77] use the short

signature scheme of [36]. The short signature scheme of [36] is described in the certificate-based signature schemes part of this survey below.

The distributed PKG approach discussed by Boneh and Franklin [32] requires PKGs to share the system's master-key. This idea was adapted by Libert and Quisquater in [102] who present a threshold ID-PKE and a mediated ID-PKE scheme based on the BF ID-PKE. The proof of security for the ID-PKE scheme in [102] is proposed in a weaker model than that of the ID-PKE in Boneh *et al.* [32], while the proof of security for the mediated ID-PKE scheme uses the same weak model as that provided in Ding and Tsudik [60][3.6]. Mediated versions of ID-PKC are schemes where an online security mediator (SEM) keeps part of each user's private key. Every decryption and signature generation requires the user to obtain help from the SEM by getting a token related to the user's private key. If the SEM is instructed not to help the user, the user's private key is effectively revoked.

Dodis and Yung [63] and Baek and Zheng [11] offer an alternative threshold solution to the Boneh and Franklin solution to the private key escrow problem by using the distributed PKG approach of threshold cryptography to share private keys of identifiers instead of the system's master-key. The solution by Dodis and Yung [63] makes use of the hierarchical ID-PKE of Gentry and Silverberg [77] to construct a $(n-1, n)$[3.7] threshold ID-PKE scheme. This of course makes the private key more exposure resilient. The solution by Baek *et al.* [11] adds identifier-based threshold decryption to the BF ID-PKE scheme and also provides mediated ID-PKE scheme. This scheme offers stronger security assurances than the ones presented in [102].

- Certificate-based infrastructure: A solution for multi-show digital certificates is proposed by Verheul [148] in which an entity constructs from its original certificate a 'self-blindable' un-linkable certificate with the same attributes as the original certificate.

[3.6] The work in [60] transforms the mediated schemes of [31] into identifier-based schemes. The schemes are constructed using an RSA primitive.

[3.7] A $(n-1, n)$ threshold encryption scheme distributes the private key amongst n entities and requires the components from all n entities to decrypt.

Gentry [76] explores a certificate-based encryption scheme which facilitates an infrastructure for traditional public key cryptography that does not require certificate revocation. A more brief explanation of [76] is given in Section 4.3.3 and a detailed explanation is in Section 7.3.

- Other Infrastracture: A new infrastructure coined 'certificateless public key cryptography' is proposed in [7]. This work is presented and extended in Part II of this thesis.

Encryption schemes: Verheul's main motivation in [147] was to prove that XTR (an efficient method or working with a specific subgroup) is more secure than supersingular elliptic curve cryptosystems. As a by-product of his investigation into this question, Verheul [147, §5.2] described an escrow-able elliptic curve ElGamal encryption scheme. The BF ID-PKE scheme uses the same Weil pairing map as the one presented by Verheul and was covered in Section 3.2.4.1. Ideas from the BF ID-PKE scheme formed the basis of at least two non-identifier based encryption schemes: a certificate-based encryption scheme [76]; and a certificateless encryption scheme [7] (these were already mentioned above). The fully forward secure public key encryption schemes in [43, 94] were built from the hierarchical identifier-based encryption scheme of Gentry and Silverberg [77]. The scheme by Katz [94] and its corresponding security analysis were improved in Canetti, Halevi and Katz in [43].

Key-insulated PKE was first introduced and formalised by Dodis *et al.* [62]. The goal was to minimise the damage caused by private key exposure. The private keys are stored on insecure devices and are refreshed at fixed time intervals via interaction with a physically secure device. The physically secure device stores a master key. The notion of ID-PKE was proved to be equivalent to that of a (not strong) key-insulted PKE by Bellare and Palacio in [19]. Although this idea was discussed briefly in [62], the work of Bellare and Palacio in [19] contains a more concrete discussion that utilises the BF ID-PKE scheme (and hence pairings) to construct a key insulated PKE scheme.

In Dodis *et al.* [61], the definition and concrete realisation of an intrusion-resilient PKE scheme are presented. In the definition intrusion-resilient schemes,

time is divided into periods and the public key remains fixed but the secret key is periodically updated. Secret information is stored by both a user and a base and the function of the base is to periodically update the user's key. The scheme in Dodis *et al.* [61] is based on the forward secure PKE scheme of Katz [94] and extends the key insulation ideas in [62]. This is because intrusion-resilient PKE schemes are secure even if the base and user are compromised, as long as they are not compromised simultaneously. Additionally, previous time periods remain secure even if both user and base are compromised simultaneously. The scheme in Dodis *et al.* [61] is proven secure in the standard model, that is, without random oracles provided that the DBDHP is hard.

Signature schemes: Many interesting signature schemes have been created using pairings on elliptic curves. They are categorised as either identifier-based or certificate-based.

- Identifier-based signature schemes: Sakai, Ohgishi and Kasahara [133] were the first to realise an ID-PKS using pairings. The set up and extraction method presented in this work is very similar to those presented in subsequent ID-PKS schemes [47, 82, 83, 105, 123, 150, 152]. The scheme is not very efficient and does not have a security proof.

  Cha and Cheon [47], Paterson [123] and Hess [83] independently produced ID-PKS schemes at about the same time. Paterson's scheme [123] and its security are closely related to that of the generalised ElGamal signature scheme. The schemes of Cha and Cheon [47] and Hess [83] (unlike the schemes in [123, 133]) are provably secure (in the random oracle model) against existential forgery on adaptively chosen message and identities, provided the CDHP is hard. Security against existential forgery means that the adversary cannot forge a signature on a single message, where the adversary has little or no control over that message [114, §11.2.4]. Hess [83] shows how general exponent group signatures schemes (such as the ElGamal and Schnorr signature schemes) give rise to pairing-based ID-PKS schemes. Hess [83] compares the efficiency of the first of his four schemes to the schemes of Cha and Cheon [47] and Paterson [123]. The

most efficient scheme in [83], scheme 4, should be avoided due to an attack by Cheon [53]. Hess also proposes the use of multiple PKGs (as in [32]) to mitigate the inherent risk of key escrow in the ID-PKS schemes.

Zhang and Kim [150] produced blind and ring ID-PKS schemes. Blind signature schemes are interactive two-party protocols which allow an entity to get a message signed by another entity without revealing any information about the message to the other party. Ring signatures are group signatures without managers which provide anonymity for the signer, since the signature could have been produced by any entity in the ring. The security of the schemes is discussed in [150]. Zhang, Safavi-Naini and Lin [152] constructed proxy ID-PKS, proxy blind ID-PKS and proxy ring ID-PKS schemes from pairings. These schemes allow the proxy signer to sign on behalf of the original signer. Applications of proxy ID-PKS schemes are also described in [152]. Lin and Wu [105] showed how to construct a ring ID-PKS scheme that is computationally more efficient than the ring ID-PKS scheme presented by Zhang and Kim [150].

Han, Yueng and Wang [82] constructed an undeniable ID-PKS scheme and showed that it has the soundness property. Unfortunately, Zhang, Safavi-Naini and Susilo [153] demonstrated two attacks on it. The most serious attack allows the attacker to forge a valid confirmer signature of any ID on an arbitrary message and confirm this signature to the verifier.

Unlike traditional group ID-PKS schemes, Chen, Zhang and Kim [51] produced a group signature scheme where each user concatenates a public key and timestamp in the identifier string presented to the PKG. This is to reduce the level of trust that is needed in the PKG. The set up of this scheme is similar to that of a certificateless signature scheme (except that it also includes time as in Gentry's scheme [76]).

- Certificate-based signature schemes: Boneh, Lynn and Shacham (BLS) [36] showed how to construct short signatures based on the hardness of the CDHP. Signing in the BLS signature scheme requires a single multiplication in $\mathbb{G}_1$ and verification requires two pairing computations. The BLS signature is novel because it offers a similar level of security to that

of a DSA signature and requires roughly only half the bits to represent a signature. The signature is the x coordinate of an element of $\mathbb{G}_1$. This optimisation is possible because the signature uses a point compression technique – recall Section 2.2. The paper provides guidance on selecting elliptic curves so that the DDHP in $\mathbb{G}_1$ is easy, whilst the CDHP in $\mathbb{G}_1$ is hard, that is, $\mathbb{G}_1$ is gap group. The fact that DDHP in $\mathbb{G}_1$ is a easy is crucial for the verification of BLS signatures.

As in the BLS scheme, gap groups were used by Boldyreva [28] to construct efficient threshold signature, multisignature and blind signature schemes. A (t, n) threshold signature distributes the secret key amongst n entities and any subset of more than t is required to construct a signature. Lin, Wu and Zhang [106] used the BLS signature to construct a structured multisignature scheme that forces the verifier to follow a particular order of verification which is predetermined by the group of signers.

Boneh, Gentry, Lynn and Shacham [34] introduce the concept of 'aggregate signatures' which allows for a single short signature to be produced from n signatures on n distinct messages from n distinct users. The scheme is based on the BLS signature scheme and is useful, for example, in reducing the size of certificate chains. However, unlike the BLS scheme, this scheme requires the extra structure provided by the pairing map and does not work with every gap group. Aggregate signatures are shown to give rise to verifiably encrypted signatures in [34]. A verifiably encrypted signature allows the verifier to test whether a ciphertext is the encryption of a signature on a given message. Boneh, Mironov and Shoup [35] produced a PKS scheme which is secure in the standard model against existential forgery under a chosen message attack, provided the CDHP is hard. This scheme is efficient compared to other signature schemes provably secure in the standard model.

Signcryption schemes: Signcryption was first proposed by Zheng in [155]. In [155] Zheng defines signcryption as a cryptographic method that fulfils both the functions of encryption and signature, but with a cost smaller than that required when executing a signature then an encryption algorithm. All the

signcryption schemes except for [38] are more efficient in terms of communication and computation than a direct composition of the ID-PKE scheme [32] and a signature. For example, a way this composition can be performed is called 'encrypt-then-sign'. A direct composition of a encrypt-then-sign identifier-based scheme provides to the encryption scheme two security services; non-repudiation and authentication. Generally, since the schemes are more efficient than this encrypt-then-sign approach, they are considered fairly interesting. The schemes in [38, 103, 108, 109, 119] are identifier-based and add at least integrity and authenticity services to the BF ID-PKE scheme [32]. The security proof of Lynn's scheme [108] extends that of the BF ID-PKE scheme [32] by allowing the adversary to select two identities, the encryptor and decryptor. Neither Lynn's [108] nor Nalla and Reddy's [119] schemes provide non-repudiation. Integrity is achieved by both schemes due to the inability of an attacker to forge ciphertext. For this reason we believe that Nalla and Reddy's signcryption scheme is more accurately labelled as an authenticated encryption scheme. Therefore, it was inaccurate of the authors to compare the efficiency of their scheme in [119] to that of Malone-Lee [109].

Malone-Lee's signcryption scheme [109] uses a variant of an ID-PKS scheme of Hess [83], and reduces the computation of an encrypt-and-sign technique by one point multiplication. Malone-Lee's scheme, however, does not achieve semantic security because the plaintext's signature can be obtained from the ciphertext – for more details see [103]. Libert and Quisquater in [103] provided an identifier-based signcryption scheme that is semantically secure (provided the BDHP is intractable) and publicly verifiable. The scheme's efficiency is comparable to that of Malone-Lee's scheme.

Boyen's signcryption scheme [38] is a sign-then-encrypt scheme. A direct composition of a sign-then-encrypt identifier-based scheme provides three security services: confidentiality, unlinkability and anonymity. Boyen's composition adds non-repudiation and authentication to the scheme's security services. In [38] a formalisation of these properties is presented. Thus, Boyen's signcryption scheme differs from Libert *et al.*'s scheme [103] because it additionally offers unlinkability and anonymity.

The reader will have noticed that pairing-based cryptography has rapidly developed over the last few years. The tremendous rate at which this topic is evolving appears not to be slowing down. Recently there has been some further developments in pairing based schemes, the works presented in [18, 29, 30] are some notable examples. The very rich 'structure' pairings provide makes them a very powerful and flexible tool for cryptographic schemes to be build upon. Hence, this recent surge in constructing cryptographic schemes from pairings in unsurprising and we anticipate that pairings will give birth to further models for the use of PKC. However, this prompts the need for more thorough analysis of the theoretical underpinnings of pairings such as the security of different curves and the security of the BDHP and its related problems.

Part II

Certificateless Public Key Cryptography

Chapter 4

Certificateless Public Key Cryptography

Contents

In this chapter a new paradigm for public key cryptography, called certificateless public key cryptography (CL-PKC), is proposed. Based on the key generation method, all CL-PKC schemes exist in one of two settings. Both settings are analysed and compared to ID-PKC and traditional certificate-based PKC.

4.1 Introduction

A major difficulty in developing secure systems based on public key cryptography is the deployment and management of infrastructures to support the authenticity of cryptographic keys: there is a need to provide an assurance to the user about the relationship between a public key and the identity (or authority) of the holder of the corresponding private key. As we saw in Section 3.2.3, in a traditional PKI this assurance is delivered in the form of a certificate, essentially a signature by a CA on a public key. The problems of PKI technology are well documented, see for example [81]. Of note are the issues associated with certificate management, including revocation, storage, distribution and the computational cost of certificate verification. These are particularly acute in processor or bandwidth-limited environments [56].

As described in Section 3.2.4, ID-PKC tackles the problem of authenticity of keys in a different way to traditional PKI. In ID-PKC an entity's public key is derived directly from certain aspects of its identity, for example, an Internet protocol (IP) address belonging to a network host, or an electronic mail (e-mail) address associated with a user. Private keys are generated for entities by the PKG. In Section 3.2.4.1 we showed an ID-PKC scheme, the BF ID-PKE scheme [32] and we illustrated how it precipitated the rapid development of ID-PKC in our survey in Section 3.6.

On the one hand the direct derivation of public keys in ID-PKC eliminates the need for certificates and some of the problems associated with them. On the other hand the dependence on a PKG, who uses a system-wide master key to generate private keys, inevitably introduces key escrow to ID-PKC systems. For example, the PKG can decrypt any ciphertext in an ID-PKE scheme. Equally, if not more problematic is the notion that the PKG can forge any entity's signatures in an ID-PKS scheme, so ID-PKC cannot offer true non-repudiation in the way that standard PKI can. The escrow problem can be solved to a certain extent by the introduction of multiple PKGs and the use of threshold techniques, but this necessarily involves extra communication and infrastructure. Moreover, the compromise of the PKG's master key could be disastrous in an ID-PKC system, and is usually more severe

than the compromise of a CA's signing key in a traditional PKI. This is because the PKG's master key (in addition to being able to compute new private keys) can be used to compute all the private keys in the system. For example, an adversary equipped with the PKG's master key can read previously encrypted communications and produce valid signatures for any entity without the need to use new identifiers. In a traditional PKI, the adversary needs to issue a new certificate and get entities to accept the new public keys in them, before being able to mount any attack. For these reasons, it seems that the use of ID-PKC may be restricted to small, closed groups or to applications with limited security requirements.

In this chapter, we introduce a new paradigm for public key cryptography, which we name certificateless public key cryptography (CL-PKC). The concept of CL-PKC grew out of a search for public key schemes that do not require the use of certificates, and yet do not have the inherent key escrow feature of ID-PKC. The solution we propose enjoys both of these properties. As we shall see, the properties of CL-PKC are in some sense intermediate between traditional PKI and ID-PKC. We will discuss the model and properties of CL-PKC, as well as provide several examples of CL-PKC applications. CL-PKC can be used to support encryption schemes, signature schemes, key agreement protocols and other public key schemes. A certificateless encryption scheme is denoted a CL-PKE scheme, whilst a certificateless signature scheme is denoted a CL-PKS scheme. We will not be presenting any concrete schemes in this chapter. This chapter, however, puts into context the schemes presented in Chapters 5 through 8.

Next we define CL-PKC and then discuss publications related to CL-PKC in Section 4.3. This is because our concept shares some features with identifier-based cryptography [32, 135], the self-certificated keys of [78, 124] and Gentry's recently proposed certificate-based encryption [76]. In Section 4.4 we outline a general adversarial model for CL-PKC. To understand the functionality of CL-PKC we study the initial enrolment of entities in Section 4.5 and in Section 4.6 we cover properties and applications of CL-PKC. A summary and a comparison table of CL-PKC with traditional PKI and ID-PKC is provided in Section 4.7.

4.2 Defining CL-PKC

Here we sketch the defining characteristics of CL-PKC (more precisely CL-PKC(A)) which will serve to assist the reader in understanding CL-PKC's adversarial model and properties.

A CL-PKC system still makes use of a TA which we name the Key Generating Center (KGC). By way of contrast to the PKG in ID-PKC, this KGC does *not* have access to entities' private keys. Instead, the KGC supplies an entity A with a *partial private key* D_A which the KGC computes from an identifer ID_A for the entity and a master key. As before we will equate A with its identifier ID_A. The process of supplying partial private keys should take place confidentially and authentically: the KGC must ensure that the partial private keys are delivered securely to the correct entities.

The entity A then combines its partial private key D_A with some secret information x_A to generate its actual private key S_A. This way A's private key is not available to the KGC. The entity A also combines its secret information x_A with some public parameters to compute its public key P_A. Note that, in general, A need not be in possession of S_A before generating P_A: all that is needed to generate both is the same secret information. The system is not identifier-based, because the public key is no longer computable from an identifier alone.

Entity A's public key might be made available to other entities by transmitting it along with messages (for example, in a signing application) or by placing it in a public directory (this would be more appropriate for an encryption setting), but no further security is applied to the protection of A's public key. In particular, there is no certificate for A's key. To encrypt a message for A or verify a signature from A, entity B makes use of P_A and ID_A.

We now sketch a simple modification to the defining characteristics of CL-PKC. Instead of the KGC computing the partial private key D_A from only an identifer

ID_A for the entity and a master key, the partial private key D_A is computed from an identifier ID_A, a public key P_A for the entity and a master key. With this alteration on how D_A is computed, the process of supplying partial private keys need not take place confidentially and authentically. We label CL-PKC schemes with this modification in place as CL-PKC(B) schemes.

A more formal model defining certificateless public key encryption (CL-PKE) will be given in Section 5.2. Much of this model is also applicable for our other certificateless primitives. In this chapter, we focus on developing the generic properties of CL-PKC, without reference to specific certificateless cryptographic primitives such as signature or encryption.

4.3 Related Work

4.3.1 Identifier-based Cryptography

Our work on CL-PKC owes much to the pioneering work of Boneh and Franklin [32] on identifier-based public key encryption. Recall that identitifier-based cryptography and the basic scheme of Boneh and Franklin were described in Section 3.2.4. In fact, the CL-PKE schemes in Chapters 5, 6 and 8 are derived from their scheme. Our security proofs require significant changes and new ideas to handle new types of adversary. Likewise, we will show in Chapter 8 that our certificateless signature, key exchange and hierarchical schemes arise by adapting existing ID-PKC schemes.

4.3.2 Self Certified Keys

Another alternative to traditional certificate-based PKI called self-certified keys was introduced by Girault [78]. Girault's schemes combine characteristics of RSA and discrete logarithms. The concept of self-certified keys was further developed by

Petersen and Horster [124] and Saeednia [130]. The schemes presented in [78, 124, 130] are structurally somewhat similar to our CL-PKC schemes. In a self-certified scheme, an entity chooses its private key x and corresponding public key y and delivers y to a TA. The TA combines y with the identity ID of that entity to produce a witness w. This witness may simply be the TA's signature on some combination of y and ID as in [78], part of a signature as in [124], or the result of inverting a trapdoor one-way function based on y and ID [130]. Given w, ID and the TA's public key, any party can extract y, while only the TA can produce the witness w from y and ID. The schemes offer implicit certification, in that the authenticity of a public key is verified implicitly through the subsequent use of the correct private key.

As in CL-PKC, self-certified keys enable the use of public key cryptography without certificates. However, it can be argued that the witness in a self-certified scheme is really just a lightweight certificate linking ID and y. As we shall see, our CL-PKC schemes do not have such witnesses. The self-certified schemes have an advantage over some of our CL-PKC schemes in that the communication between an entity and the TA need not be confidential: there are no partial private keys to be transported to entities. Moreover, the private key in self-certified systems needs to be chosen before the public key can be generated, so the elegant applications of ID-PKC to controlling workflows cannot be realized in self-certified systems. Nor do the self-certified schemes enjoy security proofs. Indeed Saeednia [131] has recently pointed out a basic flaw in the scheme of [78] which allows a cheating TA (who knows the factorisation of system-wide values) to extract an entity's private key; the consequence is that far larger (and less efficient) parameters are needed to create a secure scheme.

Girault stated in [78] that "Self-certified public keys contribute to reduce the amount of storage and computation in public key systems." This primary goal of using self-certified keys was effectively eliminated by Saeednia's observation [131]. Furthermore, there is no other benefit using self-certified keys instead of the traditional certificate-based model Petersen and Horster summarised this in [124] by saying: "Self-certified keys offer *no* structural advantage over certificate-based keys, they offer a concept of equal possibilities for which many useful applications are known."

Self-certified keys should not be confused with the related concept of 'Self Certification' [100]. Self certification allows for the explicit authentication of public keys and proof of possession of private keys. This is done by each entity issuing a certificate for themselves using their own private key. Self certification has been used in conjunction with traditional certificate-based PKI [101] (where a signer generates temporary signing keys) and self-certified keys [100] (where the schemes in [124] are extended by adding certificates) and there is no reason why it could not also be used in conjunction with identity based or certificateless infrastructures. Of course some properties and benefits of these infrastructures are altered due to the addition of these self certificates.

4.3.3 Gentry's Certificate-based Encryption Scheme

Recent work of Gentry [76] exploits pairings to simplify certificate revocation in traditional PKI systems. In Gentry's model, an entity A's private key consists of two components: a first component which that entity chooses for itself and keeps private, and a component which is time-dependent and is issued to A on a regular basis by a CA. Matching the two private key components are two public key components. The first of these is chosen by A while the second can be computed by B using only some public parameters of the scheme's CA together with the current time value and the assumed value of A's public key. Due to the structure of the certificate-based encryption (CBE) scheme, entity B is assured that A can only decrypt if he is in possession of both private components. Thus, the second private component acts as an *implicit certificate* for relying parties: one that a relying party can be assured is only available to A provided that A's certification has been issued for the current time period by the CA. This approach provides an implicit revocation mechanism for PKIs: notice that there is no need for B to make any status checks on A's public key before encrypting a message for A; rather B's assurance that only A can decrypt comes through trusting the CA to properly update and distribute the second components of private keys.

Gentry's schemes [76] are presented in the context of a traditional PKI model, whereas our work in this and the next two chapters departs from the traditional PKI and ID-PKC models to present a new paradigm for the use of public-key cryptography. The CBE scheme's certificate can be verified like a signature as explicit proof of certification. If explicit verification is used, many of the analysis presented in this chapter is inappropriate. However, the two models bear some conceptual resemblance: both make use of keys that are composed of two parts, one chosen by an entity for itself and the other derived from a trusted authority. In fact, it may be possible to modify Gentry's work [76] to divorce it from the setting of a traditional PKI. Conversely, we can modify our scheme to provide CBE functionality by the simple expedient of including a time period (that is, expiry information) and public keys in identifier strings, we will be showing this in Section 7.4.1. The concrete realizations of the two models are different and they are independently developed. Even so, they are closely related. These issues will be further discussed in Section 7.3.

4.4 An Adversarial Model for CL-PKC

Due to the lack of authenticating information for public keys (in the form of a certificate, for example), we must assume that an adversary can replace A's public key by a false key of the adversary's choice. This might seem to give the adversary tremendous power and to be disastrous for CL-PKC. However, we will see that specific schemes can be developed where an active adversary who attacks in this way gains nothing useful: without the correct private key, whose production requires the partial private key and therefore the cooperation of the KGC, an adversary will not be able to decrypt ciphertexts encrypted under the false public key, produce signatures that verify with the false public key, and so on. (Formally, in the encryption setting, the adversary will not be able to decrypt a challenge ciphertext or distinguish the encryptions of distinct messages of his choice.)

Of course, we must assume that the KGC does not mount an attack of this type: armed with the partial private key and the ability to replace public keys, the KGC

could impersonate any entity in generating a private/public key pair and then making the public key available. Thus, we must assume that while the KGC is in possession of the master key and hence all partial private keys, it is trusted not to replace entities' public keys. However, we assume that the KGC might engage in other adversarial activity, eavesdropping on ciphertexts and making decryption queries, for example. In this way, users invest roughly the same level of trust in the KGC as they would in a CA in a traditional PKI. Further explanation on why we use the term *roughly* here is made in Section 4.6.5. A formal model for the capabilities of adversaries and a definition of security for certificateless encryption schemes will be given in Chapters 5 and 6.

4.5 Key Generation Techniques for CL-PKC

Names, e-mail adresses or IP addresses of hosts are often proposed as potential identifiers ID_A. The identifier ID_A could additionally include conditions (in the form of attributes) that A satisfies. For example, the identifier might contain A's age, or sex, or date of birth, or even A's public key P_A. Since we wish to examine the last case in detail, we denote an identifier of A which contains the public key P_A, as $\mathsf{ID}_A \| P_A$. This will eliminate any ambiguity in our analysis and make explicit the benefits of including P_A.

In Figure 4.1, we illustrate the main differences in the registration processes by showing the nature of the communication and communication channel in traditional certificate-based PKC, CL-PKC and ID-PKC. Notice that we have two registration procedures for CL-PKC, both of which will be explored next. First we will outline the benefits of excluding P_A from the identifier. Then we outline an alternative key generation technique where P_A is included as part of the identifier. As we shall see, including P_A in the identifier enhances the resilience of our schemes against a cheating KGC and allows for non-repudation of certificateless signatures.

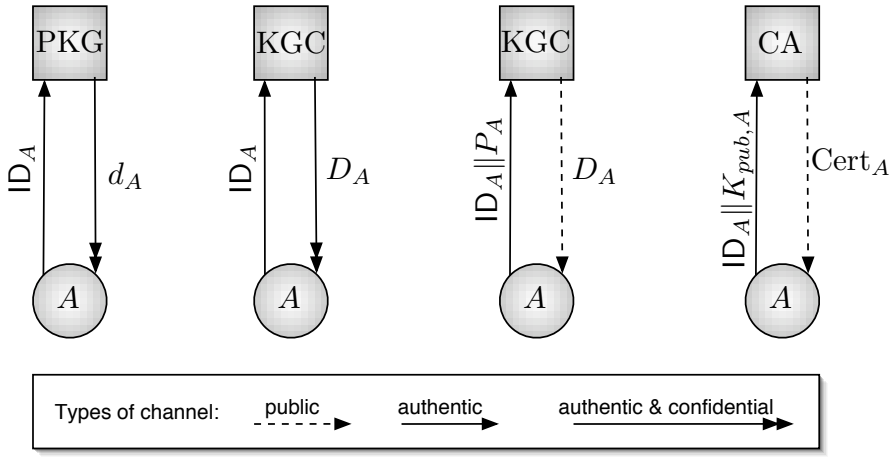

Figure 4.1: Authentication (or witnessing/enrolling) of entity A by the TA for ID-PKC, CL-PKC(A), CL-PKC(B) and traditional certificate-based PKC respectively. Authentications performed by the PKG and KGC, only need to occur before using the private key.

4.5.1 Identifier Context: Excluding P_A

The setting in which public keys are explicitly excluded from the identifiers CL-PKC will be named CL-PKC(A). Here we assume that the KGC is trusted not to replace the public keys of users and will only issue one copy of each partial private key to the correct recipient. This may involve an unacceptable level of trust in the KGC for some users. This setting also allows users to create more than one public key for the same partial private key. This property can be desirable in some applications, but undesirable in others.

An example of a desirable application for CL-PKC(A) is in the construction of simple key renewal schemes with forward security. For each cryptographic use entity A uses a unique public key, $P_{A,j}$, where $j \in \mathbb{N}$. Once $P_{A,j}$ is used, entity A updates it to $P_{A,(j+1)}$. The onetime use of each $P_{A,j}$ ensures that the compromise of one public key (for example, via the exposure of its private key or part of the private key but not the exposure of the partial private key) does not result in the compromise of

any prior public keys. Hence $P_{A,j}$ can be viewed as a short-term key and these schemes offer forward security. In fact, our schemes extends the notion of forward security because the compromise of one public key does not result in the compromise of *any other* public keys. Moreover, this setting can be used to construct schemes which are related to key-insulated PKC [62] and intrusion-resiliant PKC [61] – recall the survey in Section 3.6. The difference, however, is that CL-PKC(A) schemes are not refreshed by distinct time periods and the public key does not remain fixed. Instead CL-PKC(A) operates under a somewhat opposite notion to [61, 62] where the identifier remains fixed and the public key changes. In CL-PKC(A), entity A is able to create a new private key for each 'refreshed' public key without being forced to re-interact with the TA (KGC), hence, CL-PKC(A) can provide a simple non-interactive key renewal mechanism.

Also notice that the benefit of using CL-PKC(A) in this way arises for two types of schemes:

1. Schemes where the public key is easily transported to the party who makes use of it, for example, CL-PKS schemes and key agreement protocols where the public keys are included with the signatures or message passes respectively.
2. Schemes where the KGC goes offline or cannot afford to maintain the computational overhead required to regularly compute new partial private keys. Recall that the functionality of a PKG is performed by a base station in the model of [61] and a secure device in the model of [62].

In CL-PKC(A) a cheating KGC can replace an entity's public key with one for which it knows the secret value without any fear of being implicated. This is because the user with the partial private key could also have been responsible for replacing the public key. Thus, we must assume that no KGC would engage in such an action, and that users trust the KGC not to do so. Note that this action is not equivalent to a CA forging a certificate in a traditional PKI: the existence of two valid certificates would surely implicate the CA (although the CA could perhaps revoke the entity's original certificate first). We will discuss this topic in more detail in Section 4.6.5.

4.5.2 Identifier Context: Including P_A

Here we sketch a simple binding technique which ensures that each user can only create one public key for which he/she knows the corresponding private key; this technique transforms CL-PKC(A) to what we call CL-PKC(B). In our technique an entity A must first fix its secret value, x_A, and its public key, P_A. The identifier is set to $\mathsf{ID}_A \| P_A$. The partial private key D_A which is delivered to entity A is derived from a function with input $\mathsf{ID}_A \| P_A$ and the private key, S_A, is derived from a function with inputs $\mathsf{ID}_A \| P_A$ and x_A. We see that this D_A and S_A are now bound to A's choice of public key. This binding effectively restricts A to using a single public key, P_A, since A can only compute a single private key using D_A.

In general, with this binding in place there is no longer any need to keep partial private keys secret: knowledge of the partial private key D_A does not help an adversary create the unique private key S_A that matches the particular public key P_A which is bound to D_A. We note that this property must actually be proved for any concrete scheme.

This binding technique is particularly important in the context of signatures: it ensures a stronger form of non-repudiation than is otherwise possible for our certificateless signature scheme in Section 8.4. Without the binding an entity can repudiate a signature by producing a second private key and claim that the KGC created the signature using that private key. This is no longer possible with the binding in place: the existence of two private keys for an identity can only result from the existence of two partial private keys binding that identity to two different public keys; only the KGC can create these two partial private keys. Thus, our binding technique can make the KGC's replacement of a public key apparent and equivalent to a CA forging a certificate in a traditional PKI. This binding also reduces the degree of trust that users need to have in the KGC in our certificateless schemes. This is because in CL-PKC(B) a cheating KGC who replaces an entity's public key can be implicated in an event of a dispute. The issue of trust and the issue of non-repudiation will be examined in more detail in Section 4.6.5.

4.6 Properties of CL-PKC

In this section we will discuss the issues of revocation, system complexity and trust in CL-PKC.

4.6.1 Revocation in CL-PKC

There are numerous ways of performing revocation in CL-PKC schemes. Revocation (of keys) in CL-PKC systems can be handled in the same way as in ID-PKC systems. In [32, §1.1.1] the idea of appending validity periods (for example, year, date or time) to identifiers ID_A is given as one convenient solution. In the context of CL-PKC, this ensures that any partial private key, and hence any private key, has a limited shelf-life. Notice that this technique can be used to send messages into the future. This is because all entities in the system can assume that the TA would not issue a relevant partial private key until the appropriate time. So time no acts as trigger for when the KGC is allowed to check the entities identifier and issue a new partial private key. Therefore decryption, which requires the partial private key with the correct time, is only possible after the appropriate time. If the identifier in the partial private key contains both the public key and validity periods, then the partial private key becomes a Gentry-like implicit certificate, that is, a type of short lived certificate system which can have future validity periods.

Alternatively revocation can be performed by revoking the identifier, a component of the identifier (such as an address attribute), or the public key of a particular entity using standard certificate-based revocation techniques. For example, this can be done by deploying a analogue of the online certificate status protocol (OCSP) or variants of certificate revocation lists (CRLs). Note that the standard revocation techniques when used in the CL-PKC setting may require less bandwidth than certificate-based counterparts. This because only public keys and/or identifiers need to be revoked and not certificates, which are typically larger. This dual way of tackling the revocation problem in CL-PKC allows for a 'best of both worlds' solution to be deployed.

Finally, by exploring a method of updating private keys (that is, implicit revocation of private keys) in ID-PKC, we can create an alternative CL-PKC revocation method. This method reduces the exposure time of the master-key in ID-PKC and is suitable for users of ID-PKC who require stronger than usual security. The method is straightforward: it requires regularly updating the PKC's public key sP in params (by using a new s for every month, for example). In ID-PKC each entity will have to re-establish an authentic and confidential channel with the PKC to obtain a private key for the updated s – recall Figure 4.1. This interaction is clearly expensive, therefore, such a solution is impractical. CL-PKC(B), however, only requires an authentic channel with the KGC. This authentic channel can be reduced to a public channel if the public key of the entity remains unchanged. This is because when the entity's public key remains the same, the KGC knows that only that entity owns the public key's matching private key so re-authentication is not required. Hence, this revocation solution becomes practical because of its low-interaction. Actually this is a solution for releasing a partial private key only at the right time and is related to the intrusion-resilient public key [61], key-insulated public key [62] and certificate-based [76] solutions. See Section 4.6.4 for more examples extending this low-interactiveness property of CL-PKC.

4.6.2 Certificate Free

It bears repeating that our CL-PKC schemes are certificate-free. CL-PKC eliminates many of the problems associated with traditional certificate-based PKC. For example:

- CL-PKC storage and communication bandwidth is low because the identifier only contains relevant information; certificate-related redundancies are not present.
- CL-PKC potentially reduces the computational bandwidth, as certificates do not need to be verified before the public keys are used.
- CL-PKC offers its users a higher degree of privacy due to its inclusion of only

relevant information in the identifier. Certificates can contain a lot of potentially irrelevant information (based on application), and with the increase of identity theft, mitigating the risk of putting personal information (such as an address or a date of birth) into the wrong hands is highly desirable.

Furthermore, due to the distributive nature of certificates, another inherent property of certificate-based PKI is its static centralised point of control and static certificate content. This can be restrictive and is seen to be the root of many business and legal related problems. For example, entity B cannot initiate any secure communication with entity A unless A owns a certificate in advance. Furthermore, entity B may reject using the public key in the valid certificate because it either originated from an untrusted CA (to B) or because it did not contain the precise information required by B. Hence, unless entity A knows for sure its certificate will be accepted by B, it will not get a certificate in advance. Thus, entity A has no incentive to pay for a certificate except in two scenarios: (i) the certificate is tailored for a very specific pre-determined application, or (ii) the certificate content is very broad in an attempt to capture all applications. From the CA's point of view, the first scenario is ideal for business, however, it requires some prior communication with B to determine what B requires A's certificate to contain. If A wishes to communicate securely with multiple Bs (who each have different certificate requirements), then A is required to re-authenticate himself with the CA if each certificate contains a different public key, we will return to this issue in Section 4.6.4. The second scenario requires the publication of a single certificate which is neither profitable for the CA nor desirable for all Bs. These problems can be mitigated by using CL-PKC.

In the CL-PKC setting, entities using the system can easily specify the content of the identifier (hence, they apply logic into the system) and so they play a more prominent role in the system. This is related to concept of cryptographic workflow which was previously discussed in Section 3.2.5. This transforms the role of TAs from policy pre-distributors (often 'blanket' policies) to that of policy enforcers. In CL-PKC, the public key P_A can either be for a specific TA (KGC) chosen by the public key owner, A, as in standard certificate-based PKC, or any TA (KGC) chosen

by the entity communicating using the public key, B – provided all KGCs share some public parameters. Both these settings are examined next.

- Dynamic point of control and identifier content: An encryptor applying some logic (by adding some conditions, for example) during encryption can pick the TA and dictate the policy under which the ciphertext he encrypted can be decrypted. For example, the schemes in Chapter 6 provide this property.
- Static point of control and dynamic identifier content: A decryptor can use a public key that is restricted to a specific TA and hence dictate the TA with which he is willing to deal. The encryptor only applies some logic during encryption which dictates the policy under which the ciphertext he encrypted can be decrypted. For example, the schemes in Chapter 8 provide this property.

4.6.3 Flexibility via Cryptographic Workflow

It was illustrated in [48, 122, 143] how ID-PKC can be used as a tool to enforce cryptographic workflows, a concept which we covered in Section 3.2.5. CL-PKC supports cryptographic workflow in the same way as ID-PKC. Furthermore, a very similar workflow procedure to that explored in Section 3.2.5 can be constructed using CL-PKC. To understand the similarities, let us consider the example presented in Section 3.2.5.1. The problem in this example remains the same, which is that B needs a simple solution for distributing a serial-number to a customer A in the absence of a credit card infrastructure. The solution to this problem, however, needs to be adapted in a minor fashion to take into account A's public key which is present in the CL-PKC setting. The only difference in the solution is that B additionally includes A's public key in the encryption stage and that a partial private key D_A is obtained from the KGC – instead of d_A from the PKG. Now D_A is used to compute S_A, which in turn is used to decrypt the ciphertext containing the serial-number. Notice that everything else including the identifier, '$\mathsf{ID}_A \| \texttt{paid}\ B\ \$\texttt{X}$', remain unchanged.

Unlike in ID-PKC, in the above example the TA cannot escrow all communications.

Moreover, the consequences of a master key compromise are far less disastrous for both the TA and B. After all, the livelihood of B may rest on the fact that its serial-numbers remain confidential. In the case where a certificate-based PKI is used, no elegant solution exists unless we use a specialised payment infrastructure such as credit card infrastructure. In the absence of any additional payment infrastructure, one solution is for A to obtain all the conditions that B will require in advance, then apply for a one time certificate containing all these suitable conditions. The certificate Cert_A has to be obtained first, since only after its successful verification will B encrypt the serial-number with the public key $K_{pub,A}$. Notice that extra rounds of communication are required. Alternative solutions involving secret sharing between B and the CA are also not as elegant as the one using CL-PKC demonstrated above.

4.6.4 Low Interaction

We have already described in Section 4.5.1 how CL-PKC(A) can be used to provide a non-interactive key renewal mechanism. Nevertheless, CL-PKC is generally considered more interactive when compared with ID-PKC: after all some ID-PKC schemes are non-interactive. However, in the next example we show a surprising result, which is that a certificateless public key signature (CL-PKS) scheme can require less interaction when compared with either a standard certificate-based PKS scheme or an ID-PKS scheme. This benefit always holds for situations where the signing entity reuses the same public key with different identifiers. Moreover, the CL-PKS scheme has better non-repudiation properties when compared to ID-PKS schemes.

4.6.4.1 An Example Use of a CL-PKS Scheme

In the scenario below we exploit the low interactiveness of CL-PKS schemes.

Problem: Entity A (for example, a manager) needs to sign many documents. Each document, however, needs to be signed using a different policy. Policies frequently

change and some policies could be conflicting. The policy could include fields such as role of the signer, date of policy, liabilities and penalties etc.

Solution: Using a CL-PKS scheme, set the identifier of A to be '$\mathsf{ID}_A \| \text{policy}_i \| P_A$'. Entity A authenticates himself to the KGC *once* with the identifier '$\mathsf{ID}_A \| \text{policy}_1 \| P_A$' using the communication channels illustrated for CL-PKC in Figure 4.1 to obtain $D_{A,i}$ for $i = 1$. All subsequent partial private keys with different policies required during signing can be obtained from the KGC by simply using public channels as long as the same P_A is included in the identifier. This is because only ID_A owns P_A's matching x_A. The KGC can check whether entity A with identity string ID_A satisfies (or continues to satisfy) each new policy before issuing a new $D_{A,i}$ for $i \geq 2$.

Analysis: The signature and verification procedure is just like that used in ID-PKS schemes. The reason this CL-PKS scheme is considered less interactive than an ID-PKS is because no additional authentication is required when requesting new partial private keys. Moreover, private channels are not required to distribute the new partial private keys. An additional property of CL-PKS schemes is that they provide non-repudiation, unlike ID-PKS schemes.

In context of traditional PKI, if we allow a CA to produces two certificates $\text{Cert}_{A,1}$ and $\text{Cert}_{A,2}$ with different policies (for example, policy_1 and policy_2) for a single public key $K_{pub,A}$, then potential problems are created. This is because the two policies in the certificates can be contradictory and during disputes the verifier (or signer) can claim that $K_{pub,A}$ (or $K_{priv,A}$) was used under the certificate $\text{Cert}_{A,2}$ and not $\text{Cert}_{A,1}$. This problem arises because the binding between the policy and public key in standard certificate-based cryptography is not as explicit as in CL-PKC. Of course if different values of $K_{pub,A}$ are used for each policy, then a new $K_{pub,A,i}$ needs to be computed each time by A. However, for each $K_{pub,A,i}$, entity A needs to re-authenticate himself to the CA, and provide a proof of possession of the private key matching $K_{pub,A,i}$.

4.6.5 Trust, Non-repudiation and Cryptographic Evidence

Trust is a fundamental resource which needs to be explicitly defined in any new cryptographic system. It is particularity important to understand the trust relationships between entities and their TAs in public key systems. Girault [78] presented a simple formalisation of trust in public key systems making use of a TA by defining three levels of trust, which are:

- Trust Level 1: The TA knows (or can easily compute) entities' private keys. The TA can impersonate any entity at any time without being detected.
- Trust Level 2: The TA does not know (or cannot easily compute) entities' private keys. However, the TA can still impersonate an entity by generating false guarantees using false authentication.
- Trust Level 3: The TA does not know (or cannot easily compute) entities' private keys. The TA can still impersonate any entity, however, the fraud of the TA can be detected.

A CA in traditional certificate-based PKI is often assumed not to issue new certificates binding arbitrary public keys and entity combinations of its choice. This is so the CA does not bind public keys where it knows the corresponding private key. In a traditional PKI, if the CA forges certificates, then the CA can be identified as having misbehaved through the existence of two valid certificates for the same identity. Hence, traditional PKIs achieve Trust Level 3. By contrast, ID-PKC only achieves Trust Level 1 because the PKG knows every entity's private key. It is instructive to examine the Trust Level that is achieved by CL-PKC. When compared to ID-PKC the trust assumptions made of the TA (that is, KGC) in CL-PKC are much reduced: in ID-PKC users must trust the PKG not to abuse its knowledge of private keys in performing passive attacks, while in CL-PKC users need only trust the KGC not to *actively* propagate false public keys. In our CL-PKC(A) schemes a new public key could have been created by the legitimate user or by the KGC, and it cannot

be easily decided which is the case. This means that our CL-PKC(A) schemes only achieve Trust Level 2. Notice that using a self certificate (recall Section 4.3.2) does not improve the fundamental trust relationship (that is, Trust Level) between each entity and the TA in CL-PKC(A). This is because the KGC can still impersonate any entity by generating false self certificates. Furthermore, self certificates generated by the entities are indistinguishable from the self certificates generated by the KGC.

As we have seen in Section 4.5.2, we can further strengthen security against a malicious KGC in our schemes by allowing entities to choose identifiers, which bind together their public keys and identities. Now the existence of two different working public keys for the same identity will identify the KGC as having misbehaved. By a 'working' public key we mean that the private key operation matching the public key has been executed. The existence of two working public keys for an identity can only result from the existence of two partial private keys binding that identity to two different public keys; only the KGC can create these two partial private keys. With this binding in place, CL-PKC(B) reaches a higher trust level than CL-PKC(A). The Trust Level attained by CL-PKC(B) is between Trust Level 2 and Trust Level 3. We explain below why CL-PKC(B) does not fully attain Trust Level 3, but first we take a closer look at encryption schemes in a traditional PKI.

A dishonest CA in standard PKC can be detected trying to impersonate A if it issues a new certificate binding its public key to A's identifier string. This new certificate contains the CA's chosen public key, K_{pub,CA_A}, and will have the form:

$$\mathrm{Cert}_{\mathrm{CA}_A} = (\mathsf{ID}_A \| K_{pub,\mathrm{CA}_A} \| \Sigma(\mathsf{ID}_A \| K_{pub,\mathrm{CA}_A}), K_{priv,\mathrm{CA}}).$$

Entity B encrypts for entity A using the (false) public key K_{pub,CA_A} from $\mathrm{Cert}_{\mathrm{CA}_A}$; the CA can decrypt the ciphtertext with K_{priv,CA_A} and then re-encrypt using A's original public key, $K_{priv,A}$, from Cert_A. Note here that the private key of the CA, $K_{priv,\mathrm{CA}}$, which is used for signing certificates is not the same as the private key used in the impersonation, that is, K_{priv,CA_A}. Entities A and B can only see that an attack has taken place if they later compare the certificate that B verified before encrypting to A with the certificate A owns. The attack is detectable only if A and

B suspect it has taken place. The evidence is the CA's signature on the false public key. Since certificates are intended to be public and readily available, this evidence is easily gathered by A or B.

Now let us examine the same set of issues for CL-PKC(B). A misbehaving KGC in CL-PKC(B) can be detected trying to impersonate A if it issues for itself a new partial private key, binding its chosen public key P_{CA_A} to A's identifier string. This new partial private key will be produced by a key generation function with input $\mathsf{ID}_A \| P_{\mathrm{CA}_A}$, instead of input $\mathsf{ID}_A \| P_A$. Entity B encrypts for A using the public key P_{CA_A}; the CA can decrypt and then re-encrypt using A's original public key, P_A. Entities A and B can only see that an attack has taken place if they later compare the public key that B used in encrypting to A with the public key A has. The attack is detectable only if A and B suspect it has taken place. If the partial private key is public, the evidence implicating the KGC is the false partial private key. However, unlike the situation with a traditional PKI, we cannot assume that the partial private key is always accessible. It may well be available in a CL-PKC(B) scheme, but there is no guarantee of this for A or B.

When the partial private key is not public, then evidence implicating the KGC is a single message encrypted with two different working public keys. One cannot simply implicate the KGC when the same message is found to be encrypted with two different public keys, by claiming that entity A has one working public key, and the KGC has the other. This is because the evidence can be disputed by the KGC, as it cannot be always assumed that entity B is an honest participant. A dishonest participant B could encrypt the same message once with P_A and send it to A and then encrypt the message with P'_A. If we assume that B is honest, then when A and B meet, B can claim that it only encrypted the message using P'_A. The KGC is then implicated because it is the only entity that could have translated the ciphertext which B encrypted using P'_A to one which uses the correct public key, P_A, by decrypting and re-encrypting. Hence, the binding does not make the Trust Level of the CL-PKC(B) encryption scheme identical to that of certificate-based PKC: rather, it rests slightly below Trust Level 3, and the exact level depends on the availability of partial private

keys and the honesty of participants. This motivates us to redefine Trust Level 3 as follows: "The TA does not know (or cannot easily compute) entities' private keys. The TA can still impersonate any entity, however, the fraud of the TA can *always* be detected." Thus, on their own, CL-PKC(B) encryption schemes do not achieve Level 3, while certificate-based PKC schemes do.

Now we consider the Trust Level for primitives other than encryption. Any communication which offers a proof of possession (PoP) of the private key corresponding to the public key, such as a signature or communication using keys agreed in an authenticated key agreement protocol, will provide evidence of a working public key which can be used to implicate the KGC. Consequently, any CL-PKC(B) scheme in which the cryptographic primitive is accompanied by a PoP of the private key will automatically achieve Trust Level 3, since the entities are always able to implicate the KGC.

The levels of trust defined in this section are related to non-repudiation (see p.37 for a definition). CL-PKC(B) schemes which achieve Trust Level 3, such as a signature scheme, automatically provide non-repudiation. This is because non-repudiation, which in essence is the inability of an entity to deny having used the private key (known only to himself), can only be attained with Trust Level 3 schemes.

Another important link between Trust Level 3 and non-repudiation arises because an entity will have to convince a court (or a legal system) of the TA's wrong doing and the court's decision will be based upon the evidence. This legal process is expensive, and so it is only practical for som cases. Furthermore, in cases where secrets are lost (often associated with encryption and key agreement), the legal process is insufficient, since compensation is usually irrelevant. If the legal process will not be used (or does not exist), the main advantage of deploying a Trust Level 3 encryption scheme instead of Trust Level 2 encryption scheme diminishes.

4.6.6 Interoperability of CL-PKC Implementation

As will become apparent, the CL-PKC schemes in this thesis are very closely related to existing pairing-based ID-PKC schemes. One consequence of this is that any infrastructure deployed to support pairing-based ID-PKC, for example, a PKG can also be used to support our CL-PKC schemes: in short, the two types of schemes can easily co-exist. In fact, an entity can be granted a private key for a pairing-based ID-PKC scheme and immediately convert it into a private key for our CL-PKC scheme. In this way, an entity who wishes to prevent the PKG from exploiting the escrow property of an identifier-based system can do so, although, at the cost of losing the identifier-based nature of its public key.

4.6.7 Efficiency

All the schemes we present in Chapters 5, 6 and 8 use a small number of pairing calculations for each cryptographic operation, some of which can usually be eliminated when repeated operations involving the same entities (identifiers) take place. Public keys are usually small in size because they are elements of $\mathbb{G}_1$, which is usually a subgroup of the group of points on an elliptic curve of moderate size.

The infrastructure needed to support CL-PKC is lightweight when compared to a traditional PKI. This is because, just as with ID-PKC, the need to manage certificates is completely eliminated. This immediately makes CL-PKC attractive for low-bandwidth, low-power situations, for example, mobile security applications, where the need to transmit and check certificates has been identified as a significant limitation [56]. Note, however, that the signature schemes enjoying very short signatures [36] could also be used to significantly decrease the size of certificates and create a lightweight PKI.

4.7 Summary of CL-PKC

In this chapter we introduced the concept of *certificateless public key cryptography*, a model for the use of public key cryptography that complements (and is intermediate between) the identity-based approach and traditional PKI.

	ID-PKC	CL-PKC(A)	CL-PKC(B)	Trad. PKC
Certificate-free	Yes	Yes	Yes	No
Cryptographic Workflow	Yes	Yes	Yes	No
Level of Trust	1	2	$2/3^{(i)}$	3
Non-repudiation of Sig.	No	No$^{(ii)}$	Yes	Yes
Non-interactive$^{(iii)}$	Yes	No$^{(iv)}$	No	No

Table 4.1: Comparison of the properties of traditional PKC, CL-PKC and ID-PKC.

(i) Trust Level 3 is achieved provided *any* of the following three conditions are true: the scheme provides a PoP of the private key; the partial private key is public; or the entity communicating by using the public key is assumed to be honest.

(ii) To create a signature, the KGC in CL-PKC(A) also needs to replace the public key of the entity before signing a message. In ID-PKC, the PKG produces a signature by using the entity's private key. Since the PKG in ID-PKC does not need to alter any value in the system, the two notions of non-repudiation for CL-PKC(A) and ID-PKC are not identical.

(iii) Note that non-interactive schemes, although desirable, cannot produce forward secure schemes. Non-interactivity is only relevant for key agreement and encryption schemes.

(iv) Not non-interactive in the usual sense, however, it produces a mechanism for non-interactive key renewal – see Section 4.5.1.

A summary of CL-PKC's core properties can be found in Table 4.1 and the nature of CL-PKC's witnessing channel was illustrated in Figure 4.1. Moreover, we explained in Sections 4.6.1 and 4.6.4 why entity A does not always need an authentic channel when communicating with the KGC in CL-PKC(B) in Figure 4.1. For example, we showed in Section 4.6.4 how repeated interactions with the KGC using the same public key can eliminate the need for re-authentication. CL-PKC has other distinct properties, which were demonstrated by the examples in Sections 4.5 and 4.6.3. The use and applications of identifiers in CL-PKC allows for a granular approach to many problems. The identifiers we used are very simple and it can be adapted to benefit

many different real world operations and processes.

To summarise, Shamir in [135, p. 47] stated when discussing the idea of ID-PKC :

> The scheme is ideal for closed groups of users such as executive of multinational company or the branches of a large bank, since the headquarters of the corporation can server as a key generation center that everyone trusts. ...

Many of the ideas and twists for ID-PKC carry forward to CL-PKC, however, the implicit restrictions of ID-PKC captured by using the word 'closed' does not hold in CL-PKC and this is to do with 'trust'. The underlying trust model of CL-PKC was examined and analysed in this chapter. As a result, the converse word 'open' is more appropriately describes CL-PKC because, for example, properties such as true non-repudiation is seldom required in closed groups.

Now that we have outlined the general properties of CL-PKC, we briefly consider examples of specific CL-PKE schemes. In Chapter 5 we will consider a simple CL-PKE scheme which is secure in a weak model. This CL-PKE scheme is built upon in Chapter 6, where we will provide a CL-PKE scheme which is secure in a stronger model. Furthermore in Chapter 6, we demonstrate some generic CL-PKE schemes that make use of any ID-PKE scheme and any standard PKE scheme. To round off the subject of CL-PKC, in Chapter 8 we derive some further examples of certificateless schemes. These schemes all share a common setting. A scheme's satisfactory deployment is related to its functionality and overall performance. If public keys can used naturally (in efficient schemes) in conjunction with conditions and workflows, then the ideas discussed in this chapter can achieve their full potential.

Chapter 5

CL-PKE – OWE Security

Contents

In this chapter we define the concept of certificateless public key encryption (CL-PKE). To illustrate CL-PKE, we focus on a simple construction that is secure in a weak model. This chapter will serve as a building block for the sequel, in which we will construct a CL-PKE scheme that is secure in a much more robust model.

5.1 Introduction

The only way to make concrete the concepts in Chapter 4 is by presenting actual CL-PKC schemes. Proposing a simple and yet practical certificateless public key en-

cryption (CL-PKE) scheme, the subject of this chapter, is an important advancement of CL-PKC.

One of the main contributions of this chapter is the precise definition of the concept of CL-PKE in Section 5.2. Another important contribution is a set of security results which will be reutilised in Chapter 6. This will make the next chapter easier to follow and will familiarise the reader with many provable security techniques. Furthermore, the simple CL-PKE scheme of this chapter will help the reader understand the concept of CL-PKC and our other CL-PKE schemes.

We will study an adversarial model for CL-PKE in Section 5.3. The adversarial model is a one-way encryption model (OWE). It captures an adversary who can replace public keys and another who has access to the master key (but does not replace public keys). We then consider a simple and computationally efficient CL-PKE scheme in Section 5.4. We prove that it is secure in our OWE security model, provided that the BDHP is hard, in Section 5.5. Note that this scheme was not presented in [6]. The OWE model is weak, and may not be appropriate for all applications. Ultimately, we will prove the security of a related CL-PKE scheme in a stronger model in the next chapter.

5.2 Certificateless Public Key Encryption

In this section we present a formal definition for certificateless public key encryption (CL-PKE). We also examine the capabilities which may be possessed by adversaries against a CL-PKE scheme and from this, derive a security model for CL-PKE. This security model used in this chapter will serve as a building block for the stronger model presented in Chapter 5.

A CL-PKE scheme is specified by seven randomized algorithms: Setup, Partial-Private-Key-Extract, Set-Secret-Value, Set-Private-Key, Set-Public-Key, Encrypt and Decrypt. We describe the function of each of these algorithms in turn.

Setup: This algorithm takes security parameter k as input and returns the system parameters params and master-key. The system parameters includes a description of the message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$. Usually, this algorithm is run by the KGC. We assume throughout that params are publicly and authentically available, but that only the KGC knows master-key.

Partial-Private-Key-Extract: This algorithm takes params, master-key and an identifier for entity A, $\mathsf{ID}_A \in \{0,1\}^*$, as input. It returns a partial private key D_A. Usually this algorithm is run by the KGC and its output is transported to entity A over a confidential and authentic channel.

Set-Secret-Value: This algorithm takes as inputs params and an entity A's identifier ID_A as inputs and outputs A's secret value x_A.

Set-Private-Key: This algorithm takes params, an entity A's partial private key D_A and A's secret value x_A as input. The value x_A is used to transform D_A into the (full) private key S_A. The algorithm returns S_A.

Set-Public-Key: This algorithm takes params and entity A's secret value x_A as input and from these constructs the public key P_A for entity A.

Normally both Set-Private-Key and Set-Public-Key are run by an entity A for itself, after running Set-Secret-Value. The same secret value x_A is used in each. Separating them makes it clear that there is no need for a temporal ordering on the generation of public and private keys in CL-PKE. Usually, A is the only entity in possession of S_A and x_A, and x_A will be chosen at random from a suitable and large set.

Encrypt ($\mathcal{E}^{\mathrm{CL}}$): This algorithm takes as inputs params, a message $M \in \mathcal{M}$, and the public key P_A and identifier ID_A of an entity A. It returns either a ciphertext $C \in \mathcal{C}$ or the null symbol $\perp$ indicating an encryption failure. Such a failure will always occur in the event that P_A does not have the correct form[5.1]. In our scheme, this is

[5.1] The encryption schemes in [6] and Chapter 8 could fail for reasons to do with the structure of the public key.

the only way an encryption failure will occur. We write $\{C, \perp\} \leftarrow \mathcal{E}^{\mathrm{CL}}(M, P_A, \mathsf{ID}_A)$.

Decrypt ($\mathcal{D}^{\mathrm{CL}}$): This algorithm takes as inputs params, $C \in \mathcal{C}$, and a private key S_A. It returns a message $M \in \mathcal{M}$ or a message $\perp$ indicating a decryption failure. We write $\{M, \perp\} \leftarrow \mathcal{D}^{\mathrm{CL}}(C, S_A)$.

Naturally, we insist that output M should result from applying algorithm Decrypt with inputs params, S_A on a ciphertext C generated by using algorithm Encrypt with inputs params, P_A, ID_A on message M. In other words:

$$\mathcal{D}^{\mathrm{CL}}(\mathcal{E}^{\mathrm{CL}}(M, P_A, \mathsf{ID}_A), S_A) = M.$$

5.3 OWE Security Model for CL-PKE

Given this formal definition of a CL-PKE scheme, we are now in a position to define one-way encryption (OWE) adversaries for such a scheme. A formal description of OWE security for standard PKE schemes can be found in Section 3.5.3; our definition of OWE security for CL-PKE builds on this definition. The usual models for security of PKE were strengthened for ID-PKC in [32] to handle adversaries who can extract the private keys of arbitrary entities and who choose the identifier $\mathsf{ID}_{\mathrm{ch}}$ of the entity on whose public key they are challenged. This extension is appropriate because the compromise of some entities' private keys should not affect the security of an uncompromised entity's encryptions. It also shows that the scheme is secure in the presence of colluding entities.

Here, we further extend the model of [32] to allow adversaries who can extract partial private keys, private keys, or both, for identities of their choice. Given that our scheme has no certificates, we must further strengthen the model to allow for adversaries who can replace the public key of any entity with a value of their choice. We must also consider carefully how a challenger should respond to key extraction for identities whose public keys have been changed.

In the next chapter, we will consider an even stronger adversary who can also decrypt arbitrary ciphertexts of his choice.

Here we provide a list of the actions that a general adversary against a CL-PKE scheme may carry out and a discussion of how each action should be handled by the challenger for that adversary.

1. **Extract partial private key of A:** Challenger $\mathcal{C}$ responds by running algorithm Partial-Private-Key-Extract to generate the partial private key D_A for entity A.

2. **Extract private key for A:** As in [32], we allow our adversary $\mathcal{A}$ to make requests for entities' private keys. If A's public key has not been replaced then $\mathcal{C}$ can respond by running algorithm Set-Private-Key to generate the private key S_A for entity A (first running Set-Secret-Value for A if necessary). Also as in [32], we insist that $\mathcal{A}$ does not at any point extract the private key for the selected challenge identifier $\mathsf{ID}_{\mathrm{ch}}$. In [6], we argued that it was unreasonable to expect $\mathcal{C}$ to be able to respond to an extract private key query if $\mathcal{A}$ has already replaced A's public key. We always disallowed this in the model essentially because it would be impossible to simulate. Even if it were possible to simulate this for the schemes in [6], it would not (immediately) lead to an attack on those schemes. However, in the scheme of this chapter (and Chapter 6) it does lead to an attack. Therefore, the grounds on which we adopt this restriction in our model has altered from [6] – this is a subtle but important point. Let us consider heuristically why an attack could exist for the scheme in this chapter. The private key consists of two *separate* components: a partial private key (corresponding to a particular identity) and a secret value (corresponding to a particular public key). Here, if and adversary $\mathcal{A}$ is, for example, allowed to replace the public key for any identifier ID with the challenge public key, P_{ch}, then $\mathcal{A}$ can derive from the private key for identifier ID the secret value x_{ch} corresponding to P_{ch} by making an extract private key request on ID. Then if $\mathcal{A}$ makes a partial private key request on $\mathsf{ID}_{\mathrm{ch}}$, $\mathcal{A}$ can construct the private key of $\mathsf{ID}_{\mathrm{ch}}$ from the partial private key and the secret value x_{ch}.

3. **Request public key of A:** Naturally, we assume that public keys are available to $\mathcal{A}$. On receiving a first request for A's public key, $\mathcal{C}$ responds by running algorithm Set-Public-Key to generate the public key P_A for entity A (first running Set-Secret-Value for A if necessary).

4. **Replace public key of A:** Adversary $\mathcal{A}$ can repeatedly replace the public key P_A for any entity A with any value P'_A of its choice. We assume here that the adversary's choice P'_A is a valid public key; this assumption can be removed (and our schemes remain secure) at the cost of some additional complexity in our definitions. The current value of an entity's public key is used by $\mathcal{C}$ in any computations (for example, preparing a challenge ciphertext) or responses to $\mathcal{A}$'s requests (for example, replying to a request for the public key). We insist that $\mathcal{A}$ cannot both replace the public key for the challenge identifier ID_{ch} before the challenge phase *and* extract the partial private key for ID_{ch} in some phase – this would enable $\mathcal{A}$ to receive a challenge ciphertext under a public key for which it could compute the private key.

We also want to consider adversaries who are equipped with master-key, in order to model security against an eavesdropping KGC. As discussed in Section 4.1, we do not allow such an adversary to replace public keys: in this respect, we invest in the KGC a similar level of trust as we do in a CA in a traditional PKI – recall Section 4.6.5. So we will distinguish between two adversary types, with slightly different capabilities:

CL-PKE Type I OWE Adversary: Such an adversary $\mathcal{A}_I$ does not have access to master-key. However, $\mathcal{A}_I$ may request public keys and replace public keys with values of its choice and extract partial private and private keys for all for identities of its choice. As discussed above, we make several natural restrictions on such a Type I adversary:

1. Adversary $\mathcal{A}_I$ cannot extract the private key for ID_{ch} at any point.

2. Adversary $\mathcal{A}_I$ cannot request the private key for any identifier if the corres-

ponding public key has already been replaced.

3. Adversary $\mathcal{A}_I$ cannot both replace the public key for the challenge identifier ID_{ch} before the challenge phase *and* extract the partial private key for ID_{ch} in some phase.

CL-PKE Type II OWE Adversary: Such an adversary $\mathcal{A}_{II}$ does have access to master-key, but may not replace public keys of entities. Adversary $\mathcal{A}_{II}$ can compute partial private keys for itself, given master-key. It can also request public keys and make private key extraction queries for identities of its choice. The restrictions on this type of adversary are:

1. Adversary $\mathcal{A}_{II}$ cannot replace public keys at any point.
2. Adversary $\mathcal{A}_{II}$ cannot extract the private key for ID_{ch} at any point.

One-way encryption security for CL-PKE: We say that a CL-PKE scheme is OWE secure if no polynomially bounded adversary $\mathcal{A}$ of Type I or Type II has a non-negligible advantage against the challenger in the following game:

Setup: The challenger takes a security parameter k as input and runs the Setup algorithm. It gives $\mathcal{A}$ the resulting system parameters params. If $\mathcal{A}$ is of Type I, then the challenger keeps master-key to itself, otherwise, it gives master-key to $\mathcal{A}$.

Phase 1: Adversary $\mathcal{A}$ issues a sequence of requests, each request being either a partial private key extraction, a private key extraction, a request for a public key or a replace public key command for a particular entity. These queries may be asked adaptively, but are subject to the previously defined rules on adversary behaviour.

Challenge Phase: Once $\mathcal{A}$ decides that Phase 1 is over it outputs an identifier ID_{ch} on which it wishes to be challenged. Again, the adversarial constraints given above apply. In particular, ID_{ch} cannot be an identifier for which the private key has been

extracted. Moreover, if $\mathcal{A}$ is of Type I, then ID_{ch} cannot be an identifier for which both the public key has been replaced and the partial private key extracted. The challenger now picks a random plaintext $M \in \mathcal{M}$ and computes C^*, the encryption of M under the current public key P_{ch} for ID_{ch}. Then C^* is delivered to $\mathcal{A}$.

Phase 2: Now $\mathcal{A}$ issues a second sequence of requests as in Phase 1, again subject to the rules on adversary behaviour above. In particular, no private key extraction on ID_{ch} is allowed, and, if $\mathcal{A}$ is of Type I, then the partial private key for ID_{ch} cannot be extracted if the corresponding public key was replaced in Phase 1.

Guess: Finally, $\mathcal{A}$ outputs a guess $M' \in \mathcal{M}$. The adversary wins the game if $M = M'$. We define $\mathcal{A}$'s advantage in this game to be $\text{Adv}(\mathcal{A}) := \Pr[M = M']$.

Notice that the definition of OWE for standard PKE is similar to that for CL-PKE. In CL-PKE we additionally have extraction queries, request/replace public key queries and the adversary is challenged on an identifier/public key pair of its choice – not a random public key.

5.4 A CL-PKE Scheme with OWE Security

The algorithms for BasicCL-PKE, our OWE secure CL-PKE scheme, are as follows:

Setup: This algorithm runs as follows:

1. Run $\mathcal{IG}$ on input k to generate output $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$. Recall the definition of $\mathcal{IG}$ in Section 2.4.1.
2. Choose an arbitrary generator $P \in \mathbb{G}_1$.
3. Select a random master-key $s \in \mathbb{Z}_q^*$ and set $P_0 = sP$.
4. Choose cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1^*$, $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^n$ and $H_5 : \mathbb{G}_1 \rightarrow \{0,1\}^n$. Here n will be the bit-length of plaintexts.

The system parameters are $\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_0, H_1, H_2, H_5 \rangle$. The master-key is $s \in \mathbb{Z}_q^*$. The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^n$.

Partial-Private-Key-Extract: This algorithm takes as input an identifier $\mathsf{ID}_A \in \{0,1\}^*$, and carries out the following steps to construct the partial private key for entity A with identifier ID_A:

1. Compute $Q_A = H_1(\mathsf{ID}_A) \in \mathbb{G}_1^*$.
2. Output the partial private key $D_A = sQ_A \in \mathbb{G}_1^*$.

The reader will notice that the partial private key of entity A here is identical to that entity's private key in the BF ID-PKE scheme described in Section 3.2.4.1. Also notice that A can verify the correctness of the Partial-Private-Key-Extract algorithm output by checking $\hat{e}(D_A, P) = \hat{e}(Q_A, P_0)$. Observe that D_A is actually a BLS signature [36, §3] on an identifier string. In fact, D_A forms a certificate only if the identifier string binds an identity and a public key. Certificates should be publicly available so that they can be verified by any entity.

Set-Secret-Value: This algorithm takes as inputs params and an entity A's identifier ID_A. It selects a random $x_A \in \mathbb{Z}_q^*$ and outputs x_A as A's secret value.

Set-Private-Key: This algorithm takes as inputs params, entity A's partial private key D_A and A's secret value $x_A \in \mathbb{Z}_q^*$. The output of the algorithm is the pair $S_A = \langle D_A, x_A \rangle$. So the private key for A is just the pair consisting of the partial private key and the secret value.

Set-Public-Key: This algorithm takes params and entity A's secret value $x_A \in \mathbb{Z}_q^*$ as inputs and constructs A's public key as $P_A = x_A P$. A valid public key is any $P_A \in \mathbb{G}_1^*$.

Encrypt: To encrypt $M \in \mathcal{M}$ for entity A with identifier $\mathsf{ID}_A \in \{0,1\}^*$ and a public

key P_A, perform the following steps:

1. Check that P_A is in $\mathbb{G}_1^*$, if not output $\perp$. This checks the validity of the public key.
2. Compute $Q_A = H_1(\mathsf{ID}_A) \in \mathbb{G}_1^*$.
3. Choose a random value $r \in \mathbb{Z}_q^*$.
4. Compute and output the ciphertext:

$$C = \langle U, V \rangle = \langle rP, M \oplus H_2(\hat{e}(Q_A, P_0)^r) \oplus H_5(rP_A) \rangle.$$

Decrypt: Suppose $C = \langle U, V \rangle \in \mathcal{C}$. To decrypt this ciphertext using the private key $S_A = \langle D_A, x_A \rangle$, compute and output

$$V \oplus H_2(\hat{e}(D_A, U)) \oplus H_5(x_A U).$$

When C is a valid encryption of M using P_A and ID_A, it is easy to see that decrypting C using $S_A = \langle D_A, x_A \rangle$ will result in an output M. This concludes the description of BasicCL-PKE.

5.5 Security of the BasicCL-PKE Construction

In order for us to prove the security of BasicCL-PKE we need to introduce two standard PKE schemes: ElG-BasicPub and BF-BasicPub. The security of both these PKE schemes is examined in Sections 5.5.3 and 5.5.4 respectively. These proofs form a foundation which the security proofs in Section 5.5.5 are built upon. Recall that the adversaries appropriate for standard PKE schemes were defined in Section 3.5.3.

5.5.1 ElG-BasicPub

Here, we define a public key encryption scheme ElG-BasicPub. The scheme is obtained by choosing a particular $\mathbb{G}$ and modifying the parameters of the certified ElGamal encryption scheme of Section 3.2.3.1. We generate parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ using $\mathcal{IG}$, and set $\mathbb{G} = \mathbb{G}_1$ and include $\mathbb{G}_2$ and $\hat{e}$ in the parameters of the scheme of Section 3.2.3.1. The reason we explicitly include both $\mathbb{G}_2$ and $\hat{e}$ in the parameters is to highlight the context in which $\mathbb{G}_1$ is generated. This is useful for security purposes.

Formally, ElG-BasicPub is specified by four algorithms: Setup, Key-Generation, Encrypt and Decrypt.

Setup:

1. Run $\mathcal{IG}$ on input k to generate $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ with the usual properties. Choose a generator $P \in \mathbb{G}_1$.
2. Choose cryptographic hash function $H_5 : \mathbb{G}_1 \rightarrow \{0,1\}^n$.

The message and ciphertext spaces for ElG-BasicPub are $\mathcal{M} = \{0,1\}^n$ and $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^n$. The system parameters are params$= \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, H_5 \rangle$.

Key-Generation:

1. Choose a random $x \in \mathbb{Z}_q^*$ and set $R = xP$.
2. Set the public key K_{pub} to be $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, H_5, R \rangle = \langle \text{params}, R \rangle$ and the private key to be $K_{priv} = x$. Notice that the value of K_{pub} now also includes params. This is because we are looking at the PKE scheme in isolation – without any established system settings.

Encrypt: To encrypt $M \in \mathcal{M}$, perform the following steps:

1. Choose a random value $r \in \mathbb{Z}_q^*$.
2. Compute and output the ciphertext: $C = \langle rP, M \oplus H_5(rR) \rangle$.

Decrypt: To decrypt $C = \langle U, V \rangle \in \mathcal{C}$ using private key $K_{priv} = x$, compute and output: $V \oplus H_5(xU)$.

This concludes the description of EIG-BasicPub.

5.5.2 BF-BasicPub

This scheme is denoted BasicPub in [32]. Here, this scheme is specified by four algorithms: Setup, Key-Generation, Encrypt and Decrypt.

Setup:

1. Run $\mathcal{IG}$ on input k to generate $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ with the usual properties. Choose a generator $P \in \mathbb{G}_1$.
2. Choose a random $s \in \mathbb{Z}_q^*$ and set $P_0 = sP$.
3. Choose cryptographic hash function $H_2 : \mathbb{G}_2 \to \{0,1\}^n$.

The message and ciphertext spaces for BF-HybridPub are $\mathcal{M} = \{0,1\}^n$ and $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^n$. The system parameters are params$= \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_0, H_2 \rangle$.

Key-Generation:

1. Choose a random $Q \in \mathbb{G}_1^*$.
2. Set the public key to be $K_{pub} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_0, H_2, Q \rangle = \langle \text{params}, Q \rangle$ and the private key to be $K_{priv} = sQ \in \mathbb{G}_1^*$.

Encrypt: To encrypt $M \in \mathcal{M}$, choose a random $r \in \mathbb{Z}_q^*$ and set the ciphertext to be:

$$C = \langle rP, M \oplus H_2(\hat{e}(Q, P_0)^r) \rangle.$$

Decrypt: To decrypt $C = \langle U, V \rangle \in \mathcal{C}$ using the private key $K_{priv} = sQ$ compute:

$$V \oplus H_2(\hat{e}(sQ, U)) = M.$$

This concludes the description of BF-HybridPub.

5.5.3 Security of ElG-BasicPub

Lemma 5.1 Suppose that H_5 is a random oracle and that there exists an OWE adversary $\mathcal{A}$ against ElG-BasicPub with advantage ϵ which makes at most q_5 queries to H_5. Then there is an algorithm $\mathcal{B}$ that solves the CDHP in $\mathbb{G}_1$ with advantage at least $(\epsilon - \frac{1}{2^n})/q_5$ and which runs in time $\mathcal{O}(time(\mathcal{A}))$. Here $\mathbb{G}_1$ is obtained from the output $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ of $\mathcal{IG}$.

Proof. Let $\mathcal{A}$ be an OWE adversary against ElG-BasicPub who makes at most q_5 queries to random oracle H_5 and who has advantage ϵ. We show how to construct an algorithm $\mathcal{B}$ which interacts with $\mathcal{A}$ to solve the CDHP in $\mathbb{G}_1$.

Suppose $\mathcal{B}$ has as inputs $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ and $\langle P, aP, bP \rangle$ (where $a, b \in \mathbb{Z}_q^*$ are unknown to $\mathcal{B}$). Let $D = abP \in \mathbb{G}_1$ denote the solution to the CDHP on these inputs. Algorithm $\mathcal{B}$ creates a public key $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, H_2, R \rangle$ for $\mathcal{A}$ by setting $R = aP$. Then $\mathcal{B}$ gives this public key to $\mathcal{A}$. Algorithm $\mathcal{B}$ now sets $U = bP$, chooses V randomly from $\{0,1\}^n$, and gives $\mathcal{A}$ the challenge ciphertext $C = \langle U, V \rangle$.

Notice that the (unknown) private key is now $D = abP$ and the (unknown) decryption of C is $M = V \oplus H_5(D)$. Hence the solution D to the CDHP can be derived from examining $\mathcal{A}$'s H_5 queries.

To simulate H_5 queries by $\mathcal{A}$, $\mathcal{B}$ maintains a list of pairs $\langle Z_j, H_{5,j}\rangle$. To respond to an H_5 query Z, $\mathcal{B}$ first checks if $Z = Z_j$ for some Z_j already on the list. If it is, then $\mathcal{B}$ responds with $H_{5,j}$. Otherwise, $\mathcal{B}$ chooses H uniformly at random from $\{0,1\}^n$ and places $\langle Z, H\rangle$ on the H_5 list.

Eventually, $\mathcal{A}$ will output its guess M' for the decryption of C. Now $\mathcal{B}$ chooses a random pair $\langle Z_j, H_{5,j}\rangle$ from the H_5 list and outputs $Z_j \in \mathbb{G}_1$ as the solution to the CDHP. (If the list is empty, $\mathcal{B}$ just outputs a random element of $\mathbb{G}_1$.)

It is easy to see that $\mathcal{A}$'s view in $\mathcal{B}$'s simulation is the same as in a real attack. So $\mathcal{A}$'s advantage in this simulation will be ϵ. We let $\mathcal{H}$ be the event that D is queried of H_5 during $\mathcal{B}$'s simulation and let δ denote the probability that event $\mathcal{H}$ occurs. Now

$$\begin{aligned} \epsilon &= \Pr[M' = M] \\ &= \Pr[M' = M|\mathcal{H}]\Pr[\mathcal{H}] + \Pr[M' = M|\neg\mathcal{H}]\Pr[\neg\mathcal{H}] \\ &\leq \delta + \tfrac{1}{2^n}(1-\delta) \end{aligned}$$

where we have used the fact that if $\mathcal{H}$ does not occur, then H_5 has not been queried on input D, so that $\mathcal{A}_{II}$'s view must be independent of the value of M.

Rearranging, we see that $\delta \geq \epsilon - \frac{1}{2^n}$. Since $\mathcal{B}$'s output is of the form Z_j chosen randomly from the H_5 list, we see that $\mathcal{B}$'s success probability is at least δ/q_5. The lemma follows.

□

5.5.4 Security of BF-BasicPub

The following result concerning the OWE security of BF-BasicPub is proven by Boneh and Franklin in in [33, Lemma 4.3].

Result 5.2 Suppose that H_2 is a random oracle. Suppose there exists an OWE adversary $\mathcal{A}$ against BF-BasicPub which makes at most q_2 queries to H_2 and which has advantage ϵ. Then there exists an algorithm $\mathcal{B}$ to solve the BDHP which runs in time $O(time(\mathcal{A}))$ and has advantage at least $(\epsilon - \frac{1}{2^n})/q_2$.

5.5.5 Security of BasicCL-PKE

Lemma 5.3 Suppose that H_1 and H_2 are random oracles and that there exists an Type II OWE adversary $\mathcal{A}_{II}$ against BasicCL-PKE with advantage ϵ which makes at most q_1 queries to H_1. Then there is an OWE adversary against ElG-BasicPub with advantage at least ϵ/q_1 which runs in time $\mathcal{O}(time(\mathcal{A}_{II}))$.

Proof. Let $\mathcal{A}_{II}$ be a Type II OWE adversary against BasicCL-PKE. Suppose $\mathcal{A}_{II}$ has advantage ϵ and makes q_1 queries to random oracle H_1. We show how to construct from $\mathcal{A}_{II}$ an OWE adversary $\mathcal{B}$ against the PKE scheme ElG-BasicPub.

Let $\mathcal{C}$ denote the challenger against our OWE adversary $\mathcal{B}$ for ElG-BasicPub. The challenger $\mathcal{C}$ begins by supplying $\mathcal{B}$ with a public key

$$K_{pub} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, H_5, R \rangle = \langle \mathsf{params}, R \rangle.$$

Adversary $\mathcal{B}$ mounts an OWE attack on the key K_{pub} using help from $\mathcal{A}_{II}$ as follows.

First of all $\mathcal{B}$ chooses an index I with $1 \leq I \leq q_1$. Then $\mathcal{B}$ simulates the algorithm Setup of BasicCL-PKE for $\mathcal{A}_{II}$ by choosing a random $s \in \mathbb{Z}_q^*$, setting $P_0 = sP$ and supplying $\mathcal{A}_{II}$ with $\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_0, H_1, H_2, H_5 \rangle$ and the value s. Here, H_1 and H_2 are additional random oracles.

Adversary $\mathcal{A}_{II}$ may make queries of H_1 or H_2 at any time. These are handled as follows:

H_1 **queries:** The H_1 queries are simulated by $\mathcal{B}$. For an ID_i query, $\mathcal{B}$ will choose a random $Q_i \in \mathbb{G}_1^*$ and return $H_1(\mathsf{ID}_i) = Q_i$ for $1 \leq i \leq q_1$. For each i where $i \neq I$, $\mathcal{B}$ chooses a random $x_i \in \mathbb{Z}_q$ and maintains a table with entries $\langle Q_i, x_i \rangle$.

H_2 **queries:** Adversary $\mathcal{B}$ simulates these and answers H_2 queries by maintaining a list of queries and replies. We do need to assume in the course of the proof that H_2 is a random oracle.

Phase 1: Now $\mathcal{A}_{II}$ launches Phase 1 of its attack, by making a series of requests, each of which is either a private key extraction or a request for a public key for a particular entity. (Recall that a Type II adversary cannot replace public keys and can make partial private key extraction queries for himself given s.) We assume that $\mathcal{A}_{II}$ always makes the appropriate H_1 query on ID before making one of these requests for that identifier. $\mathcal{B}$ replies to these requests as follows:

Private Key Extraction: If the request is on ID_I then $\mathcal{B}$ aborts. Otherwise, if the request is on ID_i with $i \neq I$, then $\mathcal{B}$ outputs $\langle sQ_i, x_i \rangle$.

Request for Public Key: If the request is on ID_I then $\mathcal{B}$ returns R. Otherwise, if the request is on ID_i for some i with $i \neq I$, then $\mathcal{B}$ returns $x_i P$.

Challenge Phase: At some point, $\mathcal{A}_{II}$ decides to end Phase 1 and picks ID_{ch} on which it wants to be challenged. We can assume that ID_{ch} has already been queried of H_1 but $\mathcal{A}_{II}$ has not extracted the private key for this identifier. Algorithm $\mathcal{B}$ responds as follows. If $\mathsf{ID}_{\text{ch}} \neq \mathsf{ID}_I$ then $\mathcal{B}$ aborts. Otherwise $\mathcal{B}$ requests from $\mathcal{C}$ a challenge ciphertext. $\mathcal{C}$ picks a random $M \in \mathcal{M}$ and responds with the challenge ciphertext $C' = \langle U', V' \rangle$, such that C' is the EIG-BasicPub encryption of M under K_{pub}. Then $\mathcal{B}$ computes $\xi' = \hat{e}(U', sQ_I)$ and sets $C^* = \langle U', V' \oplus H_2(\xi') \rangle$ and delivers C^* to $\mathcal{A}_{II}$. It is not hard to see that C^* is the BasicCL-PKE encryption of M for identifier ID_I (with public key R).

Phase 2: Adversary $\mathcal{B}$ continues to respond to requests in the same way as it did in Phase 1. Of course, we now restrict $\mathcal{A}_{II}$ to not make private key extraction requests on ID_{ch}.

Guess: Eventually, $\mathcal{A}_{II}$ will make a guess M'. Algorithm $\mathcal{B}$ outputs M' as its guess for the decryption of C^*.

Analysis: Now we analyze the behavior of $\mathcal{B}$ and $\mathcal{A}_{II}$ in this simulation. We claim that if algorithm $\mathcal{B}$ does not abort during the simulation then algorithm $\mathcal{A}_{II}$'s view is identical to its view in the real attack. Moreover, if $\mathcal{B}$ does not abort then

$\Pr[M = M'] \geq \epsilon$.

We justify this claim as follows. $\mathcal{B}$'s responses to H_1 and H_2 queries are uniformly and independently distributed in $\mathbb{G}_1^*$ and $\{0,1\}^n$ respectively, as in the real attack. All responses to $\mathcal{A}_{II}$'s requests are valid, provided of course that $\mathcal{B}$ does not abort. Furthermore, the challenge ciphertext C^* is a valid BasicCL-PKE encryption of M. Thus, by definition of algorithm $\mathcal{A}_{II}$ we have that $\Pr[M = M'] \geq \epsilon$.

The probability that $\mathcal{B}$ does not abort during the simulation remains to be calculated. Examining the simulation, we see that $\mathcal{B}$ can abort for two reasons: (i) because $\mathcal{A}_{II}$ made a private key extraction on ID_I at some point, or (ii) because $\mathcal{A}_{II}$ did not choose $\mathsf{ID}_{\text{ch}} = \mathsf{ID}_I$. We name the events that can cause $\mathcal{B}$ to abort as $\mathcal{Q}_1$ and $\mathcal{Q}_2$.

Notice that the event $\neg\mathcal{Q}_2$ implies the event $\neg\mathcal{Q}_1$ (if $\mathcal{A}_{II}$ chooses ID_{ch} equal to ID_I, then no private key extraction on ID_I is allowed). Hence we have

$$\begin{aligned} \Pr[\mathcal{B} \text{ does not abort}] &= \Pr[\neg\mathcal{Q}_1 \wedge \neg\mathcal{Q}_2] \\ &= \Pr[\neg\mathcal{Q}_2] \\ &= 1/q_1 \end{aligned}$$

where the last equality follows from $\mathcal{B}$'s random choice of I being independent of $\mathcal{A}_{II}$'s choice of ID_{ch}.

Thus we see that $\mathcal{B}$'s advantage is at least ϵ/q_1 and the proof is complete.

□

Lemma 5.4 Suppose that H_1, H_2 and H_5 are random oracles and that there exists an Type I OWE adversary $\mathcal{A}_I$ against BasicCL-PKE. Suppose $\mathcal{A}_I$ has advantage ϵ, runs in time t, makes at most q_1, q_2 and q_5 queries to H_1, H_2 and H_5 respectively. Then there is an algorithm $\mathcal{B}$ which acts as either a BF-BasicPub or an EIG-BasicPub OWE adversary. Moreover, $\mathcal{B}$ either has advantage at least $\epsilon/4q_1$ when playing as a BF-BasicPub adversary, or has advantage at least $\epsilon/4q_1$ when playing as an EIG-BasicPub adversary. Algorithm $\mathcal{B}$ runs in time $\mathcal{O}(time(\mathcal{A}_I))$.

Proof. Let $\mathcal{A}_I$ be a Type I IND-CCA adversary against BasicCL-PKE. Suppose $\mathcal{A}_I$ has advantage ϵ, runs in time t, makes q_1, q_2 and q_5 queries to random oracles H_1,

H_2 and H_5 respectively. We show how to construct from $\mathcal{A}_I$ an adversary $\mathcal{B}$ that acts either as an OWE adversary against the PKE scheme BF-BasicPub or as an OWE adversary against the PKE scheme ElG-BasicPub. We assume that challengers $\mathcal{C}_I$ and $\mathcal{C}_{II}$ for both types of games are available to $\mathcal{B}$.

Adversary $\mathcal{B}$ begins by choosing a random bit c and an index I uniformly at random with $1 \leq I \leq q_1$. If $c = 0$, then $\mathcal{B}$ chooses to play against $\mathcal{C}_I$ and aborts $\mathcal{C}_{II}$. Here, $\mathcal{B}$ will build an OWE adversary against BF-BasicPub and fail against $\mathcal{C}_{II}$. When $c = 1$, $\mathcal{B}$ chooses to play against $\mathcal{C}_{II}$ and aborts $\mathcal{C}_I$. Here, $\mathcal{B}$ will build a OWE adversary against ElG-HybridPub and fail against $\mathcal{C}_I$. In either case, $\mathcal{C}$ will denote the challenger against which $\mathcal{B}$ plays for the remainder of this proof.

We define three events $\mathcal{H}$, $\mathcal{F}_0$ and $\mathcal{F}_1$:

- $\mathcal{H}$: Adversary $\mathcal{A}_I$ chooses ID_I as the challenge identifier ID_{ch}.
- $\mathcal{F}_0$: Adversary $\mathcal{A}_I$ extracts the partial private key for entity ID_I.
- $\mathcal{F}_1$: Adversary $\mathcal{A}_I$ replaces the public key of entity ID_I at some point in its attack.

The general strategy of the proof is as follows. If $(c = 0) \wedge \mathcal{F}_0$ occurs, $\mathcal{B}$ will have to abort and will be unsuccessful. If $\neg\mathcal{F}_0 \wedge \mathcal{H}$ occurs, then $\mathcal{B}$'s success probability will be related to that of $\mathcal{A}_I$. On the other hand, if $(c = 1) \wedge \mathcal{F}_1$ occurs, $\mathcal{B}$ will again have to abort and will be unsuccessful. If $\neg\mathcal{F}_1 \wedge \mathcal{H}$ occurs, then $\mathcal{B}$'s success probability will again be related to that of $\mathcal{A}_I$. Overall, we will show that $\mathcal{B}$'s advantage in its mixed-game strategy is non-negligible if $\mathcal{A}_I$'s is. It is then easy to see that $\mathcal{B}$ has a non-negligible advantage for at least one of the two game types.

If $c = 0$, then $\mathcal{C}$ is an OWE challenger for BF-BasicPub and begins by supplying $\mathcal{B}$ with a public key $K_{pub} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_0, H_2, Q \rangle$. If $c = 1$, then $\mathcal{C}$ is an OWE challenger for ElG-BasicPub and so supplies $\mathcal{B}$ with a public key $K_{pub} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, H_5, R \rangle$.

Then $\mathcal{B}$ simulates the algorithm Setup of BasicCL-PKE for $\mathcal{A}_I$. When $c = 0$, $\mathcal{B}$ will handle H_5 queries, while when $c = 1$, $\mathcal{B}$ will handle H_2 queries. Additionally, when $c = 1$, $\mathcal{B}$ chooses a random $s \in \mathbb{Z}_q^*$ and sets $P_0 = sP$. Thus, $\mathcal{B}$ supplies $\mathcal{A}_I$ with params= $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_0, H_1, H_2, H_5 \rangle$. Here H_1 is a random oracle that will be controlled by $\mathcal{B}$.

Adversary $\mathcal{A}_I$ may make queries of the random oracles H_1, H_2 and H_5, at any time during its attack. These are handled as follows:

H_1 **queries:** Adversary $\mathcal{B}$ maintains a list of tuples $\langle \mathsf{ID}_i, Q_i, b_i, x_i, P_i \rangle$ which we call the H_1 list. The list is initially empty, and when $\mathcal{A}_I$ queries H_1 on input $\mathsf{ID} \in \{0,1\}^*$, $\mathcal{B}$ responds as follows:

1. If ID already appears on the H_1 list in a tuple $\langle \mathsf{ID}_i, Q_i, b_i, x_i, P_i \rangle$, then $\mathcal{B}$ responds with $H_1(\mathsf{ID}) = Q_i$.
2. Suppose ID does not already appear on the list and ID is the I-th distinct H_1 query made by $\mathcal{A}_I$. For $c = 0$, $\mathcal{B}$ outputs $H_1(\mathsf{ID}) = Q$, selects a random $x_I \in \mathbb{Z}_q^*$ and adds the entry $\langle \mathsf{ID}, Q, \perp, x_I, x_I P \rangle$ to the H_1 list. For $c = 1$, $\mathcal{B}$ selects $b_I \in \mathbb{Z}_q^*$, outputs $H_1(\mathsf{ID}) = b_I P$ and adds the entry $\langle \mathsf{ID}, b_I P, b_I, \perp, R \rangle$ to the H_1 list.
3. Otherwise, when ID does not already appear on the list and ID is the i-th distinct H_1 query made by $\mathcal{A}_I$ where $i \neq I$, $\mathcal{B}$ picks random $x_i, b_i \in \mathbb{Z}_q^*$, sets $Q_i = b_i P$, outputs $H_1(\mathsf{ID}) = Q_i$ and adds $\langle \mathsf{ID}, b_i P, b_i, x_i, x_i P \rangle$ to the H_1 list.

Notice that with this specification of H_1, the BasicCL-PKE partial private key for ID_i $(i \neq I)$ is equal to $b_i P_0$ while the public key for ID_i $(i \neq I)$ is $P_i = x_i P$ and the private key for ID_i $(i \neq I)$ is $\langle b_i P_0, x_i \rangle$. These can all be computed by $\mathcal{B}$. When $c = 1$, $\mathcal{B}$ sets the public key of ID_I to be R and can compute the partial private key of ID_I as $sb_I P$. When $c = 0$, $\mathcal{B}$ knows neither the partial private key nor the private key for ID_I.

H_2 queries: When $c = 0$ any H_2 queries made by $\mathcal{A}_I$ are passed to $\mathcal{C}$ to answer. When $c = 1$ any H_2 queries made by $\mathcal{A}_I$ are simulated by $\mathcal{B}$ using the standard approach of maintaining a list of queries and replies. We do need to assume in the course of the proof that H_2 is a random oracle.

H_5 queries: Any H_5 queries made by $\mathcal{A}_I$ are passed to $\mathcal{C}$ to answer when $c = 1$. When $c = 0$, $\mathcal{B}$ maintains a list of tuples $\langle \mu_i, H_{5,i} \rangle$ which we call the H_5 list. The list is initially empty, and when $\mathcal{A}_I$ queries H_5 on input $\mu \in \mathbb{G}_1$, $\mathcal{B}$ responds as follows:

1. If μ already appears on the H_5 list in a tuple $\langle \mu_i, H_{5,i} \rangle$, then $\mathcal{B}$ responds with $H_5(\mu) = H_{5,i}$.
2. Suppose μ does not already appear on the list. If the H_5 query is made before the challenge phase, then $\mathcal{B}$ goes to step 3 below. Otherwise, let P_{ch} denote the value of the public key for the challenge identifier ID_{ch} during the challenge phase, let $C^* = \langle U^*, V^* \rangle$ be the challenge ciphertext delivered to $\mathcal{A}_I$ by $\mathcal{B}$, and let ξ be the value, to be defined below, used by $\mathcal{B}$ in the challenge phase. $\mathcal{B}$ tests if μ satisfies $\hat{e}(\mu, P) = \hat{e}(U^*, P_{\text{ch}})$. If equality holds, then $\mathcal{B}$ adds $\langle \mu, \xi \rangle$ to the H_5 list and outputs $\xi = H_5(\mu)$. If the equality does not hold, then $\mathcal{B}$ goes to step 3.
3. Supposing μ to be the i-th distinct H_5 query made by $\mathcal{A}_I$, $\mathcal{B}$ selects a random $H_{5,i} \in \{0,1\}^n$, outputs $H_5(\mu) = H_{5,i}$ and adds $\langle \mu_i, H_{5,i} \rangle$ to the H_5 list.

Informally, the reason we simulate H_5 this way is to make sure that if $\mathcal{A}_I$ queries H_5 in the course of its attack, then H_5 behaves consistently and produces the same output as that produced during it use in the construction of the challenge ciphertext. Recall that the value of P_{ch} could be selected by $\mathcal{A}_I$, hence, the corresponding private key is unknown to $\mathcal{B}$ – and it could also be unknown to $\mathcal{A}_I$.

Phase 1: After receiving params from $\mathcal{B}$, $\mathcal{A}_I$ launches Phase 1 of its attack, by making a series of requests, each of which is either a partial private key extraction for an entity, a private key extraction for an entity, a request for a public key for

an entity or a replacement of a public key for an entity. We assume that $\mathcal{A}_I$ always makes the appropriate H_1 query on the identifier ID for that entity before making one of these requests. $\mathcal{B}$ replies to these requests as follows:

Partial Private Key Extraction: Suppose the request is on ID_i. There are three cases:

1. If $i \neq I$, then $\mathcal{B}$ replies with $b_i P_0$.
2. If $i = I$ and $c = 1$, then $\mathcal{B}$ replies with $b_I P_0$.
3. If $i = I$ and $c = 0$, then $\mathcal{B}$ aborts.

Private Key Extraction: Suppose the request is on ID_i. We can assume that the public key for ID_i has not been replaced. There are two cases:

1. If $i \neq I$, then $\mathcal{B}$ outputs $\langle b_i P_0, x_i \rangle$.
2. If $i = I$, then $\mathcal{B}$ aborts.

Request for Public Key: If the request is on ID_i then $\mathcal{B}$ returns P_i by accessing the H_1 list.

Replace Public Key: Suppose the request is to replace the public key for ID_i with value P_i'. There are three cases:

1. If $i = I$ and $c = 1$, then $\mathcal{B}$ aborts.
2. If $i = I$ and $c = 0$, then $\mathcal{B}$ replaces the current entry in the H_1 list with the new entry P_I' and updates the tuple to $\langle \mathsf{ID}_I, Q, \perp, \perp, P_I' \rangle$.
3. Otherwise, $\mathcal{B}$ replaces the current entry in the H_1 list with the new entry P_i' $(i \neq I)$ and updates the tuple to $\langle \mathsf{ID}_i, b_i P, b_i, \perp, P_i' \rangle$.

Challenge Phase: At some point, $\mathcal{A}_I$ should decide to end Phase 1 and pick ID_{ch} on which it wishes to be challenged. We can assume that ID_{ch} has already been queried of H_1 but that $\mathcal{A}_I$ has not extracted the private key for this identifier. Algorithm $\mathcal{B}$ responds as follows. If $\mathsf{ID}_{\text{ch}} \neq \mathsf{ID}_I$ then $\mathcal{B}$ aborts. Now $\mathcal{B}$ requests a challenge ciphertext of its challenger $\mathcal{C}$. There are now two cases:

- When $c = 0$, $\mathcal{C}$ picks a random $M \in \mathcal{M}$ and responds with the challenge ciphertext $C' = \langle U', V' \rangle$, a BF-BasicPub encryption of M under K_{pub}. Now $\mathcal{B}$ checks each entry $\langle \mu_i, H_{5,i} \rangle$ in the H_5 list to see if it satisfies the equality $\hat{e}(\mu_i, P) = \hat{e}(U', P_{\text{ch}})$. It is easy to see that at most one entry can do so. If $\mathcal{B}$ finds that the j-th entry satisfies the equality, then $\mathcal{B}$ sets $C^* = \langle U', V' \oplus H_{5,j} \rangle$ and delivers C^* to $\mathcal{A}_I$ as the challenge ciphertext. Otherwise, if no entry satisfies this test, $\mathcal{B}$ selects a random $\xi \in \{0,1\}^n$, sets $C^* = \langle U', V' \oplus \xi \rangle$ and delivers C^* to $\mathcal{A}_I$.
- When $c = 1$, $\mathcal{C}$ picks a random $M \in \mathcal{M}$ and responds with the challenge ciphertext $C' = \langle U', V' \rangle$, such that C' is the EIG-BasicPub encryption of M under K_{pub}. Then $\mathcal{B}$ sets $C^* = \langle U', V' \oplus H_2(\hat{e}(U', b_I s P)) \rangle$ and delivers C^* to $\mathcal{A}_I$.

It is easy to see that in both cases C^* is the BasicCL-PKE encryption of M for identifier ID_{ch} under public key P_{ch}. We now let P_{ch} denote the particular value of the public key for identifier ID_{ch} during the challenge phase ($\mathcal{A}_I$ may change this value in Phase 2 of its attack).

Phase 2: Adversary $\mathcal{B}$ continues to respond to $\mathcal{A}_I$'s requests in the same way as it did in Phase 1. However, the same restrictions, as identified in Section 5.3, on $\mathcal{A}_I$'s behaviour apply in this phase.

Guess: Eventually, $\mathcal{A}_I$ will make a guess M'. Algorithm $\mathcal{B}$ outputs M' as its guess for the decryption of C^*.

Analysis: Now we analyze the behavior of $\mathcal{B}$ and $\mathcal{A}_I$ in this simulation. We claim that if algorithm $\mathcal{B}$ does not abort during the simulation, then algorithm $\mathcal{A}_I$'s view is identical to its view in the real attack. Moreover, if this is the case, then $\Pr[M = M'] \geq \epsilon$. This is not hard to see: Adversary $\mathcal{B}$'s responses to all hash queries are uniformly and independently distributed as in the real attack. All responses to $\mathcal{A}_I$'s requests are valid, provided of course that $\mathcal{B}$ does not abort. Furthermore, the challenge ciphertext C^* is a valid BasicCL-PKE encryption of M under the current public key for identifier ID_{ch}. Thus, by definition of algorithm $\mathcal{A}_I$ we have that $\Pr[M = M'] \geq \epsilon$.

So we must examine the probability that $\mathcal{B}$ does not abort during the simulation. Examining the simulation, we see that $\mathcal{B}$ can abort for one of four reasons:

0. Because $c = 0$ and the event $\mathcal{F}_0$ occurred during the simulation.
1. Because $c = 1$ and event $\mathcal{F}_1$ occurred during the simulation.
2. Because $\mathcal{A}_I$ made a private key extraction on ID_I at some point.
3. Or because $\mathcal{A}_I$ chose $\mathsf{ID}_{\text{ch}} \neq \mathsf{ID}_I$.

We name the event $(c = i) \wedge \mathcal{F}_i$ as $\mathcal{H}_i$ for $i = 0, 1$. We also name the last two events here as $\mathcal{F}_2$ and $\mathcal{F}_3$. Of course, $\mathcal{F}_3$ is the same as event $\neg\mathcal{H}$. Now $\mathcal{A}_I$ makes q_1 queries of H_1 and chooses ID_{ch} from amongst the responses ID_i, while $\mathcal{B}$'s choice of I is made uniformly at random from the set of q_1 indices i. So the probability that $\mathsf{ID}_{\text{ch}} = \mathsf{ID}_I$ is equal to $1/q_1$. Hence $\Pr[\mathcal{H}] = 1/q_1$. Notice too that the event $\neg\mathcal{F}_3$ implies the event $\neg\mathcal{F}_2$ (if $\mathcal{A}_I$ chooses $\mathsf{ID}_{\text{ch}} = \mathsf{ID}_I$, then no private key extraction on ID_I is allowed). Gathering this information together, we have:

$$\begin{aligned}
\Pr[\mathcal{B} \text{ does not abort}] &= \Pr[\neg\mathcal{H}_0 \wedge \neg\mathcal{H}_1 \wedge \neg\mathcal{F}_2 \wedge \neg\mathcal{F}_3] \\
&= \Pr[\neg\mathcal{H}_0 \wedge \neg\mathcal{H}_1 \wedge \mathcal{H}] \\
&= \Pr[\neg\mathcal{H}_0 \wedge \neg\mathcal{H}_1 | \mathcal{H}] \cdot \Pr[\mathcal{H}] \\
&= \tfrac{1}{q_1} \cdot \Pr[\neg\mathcal{H}_0 \wedge \neg\mathcal{H}_1 | \mathcal{H}].
\end{aligned}$$

Notice now that the events $\mathcal{H}_0$ and $\mathcal{H}_1$ are mutually exclusive (because one involves $c = 0$ and the other $c = 1$). Therefore we have

$$\Pr[\neg\mathcal{H}_0 \wedge \neg\mathcal{H}_1|\mathcal{H}] = 1 - \Pr[\mathcal{H}_0|\mathcal{H}] - \Pr[\mathcal{H}_1|\mathcal{H}].$$

Moreover,

$$\begin{aligned}\Pr[\mathcal{H}_i|\mathcal{H}] &= \Pr[(c = i) \wedge \mathcal{F}_i|\mathcal{H}] \\ &= \Pr[\mathcal{F}_i|(\mathcal{H} \wedge (c = i))] \cdot \Pr[c = i] \\ &= \tfrac{1}{2}\Pr[\mathcal{F}_i|\mathcal{H}]\end{aligned}$$

where the last equality follows because the event $\mathcal{F}_i|\mathcal{H}$ is independent of the event $c = i$. So we have

$$\Pr[\mathcal{B} \text{ does not abort}] = \frac{1}{q_1}\left(1 - \frac{1}{2}\Pr[\mathcal{F}_0|\mathcal{H}] - \frac{1}{2}\Pr[\mathcal{F}_1|\mathcal{H}]\right).$$

Finally, we have that $\Pr[\mathcal{F}_0 \wedge \mathcal{F}_1|\mathcal{H}] = 0$ because of the rules on adversary behaviour described in Section 5.2 (an adversary cannot both extract the partial private key and change the public key of the challenge identifier). This implies that $\Pr[\mathcal{F}_0|\mathcal{H}] + \Pr[\mathcal{F}_1|\mathcal{H}] \leq 1$. Hence we see that

$$\Pr[\mathcal{B} \text{ does not abort}] \geq \frac{1}{2q_1}.$$

It is now easy to see that $\mathcal{B}$'s advantage is at least $\frac{\epsilon}{2q_1}$. It follows that either $\mathcal{B}$'s advantage as a Type I adversary against BF-BasicPub or $\mathcal{B}$'s advantage as a Type II adversary against ElG-BasicPub is at least $\frac{\epsilon}{4q_1}$. The running time of $\mathcal{B}$ is $\mathcal{O}(time(\mathcal{A}_I))$. This completes the proof of the lemma.

□

Next is our main theorem about the security of BasicCL-PKE in our OWE model.

Theorem 5.5 Let H_1, H_2 and H_5 be random oracles. Suppose further that there is no polynomially bounded algorithm that can solve the BDHP with non-negligible advantage. Then BasicCL-PKE is OWE secure.

Proof. The proof of Theorem 5.5 is performed in two parts, one for a Type I adversary and one for a Type II adversary. We first consider a Type I adversary.

Type I adversary: Lemma 5.4 provides a reduction relating the OWE security of BasicCL-PKE to that of ElG-BasicPub or BF-BasicPub in the OWE model for standard PKE. Lemma 5.1 and Result 5.2 relate the security of these PKE schemes to the hardness of the CDHP or BDHP respectively.

By actually composing the intermediate security results we can relate the security of BasicCL-PKE against Type I adversaries directly to the hardness of the BDHP or CDHP. Suppose hash functions H_1, H_2 and H_5 are random oracles. Suppose $\mathcal{A}_I$ is a Type I adversary against BasicCL-PKE, that runs in time time($\mathcal{A}_I$) and has advantage ϵ against BasicCL-PKE. Then there is an algorithm $\mathcal{B}$ with running time $O(\text{time}(\mathcal{A}_I))$. Algorithm $\mathcal{B}$ either solves the CDHP in $\mathbb{G}_1$ with advantage at least

$$\frac{1}{q_5}\left(\frac{\epsilon}{4q_1} - \frac{1}{2^n}\right).$$

or algorithm $\mathcal{B}$ solves the BDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}\rangle$ with advantage at least

$$\frac{1}{q_2}\left(\frac{\epsilon}{4q_1} - \frac{1}{2^n}\right).$$

Type II adversary: Lemma 5.3 shows that the OWE security of BasicCL-PKE can be reduced to the OWE security of a related (normal) public key encryption scheme ElG-BasicPub. Lemma 5.1 relates the security of ElG-BasicPub to the hardness of the CDHP in $\mathbb{G}_1$.

As above, we can relate the security of BasicCL-PKE against Type I adversaries directly to the hardness of the CDHP. Suppose hash functions H_1, H_2 and H_5 are random oracles. Suppose $\mathcal{A}_{II}$ is a Type II adversary against BasicCL-PKE, that runs in time time($\mathcal{A}_{II}$), and has advantage ϵ against BasicCL-PKE. Then there is an algorithm $\mathcal{B}$ with running time $O(\text{time}(\mathcal{A}_{II}))$ that solves the CDHP in $\mathbb{G}_1$ with advantage at least

$$\frac{1}{q_5}\left(\frac{\epsilon}{q_1} - \frac{1}{2^n}\right).$$

Compared to the CDHP (in $\mathbb{G}_1$ output by algorithm $\mathcal{IG}(k)$), the BDHP (in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}\rangle$ output by the same algorithm $\mathcal{IG}(k)$) is an easier problem, in the sense that an

algorithm to solve the CDHP in $\mathbb{G}_1$ can be transformed into an algorithm to solve the BDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ – see the proof in p.29. Hence the security of BasicCL-PKE rests on the hardness of the BDHP.

□

5.6 Summary

In this chapter, we showed how our concept of certificateless public key cryptography can be realized by specifying a certificateless public key encryption (CL-PKE) scheme that is based on bilinear maps. The scheme BasicCL-PKE has one-way encryption security in the random oracle model, assuming that the BDHP is hard. In Chapter 6 we will build on and improve the CL-PKE scheme presented in this chapter. We will construct a scheme which is secure in a more robust model, one allowing decryption queries by the adversary and in which the adversary's task is to distinguish which of the two messages has been encrypted.

Chapter 6

CL-PKE – Semantic Security

Contents

In this chapter we develop an adversarial model for a certificateless public key encryption (CL-PKE) scheme. The adversarial model is fully adaptive, it captures an adversary who has access to the master key (but does not replace public keys) and another who can replace public keys. We propose an efficient CL-PKE scheme and prove that it is secure in the fully adaptive adversarial model, provided that the BDHP is hard.

6.1 Introduction

The work presented in this chapter focuses on CL-PKE schemes. We will present an adversarial model for CL-PKE which formally captures the adversarial capabilities. The model we present in this chapter is a natural generalization of the fully adaptive, multi-user model of [32] to the CL-PKC setting, and involves two distinct types of adversary: one who can replace public keys at will and another who has knowledge of the master key but does not replace public keys. Recall that the derived OWE security model for CL-PKE in Chapter 5 was weak and not fully adaptive, we will rectify this here by giving adversaries additional adaptive access to decryption oracles. The stronger model of this chapter which is fully-adaptive and involves the indistinguishibility of encryptions, is used to prove the semantic security of a concrete, efficient CL-PKE scheme. The semantically secure CL-PKE scheme of this chapter is based on the owe secure CL-PKE scheme of Chapter 5.

6.2 IND-CCA Security Model for CL-PKE

Given the formal definition of a CL-PKE scheme in Section 5.2, and the subsequent security treatment of CL-PKE, we are now in a position to define stronger adversaries for CL-PKE than that of Chapter 5. The definition will involve the indistinguishability of encryptions against a fully-adaptive chosen ciphertext (IND-CCA) attacker.

Recall the definition of IND-CCA security for PKE in Section 3.5.3. In that definition, there are two parties, the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$. The adversary operates in three phases after being presented with a random public key. In Phase 1, $\mathcal{A}$ may make decryption queries on ciphertexts of its choice. In the Challenge Phase, $\mathcal{A}$ chooses two messages M_0, M_1 and is given a challenge ciphertext C^* for one of these two messages M_b by the challenger. In Phase 2, $\mathcal{A}$ may make further decryption queries, but may not ask for the decryption of C^*. The attack ends with $\mathcal{A}$'s guess b' for the bit b. The adversary's advantage is defined to be

$\text{Adv}(\mathcal{A}) = 2(\Pr[b' = b] - \frac{1}{2})$[6.1].

There are two alternative ways to think of the model which we will present in this section. One is an extension of the IND-CCA model described above and the other is an extension of the model described in Chapter 5. The IND-CCA model is extended in the same way as the OWE model was extend in Chapter 5. We now briefly examine the model of Chapter 5, which was an extension to the model in [32]. The model allowed adversaries to extract partial private keys, or private keys, or both, for identities of their choice. Furthermore, it allowed for adversaries to replace the public key of any entity with a value of their choice. In this chapter we must extend this model and consider how a challenger should respond to decryption queries for identities whose public keys may have been changed. Moreover, the adversarial definition which we will present is an indistinguishability-based definition, so the nature of the challenge phase differs from that in Chapter 5.

Here then is a list of the actions that an IND-CCA adversary against a CL-PKE scheme may carry out and a discussion of how each action should be handled by the challenger for that adversary.

1. **Extract partial private key of A:** Identical to **Extract partial private key of** A in Section 5.3.
2. **Extract private key for A:** Identical to **Extract private key for** A in Section 5.3.
3. **Request public key of A:** Identical to **Request public key of** A in Section 5.3.
4. **Replace public key of A:** Identical to **Replace public key of** A in Section 5.3.

[6.1] In [32], an extension to the standard IND-CCA security model was presented for ID-PKE. This extension was labelled IND-ID-CCA and handles adversaries who can extract the private keys of arbitrary entities and who choose the identifier ID_{ch} of the entity on whose public key they are challenged.

5. **Decryption query for ciphertext C and entity A:** If $\mathcal{A}$ has not replaced the public key of entity A, then $\mathcal{C}$ responds by running the algorithm Set-Private-Key to obtain the private key S_A, then running Decrypt on ciphertext C and private key S_A and returning the output to $\mathcal{A}$. However, if $\mathcal{A}$ has already replaced the public key of A, then in following this approach, $\mathcal{C}$ will (in general) not decrypt using a private key matching the current public key. So $\mathcal{C}$'s reply to $\mathcal{A}$'s decryption query is likely to be incorrect. Indeed $\mathcal{C}$ most likely will not even know what the private key matching the current public key is! In defining our security model for CL-PKE, we have two options: we could simply accept that these decryptions will be incorrect, or we can insist that $\mathcal{C}$ should somehow properly decrypt ciphertexts even for entities whose public keys have been replaced. The former option could be argued for on grounds of reasonableness: after all, how can $\mathcal{C}$ be expected to provide correct decryptions when $\mathcal{A}$ gets to choose the public key? On the other hand, the latter option results in a more powerful security model, because now decryption queries made under public keys that have been changed will potentially be far more useful to $\mathcal{A}$. For this reason, we adopt the latter option for our model, even though it substantially complicates our proofs of security. (These decryptions will be handled using special purpose knowledge extractors in our security proofs.) Naturally, as in [32], we prohibit $\mathcal{A}$ from ever making a decryption query on the challenge ciphertext C^* for the combination of identifier ID_{ch} and public key P_{ch} that was used to encrypt M_b. However $\mathcal{A}$ is, for example, allowed to replace the public key for ID_{ch} with a new value and then request a decryption of C^*, or to change another entity A's public key to P_{ch} (or any other value) and then request the decryption of C^* for entity A.

We also want to consider adversaries who are equipped with master-key, in order to model security against an eavesdropping KGC. As discussed in Section 4.1, we do not allow such an adversary to replace public keys: in this respect, we invest in the KGC a similar level of trust as we do in a CA in a traditional PKI – recall Section 4.6.5. So by adapting the adversaries of Chapter 5 we will distinguish between two adversary types:

CL-PKE Type I IND-CCA Adversary: Such an adversary $\mathcal{A}_I$ does not have access to master-key. However, $\mathcal{A}_I$ may request public keys and replace public keys with values of its choice, extract partial private and private keys and make decryption queries, all for identities of its choice. As discussed above, we make several natural restrictions on such a Type I adversary:

1. Adversary $\mathcal{A}_I$ cannot extract the private key for ID_{ch} at any point.
2. Adversary $\mathcal{A}_I$ cannot request the private key for any identifier if the corresponding public key has already been replaced.
3. Adversary $\mathcal{A}_I$ cannot both replace the public key for the challenge identifier ID_{ch} before the challenge phase *and* extract the partial private key for ID_{ch} in some phase.
4. In Phase 2, $\mathcal{A}_I$ cannot make a decryption query on the challenge ciphertext C^* for the combination of identifier ID_{ch} and public key P_{ch} that was used to encrypt M_b.

CL-PKE Type II IND-CCA Adversary: Such an adversary $\mathcal{A}_{II}$ does have access to master-key, but may not replace public keys of entities. Adversary $\mathcal{A}_{II}$ can compute partial private keys for itself, given master-key. It can also request public keys, make private key extraction queries and decryption queries, both for identities of its choice. The restrictions on this type of adversary are:

1. Adversary $\mathcal{A}_{II}$ cannot replace public keys at any point.
2. Adversary $\mathcal{A}_{II}$ cannot extract the private key for ID_{ch} at any point.
3. In Phase 2, $\mathcal{A}_{II}$ cannot make a decryption query on the challenge ciphertext C^* for the combination of identifier ID_{ch} and public key P_{ch} that was used to encrypt M_b.

Chosen ciphertext security for CL-PKE: We say that a CL-PKE scheme is semantically secure against an adaptive chosen ciphertext attack ("IND-CCA secure") if no polynomially bounded adversary $\mathcal{A}$ of Type I or Type II has a non-negligible advantage against the challenger in the following game:

Setup: The challenger takes a security parameter k as input and runs the Setup algorithm. It gives $\mathcal{A}$ the resulting system parameters params. If $\mathcal{A}$ is of Type I, then the challenger keeps master-key to itself, otherwise, it gives master-key to $\mathcal{A}$.

Phase 1: Adversary $\mathcal{A}$ issues a sequence of requests, each request being either a partial private key extraction, a private key extraction, a request for a public key, a replace public key command or a decryption query for a particular entity. These queries may be asked adaptively, but are subject to the previously defined rules on adversary behaviour.

Challenge Phase: Once $\mathcal{A}$ decides that Phase 1 is over it outputs the challenge identifier ID_{ch} and two equal length plaintexts $M_0, M_1 \in \mathcal{M}$. Again, the adversarial constraints given above apply. In particular, ID_{ch} cannot be an identifier for which the private key has been extracted. Moreover, if $\mathcal{A}$ is of Type I, then ID_{ch} cannot be an identifier for which both the public key has been replaced and the partial private key extracted. The challenger now picks a random bit $b \in \{0, 1\}$ and computes C^*, the encryption of M_b under the current public key P_{ch} for ID_{ch}. Then C^* is delivered to $\mathcal{A}$.

Phase 2: Now $\mathcal{A}$ issues a second sequence of requests as in Phase 1, again subject to the rules on adversary behaviour above. In particular, no private key extraction on ID_{ch} is allowed, and, if $\mathcal{A}$ is of Type I, then the partial private key for ID_{ch} cannot be extracted if the corresponding public key was replaced in Phase 1. Moreover, no decryption query can be made on the challenge ciphertext C^* for the combination of identifier ID_{ch} and public key P_{ch} that was used to encrypt M_b.

Guess: Finally, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. The adversary wins the game if

$b = b'$. We define $\mathcal{A}$'s advantage in this game to be $\mathrm{Adv}(\mathcal{A}) := 2|\Pr[b = b'] - \frac{1}{2}|$.

6.3 A CL-PKE Scheme with Chosen Ciphertext Security

We showed in Section 5.4 how to combine the BF ID-PKE scheme of Section 3.2.4.1 and a variant of the ElGamal PKE Scheme of Section 3.2.3.1 to produce the OWE-secure CL-PKE scheme BasicCL-PKE. The scheme we present here is an IND-CCA version of BasicCL-PKE, obtained essentially by applying the Fujisaki-Okamoto hybridisation technique (see Section 6.4) to BasicCL-PKE. Note that the scheme presented here differs from the one given in [6]. As with BasicCL-PKE, this scheme can also be regarded as resulting from the optimisation of a double encryption construction for CL-PKE – we will present such double encryption constructions for CL-PKE in Section 7.2.

The algorithms for FullCL-PKE, our IND-CCA secure CL-PKE scheme, are as follows:

Setup: This algorithm runs as follows:

1. Run $\mathcal{IG}$ on input k to generate output $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$. Recall the definition of $\mathcal{IG}$ in Section 2.4.1.
2. Choose an arbitrary generator $P \in \mathbb{G}_1$.
3. Select a random master-key $s \in \mathbb{Z}_q^*$ and set $P_0 = sP$.
4. Choose cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1^*$, $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^n$, $H_3 : \{0,1\}^n \times \{0,1\}^n \rightarrow \mathbb{Z}_q^*$, $H_4 : \{0,1\}^n \rightarrow \{0,1\}^n$ and $H_5 : \mathbb{G}_1 \rightarrow \{0,1\}^n$. Here n will be the bit-length of plaintexts.

The system parameters are params$= \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_0, H_1, H_2, H_3, H_4, H_5 \rangle$. The master-key is $s \in \mathbb{Z}_q^*$. The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is

$\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^{2n}$. Notice that this is identical to Setup of BasicCL-PKE in Section 5.4, except for the additional hash functions H_3 and H_4.

Partial-Private-Key-Extract: This algorithm takes as input an identifier $\mathsf{ID}_A \in \{0,1\}^*$, and carries out the following steps to construct the partial private key for entity A with identifier ID_A:

1. Compute $Q_A = H_1(\mathsf{ID}_A) \in \mathbb{G}_1^*$.
2. Output the partial private key $D_A = sQ_A \in \mathbb{G}_1^*$.

The observations concerning D_A from Chapter 6 also apply to here.

The algorithms Partial-Private-Key-Extract, Set-Secret-Value, Set-Private-Key and Set-Public-Key are identical to BasicCL-PKE. They are included here for completeness.

Set-Secret-Value: This algorithm takes as inputs params and an entity A's identifier ID_A. It selects a random $x_A \in \mathbb{Z}_q^*$ and outputs x_A as A's secret value.

Set-Private-Key: This algorithm takes as inputs params, entity A's partial private key D_A and A's secret value $x_A \in \mathbb{Z}_q^*$. The output of the algorithm is the pair $S_A = \langle D_A, x_A \rangle$. So the private key for A is just the pair consisting of the partial private key and the secret value.

Set-Public-Key: This algorithm takes params and entity A's secret value $x_A \in \mathbb{Z}_q^*$ as inputs and constructs A's public key as $P_A = x_A P$. The test of validity for a public key P_A is that $P_A \in \mathbb{G}_1^*$. Comparing to the scheme in [6], we see that our public key has no 'Y_A' component, and there is no structural requirement that $Y_A = sX_A$. We will discuss this further in Chapter 8.

Encrypt: To encrypt $M \in \mathcal{M}$ for entity A with identifier $\mathsf{ID}_A \in \{0,1\}^*$ and a public key P_A, perform the following steps:

1. Check that P_A is in $\mathbb{G}_1^*$, if not output $\perp$. This checks the validity of the public key.
2. Compute $Q_A = H_1(\mathsf{ID}_A) \in \mathbb{G}_1^*$.
3. Choose a random $\sigma \in \{0,1\}^n$.
4. Set $r = H_3(\sigma, M)$.
5. Compute and output the ciphertext:

$$C = \langle rP, \sigma \oplus H_2(\hat{e}(Q_A, P_0)^r) \oplus H_5(rP_A), M \oplus H_4(\sigma) \rangle.$$

Notice that $H_2(\hat{e}(Q_A, P_0)^r)$ is identical to the mask used in the BF ID-PKE scheme in Section 3.2.4.1, while $H_5(rP_A)$ is the same as the mask used in the ElGamal PKE scheme in Section 3.2.3.1.

Decrypt: Suppose $C = \langle U, V, W \rangle \in \mathcal{C}$. To decrypt this ciphertext using the private key $S_A = \langle D_A, x_A \rangle$:

1. Compute $V \oplus H_2(\hat{e}(D_A, U)) \oplus H_5(x_A U) = \sigma'$.
2. Compute $W \oplus H_4(\sigma') = M'$.
3. Set $r' = H_3(\sigma', M')$ and test if $U = r'P$. If not, output $\perp$ and reject the ciphertext.
4. Output M' as the decryption of C.

When C is a valid encryption of M using P_A and ID_A, it is easy to see that decrypting C will result in an output $M' = M$. This concludes the description of FullCL-PKE.

We note that W in FullCL-PKE can be replaced by $W = E^{sym}_{H_4(\sigma)}(M)$, where E^{sym} is a symmetric encryption algorithm meeting the definition of Section 6.4.2. Also note that our security proofs will require some modifications to handle this case.

6.4 The Fujisaki-Okamoto Hybridisation Technique

Fujisaki and Okamoto [71] provided an elegant conversion from an OWE secure PKE scheme to a IND-CCA secure PKE scheme in the random oracle model. To introduce this conversion we need to define two schemes: a basic PKE scheme and a symmetric encryption scheme.

6.4.1 A Basic PKE Scheme

In order to apply the Fujisaki-Okamoto result, we need to define a PKE scheme with certain properties. Hence, an alternative PKE definition to that specified in Section 3.2.1.1 will be presented next.

We specify a basic PKE scheme Π^{basic} by four algorithms: Setup, Key-Generation, Encrypt and Decrypt, where:

Setup: Similar to the Setup algorithm of the PKE scheme in Section 3.2.1.1. However, the finite message space is now denoted as $\mathcal{M}^{basic}$ and a finite coin space, $COIN^{basic}$, is defined. Both $\mathcal{M}^{basic}$ and $COIN^{basic}$ are defined by the security parameter k.

Key-Generation ($\mathcal{K}$): is identical to the Key-Generation algorithm of the PKE scheme in Section 3.2.1.1.

Encrypt ($\mathcal{E}^{basic}$): is a probabilistic polynomial time algorithm, which takes as input a $x \in \mathcal{M}^{basic}$, params, the public key K_{pub} and a random element $r \in COIN^{basic}$. It returns a ciphertext $y = \mathcal{E}^{basic}_{K_{pub}}(x; r) \in \mathcal{C}$.

Decrypt ($\mathcal{D}^{basic}$): is a deterministic polynomial time algorithm, which takes as input a $y \in \mathcal{C}$, params, and a private key K_{priv}. It returns a message $x = \mathcal{D}^{basic}_{K_{priv}}(y) \in \mathcal{M}^{basic}$.

As we can see, the definition of basic PKE scheme involves a random value r in

the encryption algorithm and the decryption algorithm is a deterministic algorithm which never outputs $\perp$.

6.4.2 A Symmetric Encryption Scheme

We specify a symmetric encryption scheme Π^{sym} by three algorithms: Setup, Encrypt and Decrypt, where:

Setup: is a probabilistic polynomial time algorithm, which takes as input a security parameter k and returns system-wide parameters 'params'. The finite message space is $\mathcal{M}^{sym}$, the finite key space is $KPSC$ and the finite ciphertext space is $\mathcal{C}$. The values of $\mathcal{M}^{sym}$, $KSPC$ and $\mathcal{C}$ are defined by the security parameter k.

Encrypt ($\mathcal{E}^{sym}$): is a deterministic polynomial time algorithm, which takes as input $x \in \mathcal{M}^{sym}$, params and a key $K \in KPSC$. It returns a ciphertext $y = \mathcal{E}_K^{sym}(x) \in \mathcal{C}$.

Decrypt ($\mathcal{D}^{sym}$): is a deterministic polynomial time algorithm, which takes as input $y \in \mathcal{C}$, params and a key $K \in KPSC$. It returns a message $x = \mathcal{D}_K^{sym}(y) \in \mathcal{M}^{sym}$.

This concludes the description of a symmetric encryption scheme. This description will also called upon in Chapter 9.

We introduce the security notion called find-guess for symmetric encryption schemes. In [71] this indistinguishability notion does not allow the adversary access to any encryption oracle. For an alternative security treatment of symmetric encryption schemes see [15].

Find-guess security for symmetric encryption: We say that a symmetric encryption scheme is secure against a find-guess attack if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in the following game:

Setup: The challenger takes a security parameter k as input and runs the Setup algorithm and selects a random key $K \in KPSC$ which it keeps secret. It gives $\mathcal{A}$ a description of the message space $\mathcal{M}^{sym}$.

Find Phase: This is a challenge phase where adversary $\mathcal{A}$ outputs two equal length plaintexts $x_0, x_1 \in \mathcal{M}^{sym}$. The challenger now picks a random bit $b \in \{0,1\}$ and computes $y^* = \mathcal{E}_K^{sym}(x_b)$, the encryption of M_b under the random key $K \in KPSC$. Ciphertext y^* is delivered to $\mathcal{A}$.

Guess: Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. The adversary wins the game if $b = b'$. We define $\mathcal{A}$'s advantage in this game to be $\text{Adv}(\mathcal{A}) := 2|\Pr[b = b'] - \frac{1}{2}|$.

6.4.3 The Fujisaki-Okamoto Hybrid PKE Scheme

We specify the hybrid PKE scheme Π^{hy}, which is constructed from a basic PKE scheme meeting the definition of Section 6.4.1 and a symmetric encryption scheme meeting the definition of Section 6.4.2, by four algorithms: Setup, Key-Generation, Encrypt and Decrypt. We have:

Setup: is similar to the Setup algorithm of the PKE scheme in Section 3.2.1.1. However, the finite message space is now denoted as $\mathcal{M}^{hy}$ and the finite coin space is $COIN^{hy}$. Both $\mathcal{M}^{hy}$ and $COIN^{hy}$ are defined by the security parameter k. Additionally choose cryptographic hash functions $G : \mathcal{M}^{basic} \rightarrow KPSC$ and $H : \mathcal{M}^{basic} \times \mathcal{M}^{sym} \rightarrow COIN^{basic}$.

Key-Generation ($\mathcal{K}$): is identical to the Key-Generation algorithm of the PKE scheme in Section 3.2.1.1.

Encrypt ($\mathcal{E}^{hy}$): is a probabilistic polynomial time algorithm, which takes as input $M \in \mathcal{M}^{hy}$, params, the public key K_{pub} and a random element $\sigma \in \mathcal{M}^{basic}$. It returns a ciphertext $\mathcal{E}^{hy}_{K_{pub}}(M;\sigma) \in \mathcal{C}$.

To encrypt $M \in \mathcal{M}^{hy}$, perform the following steps:

1. Choose a random $\sigma \in \mathcal{M}^{basic}$.
2. Set $r_1 = H(\sigma, M)$.
3. Set $r_2 = G(\sigma)$.
4. Compute and output the ciphertext:

$$\begin{aligned} \mathcal{E}^{hy}_{K_{pub}}(M;\sigma) &= \langle c_1, c_2 \rangle \\ &= \langle \mathcal{E}^{basic}_{K_{pub}}(\sigma; r_1), \mathcal{E}^{sym}_{r_2}(M) \rangle \\ &= \langle \mathcal{E}^{basic}_{K_{pub}}(\sigma; H(\sigma, M)), \mathcal{E}^{sym}_{G(\sigma)}(M) \rangle. \end{aligned}$$

Decrypt ($\mathcal{D}^{hy}$): is a deterministic polynomial time algorithm, which takes as input $\langle c_1, c_2 \rangle \in \mathcal{C}$, params and a private key K_{priv}. It returns a message $M = \mathcal{D}^{hy}_{K_{priv}}(\langle c_1, c_2 \rangle) \in \mathcal{M}^{hy}$.
To decrypt $\langle c_1, c_2 \rangle \in \mathcal{C}$ using private key K_{priv}, do the following:

1. Compute $\mathcal{D}^{basic}_{K_{priv}}(c_1) = \sigma'$.
2. Set $r'_2 = G(\sigma')$ and compute $\mathcal{D}^{sym}_{r'_2}(c_2) = M'$.
3. Set $r'_1 = H(\sigma', M')$ and test if $c_1 = \mathcal{E}^{basic}_{K_{pub}}(\sigma'; r'_1)$. If not, output $\perp$ and reject the ciphertext.
4. Output M' as the decryption of C.

This concludes the description of the hybrid PKE scheme.

Note that in the hybrid PKE scheme definition, the hybrid message space $\mathcal{M}^{hy}$ is equal to $\mathcal{M}^{sym}$. We say that Π^{hy} is the result of applying the Fujisaki-Okamoto hybridisation technique to the schemes Π^{basic} and Π^{sym}.

6.4.4 Security Results

The following result concerning IND-CPA security of the scheme Π^{hy} appears as [71, Lemma 10].

Result 6.1 Suppose that G and H are random oracles and that there exists an IND-CPA adversary $\mathcal{A}$ against Π^{hy} with advantage $\epsilon(k)$ which has running time $t(k)$ and makes at most q_g, q_h queries to G, H respectively. Suppose Π^{basic} is OWE secure against adversaries with running time $t_1(k)$ and advantage $\epsilon_1(k)$ and Π^{sym} is Find-Guess secure against adversaries with running time $t_2(k)$ and advantage $\epsilon_2(k)$ where

$$\begin{aligned} t(k) &= \min(t_1(k), t_2(k)) - \mathcal{O}(l_1 + l_2) \text{ and} \\ \epsilon(k) &= 2(q_g + q_h) \cdot \epsilon_1(k) + \epsilon_2(k). \end{aligned}$$

Here, l_1 and l_2 are the sizes of $\mathcal{M}^{basic}$ and $\mathcal{M}^{sym}$ respectively.

Before exploring any more security results, we define a property of PKE schemes called γ-uniformity [70, 71].

Definition 6.1 Let $\mathcal{E}$ be the encryption algorithm of a PKE scheme meeting the definition of Section 6.4.1. For a given $x \in \mathcal{M}$ and $y \in \{0,1\}^*$, define

$$\gamma(x, y) = \Pr[r \leftarrow \text{random element in } COIN : y = \mathcal{E}_{K_{pub}}(x; r)].$$

We say that the PKE scheme is γ-uniform (in k) if for any $\langle K_{pub}, K_{priv} \rangle$, any $x \in \mathcal{M}$ and any $y \in \{0,1\}^*$, $\gamma(x, y) \leq \gamma$.

The scheme Π^{hy} was shown in [71] to be an IND-CCA secure PKE scheme. A key reason for this is that the scheme Π^{hy} is *plaintext aware*, which implies IND-CCA security in the random oracle model [16]. Plaintext awareness is a property that allows the adversary to output a ciphtertext only if it actually *knows* the corresponding plaintext. Hence, the intuition is that plaintext awareness ensures that the

adversary gains nothing from querying the decryption oracle. This idea was introduced by Bellare and Rogaway [23] and further refined and formalised by Bellare, Desai, Pointcheval and Rogaway [16]. The IND-CCA security result makes use of a special purpose algorithm called a knowledge extractor. This algorithm handles all decryption queries and, with a high probability, outputs the correct decryption of ciphtertexts.

Next we consider an important special case of the Fujisaki-Okamoto construction in Section 6.4.3, in which the symmetric encryption algorithm is replaced by a one-time pad, that is, $\mathcal{E}_K^{sym}(x) = K \oplus x$ and $\mathcal{D}_K^{sym}(y) = K \oplus y$. We let $\Pi^{hy\star}$ denote this hybrid scheme. If we define $\mathcal{M}^{sym}$ and $KSPC$ to be $\{0,1\}^{l_2}$, then the cryptographic hash functions are $G : \mathcal{M}^{basic} \rightarrow \{0,1\}^{l_2}$ and $H : \mathcal{M}^{basic} \times \{0,1\}^{l_2} \rightarrow COIN^{basic}$. The hybrid encryption of plaintext M becomes:

$$\mathcal{E}_{K_{pub}}^{hy\star}(M) = \langle \mathcal{E}_{K_{pub}}^{basic}(\sigma; H(\sigma, M)), G(\sigma) \oplus M \rangle. \tag{6.1}$$

In this setting the following lemma applies.

Lemma 6.2 Suppose that G and H are random oracles and that there exists an IND-CPA adversary $\mathcal{A}$ against $\Pi^{hy\star}$ with advantage $\epsilon(k)$ which has running time $t(k)$ and makes at most q_g, q_h queries to G, H respectively. Then there is an OWE adversary $\mathcal{B}$ against Π^{basic} with running time $t_1(k)$ and advantage $\epsilon_1(k)$ where

$$\begin{aligned} t(k) &= t_1(k) - \mathcal{O}(l_1 + l_2) \text{ and} \\ \epsilon(k) &= 2(q_g + q_h) \cdot \epsilon_1(k). \end{aligned}$$

Here, l_1 and l_2 are the sizes of $\mathcal{M}^{basic}$ and $\mathcal{M}^{sym}$, the asymmetric and symmetric (that is, one-time pad) message space respectively.

Proof. The result follows by specialising Result 6.1 to the setting of $\Pi^{hy\star}$. In $\Pi^{hy\star}$, the symmetric encryption scheme is replaced with a one-time pad. Since the key K for this one-time pad is chosen uniformly at random and used only once in the find-guess game of Section 6.4.2, a find-guess adversary gains no advantage and we have $\epsilon_2(k) = 0$. The lemma follows.

□

Now we are in a position to state a result concerning the IND-CCA security of $\Pi^{hy\star}$. The result appears as [71, Theorem 14].

Result 6.3 Suppose $\Pi^{hy\star}$ is constructed from a γ-uniform PKE scheme Π^{basic} and the one-time pad. Suppose that G and H are random oracles and that there exists an IND-CCA adversary $\mathcal{A}$ against $\Pi^{hy\star}$ with advantage $\epsilon(k)$ which has running time $t(k)$ and makes at most q_d decryption queries and at most q_g, q_h queries to G, H respectively. Then there is an OWE adversary $\mathcal{B}$ against Π^{basic} with running time $t_1(k)$ and advantage $\epsilon_1(k)$ where

$$t(k) = t_1(k) - \mathcal{O}((q_g + q_h) \cdot (l_1 + l_2)) \text{ and} \tag{6.2}$$

$$\epsilon(k) = (2(q_g + q_h) \cdot \epsilon_1(k) + 1) \cdot (1 - \gamma - 2^{-l_2})^{-q_d} - 1. \tag{6.3}$$

Here, l_1 and l_2 are the sizes of $\mathcal{M}^{basic}$ and $\mathcal{M}^{sym}$ respectively.

The $\Pi^{hy\star}$ hybridisation construction is used to form the hybrid schemes in Sections 6.5.1 and 6.5.2. As we shall see, Lemma 6.2 and Result 6.3 will be used to prove the security of these schemes.

6.5 Security of the FullCL-PKE Construction

We need the following two PKE schemes, ElG-HybridPub and BF-HybridPub, as they appear in intermediate steps of the security proof for FullCL-PKE. The IND-CCA and IND-CPA adversaries appropriate for PKE schemes were described in Section 3.5.3.

6.5.1 ElG-HybridPub

We define a public key encryption scheme ElG-HybridPub. The scheme is obtained by applying the hybridisation construction described in Section 6.4 to the encryption

scheme EIG-BasicPub of Section 5.5.1.

This scheme is specified by four algorithms: Setup, Key-Generation, Encrypt and Decrypt.

Setup:

1. Run $\mathcal{IG}$ on input k to generate $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ with the usual properties. Choose a generator $P \in \mathbb{G}_1$.
2. Choose cryptographic hash functions $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$, $H_4 : \{0,1\}^n \to \{0,1\}^n$ and $H_5 : \mathbb{G}_1 \to \{0,1\}^n$.

The message and ciphertext spaces for EIG-HybridPub are $\mathcal{M} = \{0,1\}^n$ and $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^{2n}$. The system parameters are params$= \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, H_3, H_4, H_5 \rangle$.

Key-Generation:

1. Choose a random $x \in \mathbb{Z}_q^*$ and set $R = xP$.
2. Set the public key K_{pub} to be $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, H_3, H_4, H_5, R \rangle = \langle \text{params}, R \rangle$ and the private key to be $K_{priv} = x$.

Encrypt: To encrypt $M \in \mathcal{M}$, perform the following steps:

1. Choose a random $\sigma \in \{0,1\}^n$.
2. Set $r = H_3(\sigma, M)$.
3. Compute and output the ciphertext:

$$C = \langle rP, \sigma \oplus H_5(rR), M \oplus H_4(\sigma) \rangle.$$

Decrypt: To decrypt $C = \langle U, V, W \rangle \in \mathcal{C}$ using private key $K_{priv} = x$, do the following:

1. Compute $V \oplus H_5(xU) = \sigma'$.
2. Compute $W \oplus H_4(\sigma') = M'$.
3. Set $r' = H_3(\sigma', M')$ and test if $U = r'P$. If not, output $\perp$ and reject the ciphertext.
4. Output M' as the decryption of C.

This concludes the description of EIG-HybridPub.

6.5.2 BF-HybridPub

The scheme BF-HybridPub is denoted BasicPub^{hy} in [32]. This scheme applies the hybridisation technique which we described in Section 6.4 to the PKE scheme BF-BasicPub of Section 5.5.2.

This scheme is specified by four algorithms: Setup, Key-Generation, Encrypt and Decrypt.

Setup:

1. Run $\mathcal{IG}$ on input k to generate $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ with the usual properties. Choose a generator $P \in \mathbb{G}_1$.
2. Choose a random $s \in \mathbb{Z}_q^*$ and set $P_0 = sP$.
3. Choose cryptographic hash functions $H_2 : \mathbb{G}_2 \to \{0,1\}^n$, $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$ and $H_4 : \{0,1\}^n \to \{0,1\}^n$.

The message and ciphertext spaces for BF-HybridPub are $\mathcal{M} = \{0,1\}^n$ and $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^{2n}$. The system parameters are params$= \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_0, H_2, H_3, H_4 \rangle$.

Key-Generation:

1. Choose a random $Q \in \mathbb{G}_1^*$.
2. Set the public key to be $K_{pub} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_0, H_2, H_3, H_4, Q \rangle = \langle \text{params}, Q \rangle$ and the private key to be $K_{priv} = sQ$.

Encrypt: To encrypt $M \in \mathcal{M}$, perform the following steps:

1. Choose a random $\sigma \in \{0,1\}^n$.
2. Set $r = H_3(\sigma, M)$.
3. Compute and output the ciphertext:

$$C = \langle rP, \sigma \oplus H_2(\hat{e}(Q, P_0)^r), M \oplus H_4(\sigma) \rangle.$$

Decrypt: To decrypt $C = \langle U, V, W \rangle \in \mathcal{C}$ using private key $K_{priv} = sQ$, do the following:

1. Compute $V \oplus H_2(\hat{e}(sQ, U)) = \sigma'$.
2. Compute $W \oplus H_4(\sigma') = M'$.
3. Set $r' = H_3(\sigma', M')$ and test if $U = r'P$. If not, output $\perp$ and reject the ciphertext.
4. Output M' as the decryption of C.

This concludes the description of BF-HybridPub.

6.5.3 Security of EIG-Hybridpub

We will prove that EIG-Hybridpub is IND-CPA and IND-CCA secure in the random oracle model.

Lemma 6.4 Suppose that H_3 and H_4 are random oracles and that there exists an IND-CPA adversary $\mathcal{A}$ against EIG-HybridPub with advantage ϵ which makes at most q_3 and q_4 queries to H_3 and H_4 respectively. Then there is an OWE adversary against EIG-BasicPub with advantage at least $\epsilon/2(q_3 + q_4)$ and which runs in time time$(\mathcal{A}) + \mathcal{O}(n)$. Here $\mathbb{G}_1$ is obtained from the output $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ of $\mathcal{IG}$.

Proof. This is proven by applying Lemma 6.2 to the scheme EIG-HybridPub, setting $l_1 = n$ and $l_2 = n$.

□

The scheme EIG-HybridPub can be shown to be IND-CPA secure in the random oracle model provided the CDHP is hard by composing the reductions in Lemma 6.4 and Lemma 5.1.

Lemma 6.5 Suppose that H_3 and H_4 are random oracles and that there exists an IND-CCA adversary $\mathcal{A}$ against EIG-HybridPub with advantage ϵ which makes at most q_3 and q_4 queries to H_3 and H_4 respectively and at most q_d decryption queries. Then there is an OWE adversary against EIG-BasicPub with advantage at least

$$\frac{(\epsilon + 1)(1 - q^{-1} - 2^{-n})^{q_d} - 1}{2(q_3 + q_4)}$$

and which runs in time time$(\mathcal{A}) + \mathcal{O}(n(q_3 + q_4))$.

Proof. We apply Result 6.3 to EIG-HybridPub, setting $l_1 = n$, $l_2 = n$ and $\gamma = q^{-1}$. We take $\gamma = q^{-1}$, since q is the order of $\mathbb{G}_1$ which determines the number of encryption variants for a given message.

□

The scheme ElG-HybridPub can be shown to be IND-CCA secure in the random oracle model provided the CDHP is hard by composing the reductions in Lemma 6.5 and Lemma 5.1.

6.5.4 Security of BF-Hybridpub

We will prove that BF-Hybridpub is IND-CPA secure in the random oracle model.

Lemma 6.6 Suppose that H_3 and H_4 are random oracles and that there exists an IND-CPA adversary $\mathcal{A}$ against BF-HybridPub with advantage ϵ which makes at most q_3 and q_4 queries to H_3 and H_4 respectively. Then there is an OWE adversary against BF-BasicPub with advantage at least $\epsilon/2(q_3 + q_4)$ and which runs in time time$(\mathcal{A}) + \mathcal{O}(n)$.

Proof. This is proved by applying Lemma 6.2 to the scheme BF-HybridPub, setting $l_1 = n$ and $l_2 = n$.

□

The scheme BF-HybridPub can be shown to be IND-CPA secure in the random oracle model provided the BDHP is hard, by composing the reductions in Lemma 6.6 and Result 5.2.

Result 6.3 is used in [33] to prove that BF-HybridPub is IND-CCA secure in the random oracle model provided the BDHP is hard. This is shown in [33] by combining Result 6.3 and Result 5.2. In [33, Theorem 4.5] the values of l_1 and l_2 are both equal to n (since σ and M are of length n) and γ is correctly set to $1/q$, where q is the size of the groups $\mathbb{G}_1$, $\mathbb{G}_2$. However, in stating their result, Boneh and Franklin set the value 2^{-l_2} in equation 6.3 to be equal to q^{-1}. Because of this, in the work of [33], the message length n needs to grow at least as fast as k in order to obtain security. This assumption is not mentioned anywhere in [33]. In proving the security of FullCL-PKE, we do not need to use this result concerning the IND-CCA security of

BF-HybridPub. Unlike [33], however, we did not specify n as a function of the group size q in Lemma 6.5, concerning the IND-CCA security of ElG-HybridPub.

6.5.5 Security of FullCL-PKE

On our route to proving the main theorem concerning the security of FullCL-PKE we need to prove three lemmas. Lemma 6.7 and Lemma 6.8 are concerned with Type II and Type I adversaries against FullCL-PKE respectively, and Lemma 6.9 handles the decryption queries required to simulate Lemma 6.8.

Lemma 6.7 Suppose that H_1 and H_2 are random oracles and that there exists a Type II IND-CCA adversary $\mathcal{A}_{II}$ against FullCL-PKE with advantage ϵ which makes at most q_1 queries to H_1. Then there is an IND-CCA adversary against ElG-HybridPub with advantage at least ϵ/q_1 which runs in time $\mathcal{O}(time(\mathcal{A}_{II}))$.

Proof. Let $\mathcal{A}_{II}$ be a Type II IND-CCA adversary against FullCL-PKE. Suppose $\mathcal{A}_{II}$ has advantage ϵ and makes q_1 queries to random oracle H_1. We show how to construct from $\mathcal{A}_{II}$ an IND-CCA adversary $\mathcal{B}$ against the PKE scheme ElG-HybridPub.

Let $\mathcal{C}$ denote the challenger against our IND-CCA adversary $\mathcal{B}$ for ElG-HybridPub. The challenger $\mathcal{C}$ begins by supplying $\mathcal{B}$ with a public key

$$K_{pub} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, H_3, H_4, H_5, R \rangle = \langle \mathsf{params}, R \rangle.$$

Adversary $\mathcal{B}$ mounts an IND-CCA attack on the key K_{pub} using help from $\mathcal{A}_{II}$ as follows.

First of all $\mathcal{B}$ chooses an index I with $1 \leq I \leq q_1$. Then $\mathcal{B}$ simulates the algorithm Setup of FullCL-PKE for $\mathcal{A}_{II}$ by choosing a random $s \in \mathbb{Z}_q^*$, setting $P_0 = sP$ and supplying $\mathcal{A}_{II}$ with $\mathsf{params} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_0, H_1, H_2, H_3, H_4, H_5 \rangle$ and the value s. Here, H_1 and H_2 are additional random oracles.

Adversary $\mathcal{A}_{II}$ may make queries of H_1 or H_2 at any time. These are handled as follows:

H_1 **queries:** The H_1 queries are simulated by $\mathcal{B}$. For an ID_i query, $\mathcal{B}$ will choose a random $Q_i \in \mathbb{G}_1^*$ and return $H_1(\mathsf{ID}_i) = Q_i$ for $1 \leq i \leq q_1$. For each i where $i \neq I$, $\mathcal{B}$ chooses a random $x_i \in \mathbb{Z}_q$ and maintains a table with entries $\langle Q_i, x_i \rangle$.

H_2 **queries:** Adversary $\mathcal{B}$ simulates these and answers H_2 queries by maintaining a list of queries and replies. We do need to assume in the course of the proof that H_2 is a random oracle.

Phase 1: Now $\mathcal{A}_{II}$ launches Phase 1 of its attack, by making a series of requests, each of which is either a private key extraction, a request for a public key for a particular entity, or a decryption query. (Recall that a Type II adversary cannot replace public keys and can make partial private key extraction queries for himself given s.) We assume that $\mathcal{A}_{II}$ always makes the appropriate H_1 query on ID before making one of these requests for that identifier. $\mathcal{B}$ replies to these requests as follows:

Private Key Extraction: If the request is on ID_I then $\mathcal{B}$ aborts. Otherwise, if the request is on ID_i with $i \neq I$, then $\mathcal{B}$ outputs $\langle sQ_i, x_i \rangle$.

Request for Public Key: If the request is on ID_I then $\mathcal{B}$ returns R. Otherwise, if the request is on ID_i for some i with $i \neq I$, then $\mathcal{B}$ returns $x_i P$.

Decryption Queries: If the request is to decrypt $\langle U, V, W \rangle$ under the private key for ID_I, then $\mathcal{B}$ computes $\xi = \hat{e}(U, sQ_I)$ and relays the decryption query $\langle U, V \oplus H_2(\xi), W \rangle$ to $\mathcal{C}$. The FullCL-PKE decryption of $\langle U, V, W \rangle$ under the (unknown) private key for ID_I is equal to the EIG-HybridPub decryption of $\langle U, V \oplus H_2(\xi), W \rangle$ under the (unknown) private key corresponding to K_{pub}. Hence $\mathcal{C}$'s response to $\mathcal{B}$'s request can be relayed to $\mathcal{A}_{II}$. On the other hand, if the request is to decrypt $\langle U, V, W \rangle$ under the private key for ID_i ($i \neq I$), then $\mathcal{B}$ can perform this decryption himself using the private key $\langle sQ_i, x_i \rangle$ for ID_i.

Challenge Phase: At some point, $\mathcal{A}_{II}$ decides to end Phase 1 and picks ID_{ch} and two messages M_0, M_1 on which it wants to be challenged. We can assume that $\mathcal{A}_{II}$ has not extracted the private key for this identifier. Algorithm $\mathcal{B}$ responds as follows. If $\mathsf{ID}_{\text{ch}} \neq \mathsf{ID}_I$ then $\mathcal{B}$ aborts. Otherwise $\mathsf{ID}_{\text{ch}} = \mathsf{ID}_I$ and $\mathcal{B}$ gives $\mathcal{C}$ the pair M_0, M_1 as the messages on which it wishes to be challenged. $\mathcal{C}$ responds with the challenge ciphertext $C' = \langle U', V', W' \rangle$, such that C' is the EIG-HybridPub encryption of M_b under K_{pub} for a random $b \in \{0, 1\}$. Then $\mathcal{B}$ computes $\xi' = \hat{e}(U', sQ_I)$ and sets $C^* = \langle U', V' \oplus H_2(\xi'), W' \rangle$ and delivers C^* to $\mathcal{A}_{II}$. It is not hard to see that C^* is the FullCL-PKE encryption of M_b for identifier ID_I (with public key R).

Phase 2: Adversary $\mathcal{B}$ continues to respond to requests in the same way as it did in Phase 1. Of course, we now restrict $\mathcal{A}_{II}$ to not make private key extraction requests on ID_{ch}. If any decryption query relayed to $\mathcal{C}$ is equal to the challenge ciphertext C' then $\mathcal{B}$ aborts.

Guess: Eventually, $\mathcal{A}_{II}$ will make a guess b' for b. $\mathcal{B}$ outputs b' as its guess for b.

Analysis: Now we analyze the behavior of $\mathcal{B}$ and $\mathcal{A}_{II}$ in this simulation. We claim that if algorithm $\mathcal{B}$ does not abort during the simulation then algorithm $\mathcal{A}_{II}$'s view is identical to its view in the real attack. Moreover, if $\mathcal{B}$ does not abort then $2|\Pr[b = b'] - \frac{1}{2}| \geq \epsilon$.

We justify this claim as follows. $\mathcal{B}$'s responses to H_1 and H_2 queries are uniformly and independently distributed in $\mathbb{G}_1^*$ and $\{0, 1\}^n$ respectively, as in the real attack. All responses to $\mathcal{A}_{II}$'s requests are valid, provided of course that $\mathcal{B}$ does not abort. Furthermore, the challenge ciphertext C^* is a valid FullCL-PKE encryption of M_b where $b \in \{0, 1\}$ is random. Thus, by definition of algorithm $\mathcal{A}_{II}$ we have that $2|\Pr[b = b'] - \frac{1}{2}| \geq \epsilon$.

The probability that $\mathcal{B}$ does not abort during the simulation remains to be calculated. Examining the simulation, we see that $\mathcal{B}$ can abort for three reasons: (i) because $\mathcal{A}_{II}$ made a private key extraction on ID_I at some point, (ii) because $\mathcal{A}_{II}$ did not choose

$\mathsf{ID}_{\text{ch}} = \mathsf{ID}_I$, or (iii) because $\mathcal{B}$ relayed a decryption query on $C' = \langle U', V', W' \rangle$ to $\mathcal{C}$ in Phase 2.

Because of the way that $\mathcal{B}$ converts ciphertexts, this last event happens only if $\mathcal{A}_{II}$ queries $\mathcal{B}$ on the ciphertext $C^* = \langle U', V' \oplus H_2(\xi'), W' \rangle$ in Phase 2. However, this is exactly $\mathcal{A}_{II}$'s challenge ciphertext on which $\mathcal{A}_{II}$ is forbidden from making a decryption query, since

$$\langle U', V' \oplus H_2(\xi') \oplus H_2(\hat{e}(U', sQ_I)), W' \rangle = \langle U', V', W' \rangle.$$

So this event never occurs in $\mathcal{B}$'s simulation. We name the remaining events that can cause $\mathcal{B}$ to abort as $\mathcal{Q}_1$ and $\mathcal{Q}_2$.

Notice that the event $\neg\mathcal{Q}_2$ implies the event $\neg\mathcal{Q}_1$ (if $\mathcal{A}_{II}$ chooses ID_{ch} equal to ID_I, then no private key extraction on ID_I is allowed). Hence we have

$$\begin{aligned} \Pr[\mathcal{B} \text{ does not abort}] &= \Pr[\neg\mathcal{Q}_1 \wedge \neg\mathcal{Q}_2] \\ &= \Pr[\neg\mathcal{Q}_2] \\ &= 1/q_1 \end{aligned}$$

where the last equality follows from $\mathcal{B}$'s random choice of I being independent of $\mathcal{A}_{II}$'s choice of ID_{ch}.

Thus we see that $\mathcal{B}$'s advantage is at least ϵ/q_1 and the proof is complete.

□

Lemma 6.8 Suppose that H_i $(1 \leq i \leq 5)$ are random oracles and that there exists a Type I IND-CCA adversary $\mathcal{A}_I$ against FullCL-PKE. Suppose $\mathcal{A}_I$ has advantage ϵ, runs in time t, makes at most q_i queries to H_i $(1 \leq i \leq 5)$ and makes at most q_d decryption queries. Then there is an algorithm $\mathcal{B}$ which acts as either a BF-HybridPub or an ElG-HybridPub IND-CPA adversary. Moreover, $\mathcal{B}$ either has advantage at least $\epsilon\lambda^{q_d}/4q_1$ when playing as a BF-HybridPub adversary, or has advantage at least $\epsilon/4q_1$ when playing as an ElG-HybridPub adversary. Algorithm $\mathcal{B}$ runs in time $t + \mathcal{O}((q_3 + q_4)q_d t')$. Here t' is the running time of the BasicCL-PKE encryption algorithm and

$$\lambda \geq 1 - (q_3 + q_4) \cdot \epsilon_{\text{OWE}}(t + \mathcal{O}((q_3 + q_4)q_d t', q_2, q_5)) - 4q^{-1} - 2^{-n+2}.$$

where $\epsilon_{\mathrm{OWE}}(T, q', q'')$ denotes the highest advantage of any OWE adversary against BasicCL-PKE which operates in time T and makes q' hash queries to H_2 and q'' hash queries to H_5.

Proof. Let $\mathcal{A}_I$ be a Type I IND-CCA adversary against FullCL-PKE. Suppose $\mathcal{A}_I$ has advantage ϵ, runs in time t, makes q_i queries to random oracle H_i $(1 \leq i \leq 5)$ and makes q_d decryption queries. We show how to construct from $\mathcal{A}_I$ an adversary $\mathcal{B}$ that acts either as an IND-CPA adversary against the PKE scheme BF-HybridPub or as an IND-CPA adversary against the PKE scheme ElG-HybridPub. We assume that challengers $\mathcal{C}_I$ and $\mathcal{C}_{II}$ for both types of games are available to $\mathcal{B}$.

Adversary $\mathcal{B}$ begins by choosing a random bit c and an index I uniformly at random with $1 \leq I \leq q_1$. If $c = 0$, then $\mathcal{B}$ chooses to play against $\mathcal{C}_I$ and aborts $\mathcal{C}_{II}$. Here, $\mathcal{B}$ will build an IND-CPA adversary against BF-HybridPub and fail against $\mathcal{C}_{II}$. When $c = 1$, $\mathcal{B}$ chooses to play against $\mathcal{C}_{II}$ and aborts $\mathcal{C}_I$. Here, $\mathcal{B}$ will build a IND-CPA adversary against ElG-HybridPub and fail against $\mathcal{C}_I$. In either case, $\mathcal{C}$ will denote the challenger against which $\mathcal{B}$ plays for the remainder of this proof.

As in Lemma 5.4, we define three events $\mathcal{H}$, $\mathcal{F}_0$ and $\mathcal{F}_1$:

- $\mathcal{H}$: Adversary $\mathcal{A}_I$ chooses ID_I as the challenge identifier $\mathsf{ID}_{\mathrm{ch}}$.
- $\mathcal{F}_0$: Adversary $\mathcal{A}_I$ extracts the partial private key for entity ID_I.
- $\mathcal{F}_1$: Adversary $\mathcal{A}_I$ replaces the public key of entity ID_I at some point in its attack.

The general strategy of the proof is similar to that of the proof of Lemma 5.4. If $(c = 0) \wedge \mathcal{F}_0$ occurs, $\mathcal{B}$ will have to abort and will be unsuccessful. If $\neg\mathcal{F}_0 \wedge \mathcal{H}$ occurs, then $\mathcal{B}$'s success probability will be related to that of $\mathcal{A}_I$. On the other hand, if $(c = 1) \wedge \mathcal{F}_1$ occurs, $\mathcal{B}$ will again have to abort and will be unsuccessful. If $\neg\mathcal{F}_1 \wedge \mathcal{H}$ occurs, then $\mathcal{B}$'s success probability will again be related to that of $\mathcal{A}_I$. Overall, we will show that $\mathcal{B}$'s advantage in its mixed-game strategy is non-negligible if $\mathcal{A}_I$'s is.

It is then easy to see that $\mathcal{B}$ has a non-negligible advantage for at least one of the two game types.

If $c = 0$, then $\mathcal{C}$ is an IND-CPA challenger for BF-HybridPub and begins by supplying $\mathcal{B}$ with a public key $K_{pub} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_0, H_2, H_3, H_4, Q \rangle$. If $c = 1$, then $\mathcal{C}$ is an IND-CPA challenger for EIG-HybridPub and so supplies $\mathcal{B}$ with a public key $K_{pub} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, H_3, H_4, H_5, R \rangle$.

Then $\mathcal{B}$ simulates the algorithm Setup of FullCL-PKE for $\mathcal{A}_I$. When $c = 0$, $\mathcal{B}$ will handle H_5 queries, while when $c = 1$, $\mathcal{B}$ will handle H_2 queries. Additionally, when $c = 1$, $\mathcal{B}$ chooses a random $s \in \mathbb{Z}_q^*$ and sets $P_0 = sP$. Thus, $\mathcal{B}$ supplies $\mathcal{A}_I$ with params$= \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_0, H_1, H_2, H_3, H_4, H_5 \rangle$. Here H_1 is a random oracle that will be controlled by $\mathcal{B}$.

Adversary $\mathcal{A}_I$ may make queries of the random oracles H_i, $1 \leq i \leq 5$, at any time during its attack. These are handled as follows:

H_1 **queries:** Adversary $\mathcal{B}$ maintains a list of tuples $\langle \mathsf{ID}_i, Q_i, b_i, x_i, P_i \rangle$ which we call the H_1 list. The list is initially empty, and when $\mathcal{A}_I$ queries H_1 on input $\mathsf{ID} \in \{0,1\}^*$, $\mathcal{B}$ responds as follows:

1. If ID already appears on the H_1 list in a tuple $\langle \mathsf{ID}_i, Q_i, b_i, x_i, P_i \rangle$, then $\mathcal{B}$ responds with $H_1(\mathsf{ID}) = Q_i$.

2. Suppose ID does not already appear on the list and ID is the I-th distinct H_1 query made by $\mathcal{A}_I$. For $c = 0$, $\mathcal{B}$ outputs $H_1(\mathsf{ID}) = Q$, selects a random $x_I \in \mathbb{Z}_q^*$ and adds the entry $\langle \mathsf{ID}, Q, \perp, x_I, x_I P \rangle$ to the H_1 list. For $c = 1$, $\mathcal{B}$ selects $b_I \in \mathbb{Z}_q^*$, outputs $H_1(\mathsf{ID}) = b_I P$ and adds the entry $\langle \mathsf{ID}, b_I P, b_I, \perp, R \rangle$ to the H_1 list.

3. Otherwise, when ID does not already appear on the list and ID is the i-th distinct H_1 query made by $\mathcal{A}_I$ where $i \neq I$, $\mathcal{B}$ picks random $x_i, b_i \in \mathbb{Z}_q^*$, sets $Q_i = b_i P$, outputs $H_1(\mathsf{ID}) = Q_i$ and adds $\langle \mathsf{ID}, b_i P, b_i, x_i, x_i P \rangle$ to the H_1 list.

Notice that with this specification of H_1, the FullCL-PKE partial private key for ID_i ($i \neq I$) is equal to $b_i P_0$ while the public key for ID_i ($i \neq I$) is $P_i = x_i P$ and the private key for ID_i ($i \neq I$) is $\langle b_i P_0, x_i \rangle$. These can all be computed by $\mathcal{B}$. When $c = 1$, $\mathcal{B}$ sets the public key of ID_I to be R and can compute the partial private key of ID_I as $sb_I P$. When $c = 0$, $\mathcal{B}$ knows neither the partial private key nor the private key for ID_I.

H_2 queries: When $c = 0$ any H_2 queries made by $\mathcal{A}_I$ are passed to $\mathcal{C}$ to answer. When $c = 1$ any H_2 queries made by $\mathcal{A}_I$ are simulated by $\mathcal{B}$ using the standard approach of maintaining a list of queries and replies. We do need to assume in the course of the proof that H_2 is a random oracle.

H_3 and H_4 queries: Adversary $\mathcal{B}$ passes $\mathcal{A}_I$'s H_3 and H_4 queries to $\mathcal{C}$ to answer, but keeps lists $\langle \sigma_j, M_j, H_{3,j} \rangle$ and $\langle \sigma'_i, H_{4,i} \rangle$ of $\mathcal{A}_I$'s distinct queries and $\mathcal{C}$'s replies to them.

H_5 queries: Any H_5 queries made by $\mathcal{A}_I$ are passed to $\mathcal{C}$ to answer when $c = 1$. When $c = 0$, $\mathcal{B}$ maintains a list of tuples $\langle \mu_i, H_{5,i} \rangle$ which we call the H_5 list. The list is initially empty, and when $\mathcal{A}_I$ queries H_5 on input $\mu \in \mathbb{G}_1$, $\mathcal{B}$ responds as follows:

1. If μ already appears on the H_5 list in a tuple $\langle \mu_i, H_{5,i} \rangle$, then $\mathcal{B}$ responds with $H_5(\mu) = H_{5,i}$.

2. Suppose μ does not already appear on the list. If the H_5 query is made before the challenge phase, then $\mathcal{B}$ goes to step 3 below. Otherwise, let P_{ch} denote the value of the public key for the challenge identifier ID_{ch} during the challenge phase, let $C^* = \langle U^*, V^*, W^* \rangle$ be the challenge ciphertext delivered to $\mathcal{A}_I$ by $\mathcal{B}$, and let ξ be the value, to be defined below, used by $\mathcal{B}$ in the challenge phase. $\mathcal{B}$ tests if μ satisfies $\hat{e}(\mu, P) = \hat{e}(U^*, P_{\text{ch}})$. If equality holds, then $\mathcal{B}$ adds $\langle \mu, \xi \rangle$ to the H_5 list and outputs $\xi = H_5(\mu)$. If the equality does not hold, then $\mathcal{B}$ goes to step 3.

3. Supposing μ to be the i-th distinct H_5 query made by $\mathcal{A}_I$, $\mathcal{B}$ selects a random

$H_{5,i} \in \{0,1\}^n$, outputs $H_5(\mu) = H_{5,i}$ and adds $\langle \mu_i, H_{5,i} \rangle$ to the H_5 list.

We simulate H_5 this way for the same reasons as those discussed in the proof of Lemma 5.4.

Phase 1: After receiving params from $\mathcal{B}$, $\mathcal{A}_I$ launches Phase 1 of its attack, by making a series of requests, each of which is either a partial private key extraction for an entity, a private key extraction for an entity, a request for a public key for an entity, a replacement of a public key for an entity or a decryption query for an entity. We assume that $\mathcal{A}_I$ always makes the appropriate H_1 query on the identifier ID for that entity before making one of these requests. $\mathcal{B}$ replies to these requests as follows:

Partial Private Key Extraction: Suppose the request is on ID_i. There are three cases:

1. If $i \neq I$, then $\mathcal{B}$ replies with $b_i P_0$.
2. If $i = I$ and $c = 1$, then $\mathcal{B}$ replies with $b_I P_0$.
3. If $i = I$ and $c = 0$, then $\mathcal{B}$ aborts.

Private Key Extraction: Suppose the request is on ID_i. We can assume that the public key for ID_i has not been replaced. There are two cases:

1. If $i \neq I$, then $\mathcal{B}$ outputs $\langle b_i P_0, x_i \rangle$.
2. If $i = I$, then $\mathcal{B}$ aborts.

Request for Public Key: If the request is on ID_i then $\mathcal{B}$ returns P_i by accessing the H_1 list.

Replace Public Key: Suppose the request is to replace the public key for ID_i with value P_i'. There are three cases:

1. If $i = I$ and $c = 1$, then $\mathcal{B}$ aborts.
2. If $i = I$ and $c = 0$, then $\mathcal{B}$ replaces the current entry in the H_1 list with the new entry P_I' and updates the tuple to $\langle \mathsf{ID}_I, Q, \perp, \perp, P_I' \rangle$.
3. Otherwise, $\mathcal{B}$ replaces the current entry in the H_1 list with the new entry P_i' $(i \neq I)$ and updates the tuple to $\langle \mathsf{ID}_i, b_i P, b_i, \perp, P_i' \rangle$.

Decryption Queries: Suppose the request is to decrypt ciphertext $\langle U, V, W \rangle$ for ID_ℓ, where the private key that should be used is the one corresponding to the current value of the public key for ID_i. Notice that even when $\ell = I$, $\mathcal{B}$ cannot make use of $\mathcal{C}$ to answer the query, because $\mathcal{B}$ is meant to be an IND-CPA adversary. Instead $\mathcal{B}$ makes use of an algorithm $\mathcal{KE}$ to perform all the decryptions. This algorithm, essentially a knowledge extractor in the sense of [16, 71], is not perfect, but as we shall show below, the probability that it decrypts incorrectly is sufficiently low that it can be used in place of a true decryption algorithm making use of private keys. Algorithm $\mathcal{KE}$ is defined as follows:

Algorithm $\mathcal{KE}$: The input to the algorithm is a ciphertext $C = \langle U, V, W \rangle$, an identifier ID_ℓ and the current value of the public key P_ℓ. We assume that $\mathcal{KE}$ also has access to the H_3 and H_4 lists. Algorithm $\mathcal{KE}$ operates as follows:

1. Find all triples $\langle \sigma_j, M_j, H_{3,j} \rangle$ on the H_3 list such that

$$\langle U, V \rangle = \mathsf{BasicCL\text{-}PKE\text{-}Encrypt}_{\mathsf{ID}_\ell, P_\ell}(\sigma_j; H_{3,j}).$$

   Here, $\mathsf{BasicCL\text{-}PKE\text{-}Encrypt}_{\mathsf{ID}_A, P_A}(M; r)$ denotes the BasicCL-PKE encryption of message M for ID_A using public key P_A and random value r. Collect all these triples in a list S_1. If S_1 is empty, output $\perp$ and halt.

2. For each triple $\langle \sigma_j, M_j, H_{3,j} \rangle$ in S_1, find all pairs $\langle \sigma'_i, H_{4,i} \rangle$ in the H_4 list with $\sigma_j = \sigma'_i$. For each such match, place $\langle \sigma_j, M_j, H_{3,j}, H_{4,i} \rangle$ on a list S_2. If S_2 is empty, then output $\perp$ and halt.

3. Check in S_2 for an entry such that $W = M_j \oplus H_{4,i}$. If such an entry exists, then output M_j as the decryption of $\langle U, V, W \rangle$. Otherwise, output $\perp$.

Lemma 6.9 shows that $\mathcal{KE}$ correctly decrypts with high probability.

Challenge Phase: At some point, $\mathcal{A}_I$ should decide to end Phase 1 and pick ID_{ch} and two messages m_0, m_1 on which it wishes to be challenged. We can assume that ID_{ch} has already been queried of H_1 but that $\mathcal{A}_I$ has not extracted the private key for this identifier. Algorithm $\mathcal{B}$ responds as follows. If $\mathsf{ID}_{\text{ch}} \neq \mathsf{ID}_I$ then $\mathcal{B}$ aborts. Otherwise $\mathsf{ID}_{\text{ch}} = \mathsf{ID}_I$ and $\mathcal{B}$ gives $\mathcal{C}$ the pair m_0, m_1 as the messages on which it wishes to be challenged. There are now two cases:

- When $c = 0$, $\mathcal{C}$ responds with the challenge ciphertext $C' = \langle U', V', W' \rangle$, a BF-HybridPub encryption of m_b under K_{pub} for a random $b \in \{0, 1\}$. Now $\mathcal{B}$ checks each entry $\langle \mu_i, H_{5,i} \rangle$ in the H_5 list to see if it satisfies the equality $\hat{e}(\mu_i, P) = \hat{e}(U', P_{\text{ch}})$. It is easy to see that at most one entry can do so. If $\mathcal{B}$ finds that the j-th entry satisfies the equality, then $\mathcal{B}$ sets $C^* = \langle U', V' \oplus H_{5,j}, W' \rangle$ and delivers C^* to $\mathcal{A}_I$ as the challenge ciphertext. Otherwise, if no entry satisfies this test, $\mathcal{B}$ selects a random $\xi \in \{0, 1\}^n$, sets $C^* = \langle U', V' \oplus \xi, W' \rangle$ and delivers C^* to $\mathcal{A}_I$.

- When $c = 1$, $\mathcal{C}$ responds with the challenge ciphertext $C' = \langle U', V', W' \rangle$, such that C' is the EIG-HybridPub encryption of m_b under K_{pub} for a random $b \in \{0, 1\}$. Then $\mathcal{B}$ sets $C^* = \langle U', V' \oplus H_2(\hat{e}(U', b_I sP)), W' \rangle$ and delivers C^* to $\mathcal{A}_I$.

It is easy to see that in both cases C^* is the FullCL-PKE encryption of m_b for identifier ID_{ch} under public key P_{ch}. We now let P_{ch} denote the particular value of the public

key for identifier ID_{ch} during the challenge phase ($\mathcal{A}_I$ may change this value in Phase 2 of its attack).

Phase 2: Adversary $\mathcal{B}$ continues to respond to $\mathcal{A}_I$'s requests in the same way as it did in Phase 1. However, the same restrictions as identified in Section 6.2 on $\mathcal{A}_I$'s behaviour apply in this phase.

Guess: Eventually, $\mathcal{A}_I$ should make a guess b' for b. Then $\mathcal{B}$ outputs b' as its guess for b. If $\mathcal{A}_I$ has used more than time t, or attempts to make more than q_i queries to random oracle H_i or more than q_d decryption queries, then $\mathcal{B}$ should abort $\mathcal{A}_I$ and output a random guess for bit b (in this case algorithm $\mathcal{KE}$ has failed to perform correctly at some point).

Analysis: Now we analyze the behavior of $\mathcal{B}$ and $\mathcal{A}_I$ in this simulation. We claim that if algorithm $\mathcal{B}$ does not abort during the simulation and if all of $\mathcal{B}$'s uses of the algorithm $\mathcal{KE}$ result in correct decryptions, then algorithm $\mathcal{A}_I$'s view is identical to its view in the real attack. Moreover, if this is the case, then $2|\Pr[b = b'] - \frac{1}{2}| \geq \epsilon$. This is not hard to see: Adversary $\mathcal{B}$'s responses to all hash queries are uniformly and independently distributed as in the real attack. All responses to $\mathcal{A}_I$'s requests are valid, provided of course that $\mathcal{B}$ does not abort and that $\mathcal{KE}$ performs correctly. Furthermore, the challenge ciphertext C^* is a valid FullCL-PKE encryption of m_b under the current public key for identifier ID_{ch}, where $b \in \{0, 1\}$ is random. Thus, by definition of algorithm $\mathcal{A}_I$ we have that $2|\Pr[b = b'] - \frac{1}{2}| \geq \epsilon$.

So we must examine the probability that $\mathcal{B}$ does not abort during the simulation given that the algorithm $\mathcal{KE}$ performs correctly. Examining the simulation, we see that $\mathcal{B}$ can abort for the same four reasons as in Lemma 5.4, that is:

0. Because $c = 0$ and the event $\mathcal{F}_0$ occurred during the simulation.
1. Because $c = 1$ and event $\mathcal{F}_1$ occurred during the simulation.
2. Because $\mathcal{A}_I$ made a private key extraction on ID_I at some point.

3. Or because $\mathcal{A}_I$ chose $\mathsf{ID}_{\mathrm{ch}} \neq \mathsf{ID}_I$.

We can use a proof identical to that of Lemma 5.4 to establish a lower bound on the probability that $\mathcal{B}$ does not abort. We obtain:

$$\Pr[\mathcal{B} \text{ does not abort}] \geq \frac{1}{2q_1}.$$

Now we examine the probability that algorithm $\mathcal{KE}$ correctly handles all of $\mathcal{A}_I$'s q_d decryption queries. We will show in Lemma 6.9 below that the probability that $\mathcal{KE}$ correctly replies to individual decryption queries is at least λ, where λ is bounded as in the statement of that lemma.

It is now easy to see that $\mathcal{B}$'s advantage is at least $\frac{\epsilon}{2q_1}\lambda^{q_d}$. It follows that either $\mathcal{B}$'s advantage as an adversary against BF-HybridPub or $\mathcal{B}$'s advantage as an adversary against EIG-HybridPub is at least $\frac{\epsilon}{4q_1}\lambda^{q_d}$. The running time of $\mathcal{B}$ is $\mathrm{time}(\mathcal{A}_I) + q_d \cdot \mathrm{time}(\mathcal{KE}) = t + O((q_3 + q_4)q_d t')$ where t' is the running time of the BasicCL-PKE encryption algorithm. This completes the proof of the lemma.

□

Lemma 6.9 In the simulation in the proof of Lemma 6.8, Algorithm $\mathcal{KE}$ correctly replies to individual decryption queries with probability at least λ where

$$\lambda \geq 1 - (q_3 + q_4) \cdot \epsilon_{\mathrm{OWE}}(t + \mathcal{O}((q_3 + q_4)q_d t', q_2, q_5)) - 4q^{-1} - 2^{-n+2}.$$

Here t is the running time of adversary $\mathcal{A}_I$, t' is the running time of the BasicCL-PKE encryption algorithm, and $\epsilon_{\mathrm{OWE}}(T, q', q'')$ denotes the highest advantage of any Type I OWE adversary against BasicCL-PKE which operates in time T and makes q' hash queries to H_2 and q'' hash queries to H_5.

Proof. We recall that queries to $\mathcal{KE}$ come in the form of a ciphertext $C = \langle U, V, W \rangle$, an identifier ID_ℓ and the current value of the public key P_ℓ for that identifier. We also assume that $\mathcal{KE}$ has access to the H_3 and H_4 lists as they stand at the point where the decryption query is made. We model the fact that $\mathcal{A}_I$ obtains a challenge

ciphertext by considering an additional list of ciphertexts $\mathcal{Y}$ in our proof. This list is empty until the challenge phase and thereafter consists of just the challenge ciphertext $C^* = \langle U^*, V^*, W^* \rangle$.

We define a sequence of events:

- Inv is the event that there exists some $C' = \langle U', V', W' \rangle \in \mathcal{Y}$ and some $\langle \sigma_j, M_j, H_{3,j} \rangle$ on the H_3 list or some $\langle \sigma'_i, H_{4,i} \rangle$ on the H_4 list such that the BasicCL-PKE decryption of $\langle U', V' \rangle$ under the private key corresponding to P_{ch} and ID_{ch} is equal to σ_j or σ'_i. (For us, Inv has zero probability until after a non-abortive challenge phase in $\mathcal{A}_I$'s attack because $\mathcal{Y}$ is empty up to this point.)
- $\mathcal{L}_1$ is the event that S_1 is non-empty.
- $\mathcal{L}_2$ is the event that S_2 is non-empty.
- Find is the event that there exists an entry $\langle \sigma_j, M_j, H_{3,j}, H_{4,i} \rangle$ in S_2 such that $W = M_j \oplus H_{4,i}$.
- Fail is the event that the output of algorithm $\mathcal{KE}$ is not the decryption of C under the private key corresponding to identifier ID_ℓ and public key P_ℓ.

We want to bound the probability of the event Fail for a particular execution of algorithm $\mathcal{KE}$. To do so, we follow the proof of [71, Lemma 11] to obtain:

$$\begin{aligned} \Pr[\mathsf{Fail}] \leq\ & \Pr[\mathsf{Inv}] + \Pr[\mathsf{Fail} | \neg\mathsf{Inv} \wedge \neg\mathcal{L}_1] \\ & + \Pr[\mathsf{Fail} | \neg\mathsf{Inv} \wedge \mathcal{L}_1 \wedge \neg\mathcal{L}_2] \\ & + \Pr[\mathsf{Fail} | \neg\mathsf{Inv} \wedge \mathcal{L}_1 \wedge \mathcal{L}_2 \wedge \neg\mathsf{Find}] \\ & + \Pr[\mathsf{Fail} | \neg\mathsf{Inv} \wedge \mathcal{L}_1 \wedge \mathcal{L}_2 \wedge \mathsf{Find}]. \end{aligned}$$

We proceed to bound each of the terms in the above inequality.

Claim: $\Pr[\mathsf{Inv}] \leq (q_3 + q_4) \cdot \epsilon_{\text{OWE}}(\text{time}(\mathcal{B}), q_2, q_5)$.

Proof. Here $\epsilon_{\text{OWE}}(T, q', q'')$ denotes the highest advantage of any Type I OWE adversary against BasicCL-PKE which operates in time T and makes q' hash queries to

H_2 and q'' hash queries to H_5, while time($\mathcal{B}$) denotes the running time of adversary $\mathcal{B}$ in the proof of Lemma 6.8.

We sketch how to construct an OWE adversary $\mathcal{B}'$ against BasicCL-PKE by adapting adversary $\mathcal{B}$ in the proof of Lemma 6.8. Our adversary $\mathcal{B}'$ will have a chance of being successful provided that the event Inv occurs in the course of $\mathcal{A}_I$'s attack.

Adversary $\mathcal{B}'$ begins by choosing a random bit c and, as with $\mathcal{B}$, it selects an index I uniformly at random with $1 \leq I \leq q_1$. Let q_1, q_3 and q_4 denote the number of H_1, H_3 and H_4 queries made in $\mathcal{A}_I$'s attack respectively.

The running time of the adversary will be the same as that of $\mathcal{B}$. The existence of this adversary will be used to bound the probability of the event Inv.

In fact $\mathcal{B}'$ is closely related to $\mathcal{B}$. $\mathcal{B}'$ is given by its challenger $\mathcal{C}'$ the system parameters of BasicCL-PKE which are

$$\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_0, H_1, H_2, H_5 \rangle$$

(and the value s when $c = 1$). $\mathcal{B}'$ now passes $\mathcal{A}_I$'s H_1, H_2, H_3, H_4 and H_5 queries to $\mathcal{C}'$ to answer, but keeps lists of all distinct queries made by $\mathcal{A}_I$ and $\mathcal{C}'$'s replies to them. $\mathcal{B}'$ answers the following requests as follows:

- Partial private key extraction queries: If $i = I$ and $c = 0$, then $\mathcal{B}'$ aborts. Otherwise, $\mathcal{B}'$ passes $\mathcal{A}_I$'s queries to $\mathcal{C}'$ to answer.
- Private key extraction queries: If $i = I$ then $\mathcal{B}'$ aborts. Otherwise, $\mathcal{B}'$ passes $\mathcal{A}_I$'s queries to $\mathcal{C}'$ to answer.
- Request for public key queries: $\mathcal{B}'$ passes $\mathcal{A}_I$'s queries to $\mathcal{C}'$ to answer.
- Replace public key queries: If $i = I$ and $c = 1$, then $\mathcal{B}'$ aborts. Otherwise, $\mathcal{B}'$ passes $\mathcal{A}_I$'s queries to $\mathcal{C}'$ to answer.
- Decryption queries: $\mathcal{B}'$ uses an algorithm $\mathcal{KE}$ to handle $\mathcal{A}_I$'s decryption queries (so these responses may be incorrect).

When $\mathcal{A}_I$ picks ID_{ch} and m_0, m_1 on which it wishes to be challenges. If $\mathsf{ID}_{\text{ch}} \neq \mathsf{ID}_I$, then $\mathcal{B}'$ aborts. Otherwise $\mathcal{B}'$ forwards ID_{ch} to C' and responds to $\mathcal{A}_I$'s request for a challenge ciphertext with $C^* = \langle U', V', W^* \rangle$ where $\langle U', V' \rangle$ is the BasicCL-PKE challenge ciphertext given to $\mathcal{B}'$ by $\mathcal{C}'$ and W^* is chosen uniformly at random from $\{0,1\}^n$.

Eventually $\mathcal{A}_I$ outputs a bit b'. If necessary (when $\mathcal{A}_I$ runs for too long or makes too many hash queries), $\mathcal{B}'$ stops $\mathcal{A}_I$. Note that $\mathcal{B}'$ may also be forced to stop because it cannot respond to a particular query from $\mathcal{A}_I$. After stopping for whatever reason, $\mathcal{B}'$ chooses an element uniformly at random from the set $\{\sigma_j : 1 \leq j \leq q_3\} \cup \{\sigma'_i : 1 \leq i \leq q_4\}$ and outputs this element as its guess for σ^*.

It can be argued that, up to the point where Inv occurs in $\mathcal{B}'$'s simulation, the two simulations $\mathcal{B}$ and $\mathcal{B}'$ are indistinguishable to $\mathcal{A}_I$. This is because we ensured that $\mathcal{B}'$'s simulation aborts in the exact situations that $\mathcal{B}$'s simulation aborts. Furthermore, all the queries in $\mathcal{B}'$'s simulation which are handled by $\mathcal{C}'$ are indistinguishable from those handled in $\mathcal{B}$'s simulation. So the probability that Inv occurs in $\mathcal{B}'$'s simulation is exactly the same that it does in $\mathcal{B}$'s. Because of the relationship between the BasicCL-PKE and FullCL-PKE public keys, it can also be seen that if event Inv occurs, then $\mathcal{B}'$ has probability $1/(q_3+q_4)$ of outputting the correct BasicCL-PKE decryption of $\langle U', V' \rangle$. So $\mathcal{B}'$'s overall success probability is at least $\Pr[\mathsf{Inv}]/(q_3+q_4)$. But this is not greater than the highest success probability of any Type I OWE adversary against BasicCL-PKE that operates in the same time as $\mathcal{B}'$ and that makes q_2 and q_5 hash queries. Since the running time of $\mathcal{B}'$ is the same as that of $\mathcal{B}$, the claim follows.

□

Claim: $\Pr[\mathsf{Fail} | \neg\mathsf{Inv} \wedge \neg\mathcal{L}_1] \leq 3/q + 3 \cdot 2^{-n}$.

Proof. We analyse the event $\mathsf{Fail} | \neg\mathsf{Inv} \wedge \neg\mathcal{L}_1$ as follows. Here $\mathcal{KE}$ outputs $\perp$ because S_1 is empty, but this is an incorrect decryption. So in fact there exists a message M such that $C = \langle U, V, W \rangle$ encrypts M under ID_ℓ, P_ℓ. It is easy to see that, because $\langle U, V \rangle$ is a valid BasicCL-PKE ciphertext for ID_ℓ, P_ℓ, there exist unique $\sigma \in \{0,1\}^n$

and $r \in \mathbb{Z}_q^*$ such that:

$$\langle U, V \rangle = \mathsf{BasicCL\text{-}PKE\text{-}Encrypt}_{\mathsf{ID}_\ell, P_\ell}(\sigma; r).$$

Since S_1 is empty, we deduce that H_3 has not been queried on an input containing σ.

We consider two cases: either a valid $C \neq C^*$ has been produced by $\mathcal{A}_I$ from a message M using coins $r = H_3(\sigma, M)$ without σ having been queried of H_3, or in fact $C = C^*$ and this query occurs after the challenge phase. In the former case, it is easy to see that C will be a valid ciphertext with probability at most $1/q$, because a valid ciphertext $C = \langle U, V, W \rangle$ will have $U = rP$ where $r \in \mathbb{Z}_q^*$ is the output of random oracle H_3 on a query not made by $\mathcal{A}_I$.

We consider the latter case, where $C = C^*$ is a valid ciphertext, further. Now $\mathcal{KE}$ can only ever be queried on this ciphertext for a combination of identifier and public key ID_ℓ, P_ℓ not equal to $\mathsf{ID}_{\text{ch}}, P_{\text{ch}}$ because of the rules on adversary behaviour. We also know that $\mathsf{ID}_{\text{ch}} = \mathsf{ID}_I$ (because to receive this query, $\mathcal{B}$ must not have aborted at the challenge phase). Suppose then that

$$C^* = \langle r^* P, \sigma^* \oplus H_2(\hat{e}(Q, P_0)^{r^*}) \oplus H_5(r^* P_{\text{ch}}), m_b \oplus H_4(\sigma^*) \rangle$$

where $r^* = H_3(\sigma^*, m_b)$ and, as usual, P_{ch} denotes the value of ID_{ch}'s public key at the time when the challenge ciphertext was computed. The values σ^*, $H_4(\sigma^*)$ and r^* are unknown to $\mathcal{B}$ and $\mathcal{KE}$ (since $\mathcal{B}$'s challenger produces C^*). Since $C = C^*$, we have $rP = U = U^* = r^* P$ and so $r = r^*$. The probability that $\sigma \neq \sigma^*$ is $1/q$. For suppose that $\sigma \neq \sigma^*$. Then we have $H_3(\sigma, M) = r = r^* = H_3(\sigma^*, m_b)$, giving equal outputs for random oracle H_3 from distinct inputs. The probability of this event is $1/q$. So with probability $1 - 1/q$, we have $\sigma = \sigma^*$. But then because $r = r^*$, we must have

$$H_2(\hat{e}(Q_\ell, P_0)^r) \oplus H_5(rP_\ell) = H_2(\hat{e}(Q, P_0)^{r^*}) \oplus H_5(r^* P_{\text{ch}}). \tag{6.4}$$

We wish to evaluate the probability that, in fact, we have $rP_\ell = r^* P_{\text{ch}}$ and $\hat{e}(Q_\ell, P_0)^r = \hat{e}(Q, P_0)^{r^*}$. To do so, we bound the probability that (6.4) holds and these equalities are not both satisfied. There are three events to consider:

1. $(rP_\ell \neq r^*P_{\text{ch}}) \wedge (\hat{e}(Q_\ell, P_0)^r = \hat{e}(Q, P_0)^{r^*})$. From (6.4) we have $H_5(rP_\ell) = H_5(r^*P_{\text{ch}})$ with $rP_\ell \neq r^*P_{\text{ch}}$, so we have a collision for H_5, an event of probability 2^{-n}.

2. $(rP_\ell = r^*P_{\text{ch}}) \wedge (\hat{e}(Q_\ell, P_0)^r \neq \hat{e}(Q, P_0)^{r^*})$. From (6.4) we have a collision of H_2 on unequal inputs, an event of probability 2^{-n}.

3. $(rP_\ell \neq r^*P_{\text{ch}}) \wedge (\hat{e}(Q_\ell, P_0)^r \neq \hat{e}(Q, P_0)^{r^*})$. Rewriting (6.4), we obtain: $H_5(r^*P_{\text{ch}}) = H_5(rP_\ell) \oplus [H_2(\hat{e}(Q_\ell, P_0)^r) \oplus H_2(\hat{e}(Q, P_0)^{r^*})]$. Since neither pair of oracle inputs is equal, we again have an event of probability 2^{-n}.

So with probability $1 - (3 \cdot 2^{-n})$ we have $(r^*P_\ell = r^*P_{\text{ch}}) \wedge (\hat{e}(Q_\ell, P_0)^{r^*} = \hat{e}(Q, P_0)^{r^*}$. Since $r = r^*$, we obtain $P_\ell = P_{\text{ch}}$ and $\hat{e}(Q_\ell, P_0) = \hat{e}(Q, P_0)$. From this we have that $P_\ell = P_{\text{ch}}$ and $Q_\ell = Q$. The second equality implies that with probability $1 - 1/q$, $\mathsf{ID}_\ell = \mathsf{ID}_{\text{ch}}$. Thus, this is in fact the challenge query which is forbidden.

To sum up, $\perp$ is output incorrectly if any of these four events occur:

- C is valid when $(C \neq C^*)$, an event of probability q^{-1}.
- $H_3(\sigma, M) = H_3(\sigma^*, M)$ when $(\sigma \neq \sigma^*, C = C^*)$, an event of probability q^{-1}.
- (6.4) holds with unequal inputs into H_2 and/or H_5 when $(\sigma = \sigma^*, C = C^*)$, an event of probability $3 \cdot 2^{-n}$.
- $H_1(\mathsf{ID}_\ell) = H_1(\mathsf{ID}_{\text{ch}})$ when $(\mathsf{ID}_\ell \neq \mathsf{ID}_{\text{ch}}, \sigma = \sigma^*, C = C^*)$, an event of probability q^{-1}.

The claim follows.

□

The probablility of the next three claims are as in [6, Lemma 9].

Claim: $\Pr[\mathsf{Fail} | \neg\mathsf{Inv} \wedge \mathcal{L}_1 \wedge \neg\mathcal{L}_2] = 2^{-n}$.

Proof. In this situation, $\mathcal{KE}$ outputs $\perp$ because S_2 is empty, but this is an incorrect decryption. So in fact there exists a message M such that $C = \langle U, V, W \rangle$ encrypts M under ID_ℓ, P_ℓ. Now it is easy to see that, because $\langle U, V \rangle$ is a valid BasicCL-PKE ciphertext for ID_ℓ, P_ℓ, there exist unique $\sigma \in \{0,1\}^n$ and $r \in \mathbb{Z}_q^*$ such that:

$$\langle U, V \rangle = \mathsf{BasicCL\text{-}PKE\text{-}Encrypt}_{\mathsf{ID}_\ell, P_\ell}(\sigma; r).$$

But S_1 is non-empty, so we also have, $\langle U, V \rangle = \mathsf{BasicCL\text{-}PKE\text{-}Encrypt}_{\mathsf{ID}_\ell, P_\ell}(\sigma_j; H_{3,j})$, for some j. This implies that $\sigma = \sigma_j$ and $r = H_{3,j} = H_3(\sigma_j, M)$. Since S_2 is empty, we can deduce that H_4 has not been queried on input σ. Yet we must have $W = M \oplus H_4(\sigma)$ if C is a proper encryption of M. Moreover, we cannot simply define M by $M = W \oplus H_4(\sigma)$, since r is already defined by $r = H_3(\sigma_j, M)$ so M is already fixed. The probability that $W \oplus H_4(\sigma)$ outputs the correct M is exactly 2^{-n} and this bounds the probability that $\mathcal{KE}$ incorrectly outputs $\perp$.

□

Claim: $\Pr[\mathsf{Fail} | \neg\mathsf{Inv} \wedge \mathcal{L}_1 \wedge \mathcal{L}_2 \wedge \neg\mathsf{Find}] = 1/q$.

Proof. Here $\mathcal{KE}$ outputs $\perp$ because a failure occurs at step 3, but this is an incorrect decryption. Arguing as in the previous claim, we deduce that there exists a message M such that $C = \langle U, V, W \rangle$ encrypts M under ID_ℓ, P_ℓ, using unique $\sigma \in \{0,1\}^n$ and $r \in \mathbb{Z}_q^*$. Moreover, there exists a j with $\sigma = \sigma_j$ and $r = H_{3,j}$. Now S_2 is non-empty, so there exists an entry $\langle \sigma_j, M_j, H_{3,j}, H_{4,i} \rangle$ on the S_2 list with $\sigma_i' = \sigma_j = \sigma$.

Now suppose that $\langle \sigma, M \rangle$ has been queried of H_3. Then we would also have an entry $\langle \sigma, M, H_{3,j}, H_{4,i} \rangle$ on the S_2 list. But since C is the encryption of M, we would also have $W = M \oplus H_{4,i}$. Then $\mathcal{KE}$ would output M instead of $\perp$. This contradiction shows that $\langle \sigma, M \rangle$ has not been queried of H_3. Yet we must have $H_3(\sigma, M) = r = H_{3,j}$ if C is a proper encryption of M. The probability of this event occurring is exactly $1/q$ and this bounds the probability that $\mathcal{KE}$ incorrectly outputs $\perp$.

□

Claim: $\Pr[\mathsf{Fail} | \neg\mathsf{Inv} \wedge \mathcal{L}_1 \wedge \mathcal{L}_2 \wedge \mathsf{Find}] = 0$.

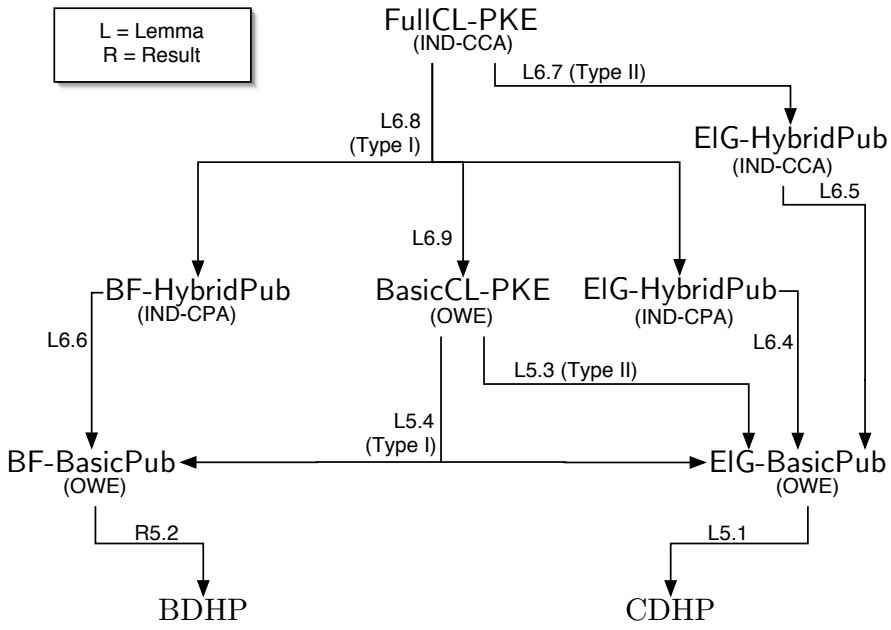

Figure 6.1: A summary of the lemmas and results of Chapters 5 and 6.

Proof. Here, $\mathcal{KE}$ outputs a message M_j whose encryption under the combination ID_ℓ, P_ℓ yields the ciphertext C with random oracles H_3 and H_4 as defined in $\mathcal{B}$'s simulation. Therefore the decryption of C is M_j, and $\mathcal{KE}$ never fails in this situation. The claim follows.

□

Gathering together each of these claims, we finally obtain

$$\Pr[\mathsf{Fail}] \leq (q_3 + q_4) \cdot \epsilon_{\mathrm{OWE}}(\mathrm{time}(\mathcal{B}) + \mathcal{O}((q_3 + q_4)q_d t', q_2, q_5) + 4q^{-1} + 2^{-n+2}.$$

The running time of $\mathcal{B}$ is $\mathrm{time}(\mathcal{A}_I) + q_d \cdot \mathrm{time}(\mathcal{KE}) = t + \mathcal{O}((q_3 + q_4)q_d t')$, where t' is the running time of the BasicCL-PKE encryption algorithm. This completes the proof of the lemma.

□

Figure 6.1 provides an overview of our overall approach to the proof of security for FullCL-PKE. It can be seen that our security proofs yield reductions to either the CDHP in $\mathbb{G}_1$ or BDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$. To conclude this section we will make a formal statement relating the security of FullCL-PKE to the hardness of the BDHP.

Theorem 6.10 Let hash functions H_i for $1 \leq i \leq 5$ be random oracles. Suppose further that there is no polynomially bounded algorithm that can solve the BDHP with non-negligible advantage. Then FullCL-PKE is IND-CCA secure.

Proof. As before, the proof of this theorem is performed in two parts where we relate the advantage of a Type I or Type II attacker against FullCL-PKE to that of an algorithm to solve BDHP or CDHP. We first consider a Type I adversary.

Type I adversary: Lemma 6.8 provides a reduction relating the IND-CCA security of FullCL-PKE to that of ElG-HybridPub or BF-HybridPub in the IND-CPA model for standard PKE. This reduction makes use of the special-purpose knowledge extraction algorithm to handle decryption queries which was studied in Lemma 6.9. Furthermore, in order for this knowledge extractor to have a large value of λ, we require the BasicCL-PKE scheme to be OWE secure if the BDHP is hard – the subject of Theorem 5.5. Thereafter, we reduce the security to that of ElG-BasicPub or BF-BasicPub against OWE adversaries using Lemma 6.4 and Lemma 6.6. Lemma 5.1 and Result 5.2 relate the security of these PKE schemes to the hardness of the CDHP or BDHP respectively. This sequence of reductions is represented in Figure 6.1.

By composing the intermediate security results we can relate the security of FullCL-PKE against Type I adversaries directly to the hardness of the BDHP or CDHP. Suppose hash functions H_i for $1 \leq i \leq 5$ are random oracles. Suppose $\mathcal{A}_I$ is a Type I adversary against FullCL-PKE. Suppose that $\mathcal{A}_I$ runs in time time($\mathcal{A}_I$), makes at most q_i queries of H_i ($1 \leq i \leq 5$), at most q_d decryption queries and has advantage ϵ against FullCL-PKE. Then there is an algorithm $\mathcal{B}$ with running time $O(\text{time}(\mathcal{A}_I) + n(q_3 + q_4) + q_d t'(q_3 + q_4))$, where t' is the running time of BasicCL-PKE encryption algorithm (defined in Section 5.4). Either algorithm $\mathcal{B}$ solves the CDHP in $\mathbb{G}_1$ with advantage at least

$$\frac{1}{q_5}\left\{\frac{1}{2(q_3+q_4)}\left[\left(\frac{\epsilon \cdot \lambda_1^{q_d}}{4q_1}+1\right)\left(1-\frac{1}{q}-2^{-n}\right)^{q_d}-1\right]-\frac{1}{2^n}\right\},$$

where λ_1 is

$$1 - (q_3 + q_4) \cdot \frac{1}{q_5} \left(\frac{\epsilon}{4q_1} - \frac{1}{2^n} \right) - 4q^{-1} - 2^{-n+2},$$

or algorithm $\mathcal{B}$ solves the BDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ with advantage at least

$$\frac{1}{q_2} \left\{ \frac{1}{2(q_3 + q_4)} \left[\left(\frac{\epsilon \cdot \lambda_2^{q_d}}{4q_1} + 1 \right) \left(1 - \frac{1}{q} - 2^{-n} \right)^{q_d} - 1 \right] - \frac{1}{2^n} \right\},$$

where λ_2 is

$$1 - (q_3 + q_4) \cdot \frac{1}{q_2} \left(\frac{\epsilon}{4q_1} - \frac{1}{2^n} \right) - 4q^{-1} - 2^{-n+2}.$$

Type II adversary: Lemma 6.7 shows that the IND-CCA security of FullCL-PKE can be reduced to the usual IND-CCA security of a related (normal) public key encryption scheme ElG-HybridPub. Lemma 6.5 reduces the security of ElG-HybridPub to that of a second public key encryption scheme ElG-BasicPub against OWE adversaries. Finally, Lemma 5.1 relates the security of ElG-BasicPub to the hardness of the CDHP in $\mathbb{G}_1$. This sequence of reductions is represented in the right hand side of Figure 6.1.

As above, we can relate security against Type II adversaries directly to the hardness of the CDHP. Suppose hash functions H_i for $1 \leq i \leq 5$ are random oracles. Suppose $\mathcal{A}_{II}$ is a Type II adversary against FullCL-PKE. Suppose that $\mathcal{A}_{II}$ runs in time time($\mathcal{A}_{II}$), makes at most q_i queries of H_i ($1 \leq i \leq 5$), at most q_d decryption queries and has advantage ϵ against FullCL-PKE. Then there is an algorithm $\mathcal{B}$ with running time $O(\text{time}(\mathcal{A}_{II}) + n(q_3 + q_4))$ that solves the CDHP in $\mathbb{G}_1$ with advantage at least

$$\frac{1}{q_5} \left\{ \frac{1}{2(q_3 + q_4)} \left[\left(\frac{\epsilon}{q_1} + 1 \right) \left(1 - \frac{1}{q} - 2^{-n} \right)^{q_d} - 1 \right] - \frac{1}{2^n} \right\}.$$

Since the CDHP in $\mathbb{G}_1$ (output by $\mathcal{IG}(k)$) is a harder problem than the BDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ (output by the same $\mathcal{IG}(k)$), we can finally say that the security of FullCL-PKE is related to the hardness of the BDHP.

□

6.6 Summary

The CL-PKE scheme presented here, which is a hybridisation of the CL-PKE scheme of Chapter 5, enjoys short public and private keys and is secure in an appropriate and robust model assuming that the BDHP is hard. The standard IND-CCA notion is the privacy notion most PKE designs aim to achieve. Our model extends the IND-CCA notion to our new setting, which includes the KGC and is fully adaptive. The scheme is fast, compact, simple, interoperable and highly practical. Furthermore, it improves on the previously published scheme in [6].

Chapter 7

Generic CL-PKE Schemes

Contents

We explore how CL-PKE schemes can be constructed generically by combining standard public key encryption (PKE) and identifier-based encryption (ID-PKE) schemes. We present an analysis of Gentry's concept of certificate-based encryption (CBE). We then explore how CBE schemes can be constructed using CL-PKE schemes.

7.1 Introduction

In this chapter we examine how an arbitrary ID-PKE scheme and an arbitrary PKE scheme can be combined to construct CL-PKE schemes. These generic constructions are important because they provide a better understanding of CL-PKE schemes.

Note that the efficient CL-PKE schemes in Chapters 5 and 6 are derived from one of our generic transformations through a process of optimisation for specific ID-PKE and standard PKE schemes.

A related idea is that of the optimised double encryption scheme presented by Gentry [76]. Gentry's scheme has closely related properties to those of the CL-PKC(B) encryption scheme described in Chapter 4 and developed in Chapters 5 and 6. Recall that CL-PKC(B) schemes include a public key in the identifier and do not use time or a certificate infrastructure as in [76]. This connection between CL-PKE and CBE was already recognised in [6] and in Section 4.3.3. We will explore it further in this chapter. To fully explore the relationship between CL-PKE and CBE, we describe and analyse the CBE definition and the CBE security model. The analysis highlights numerous weaknesses in the definitions and models of [76]. An improved definition of CBE is provided. We then use this definition to show how to build IND-CCA secure CBE schemes from any IND-CCA secure CL-PKE scheme.

7.2 Some Generic CL-PKE Constructions

In what follows generic methods of constructing CL-PKE schemes by combining a general ID-PKE scheme with a standard PKE schemes will be briefly considered. We will provide three generic CL-PKE schemes constructed in this way: CL-1, CL-2 and CL-3. Roughly speaking, for each of the constructions, the Partial-Private-Key-Extract algorithm is handled by the ID-PKE scheme, and the Set-Private-Key/Set-Public-Key algorithms are handled by the standard PKE scheme.

A generic scheme of the type constructed here can be used to add cryptographic workflow to a standard PKE scheme by composing the standard PKE scheme with an ID-PKE scheme; the resultant scheme no longer requires certificates. Similarly, a generic scheme can be constructed to enhance the level of trust offered by an ID-PKE scheme by composing the ID-PKE scheme with a standard PKE scheme; the resultant scheme will, however, no longer be identifier-based.

Now let us consider an IND-ID-CCA secure ID-PKE scheme, Π^{ID}, and an IND-CCA secure standard PKE scheme, Π^{PK}. These will be composed in order to create our first generic CL-PKE scheme, denoted $\Pi^{\mathrm{CL}-1}$. Note that Canetti *et al.* [44] show how IND-CCA secure PKE schemes can be constructed using any CPA secure ID-PKE scheme. The result in [44] allows Π^{ID} and Π^{PK} to share many algorithms.

In what follows we assume that Π^{ID} and Π^{PK} are compatible in the sense that the ciphertext space of Π^{PK} is equal to the message (plaintext) space of Π^{ID}. The seven algorithms needed to define $\Pi^{\mathrm{CL}-1}$ are described next. We assume that schemes Π^{PK} and Π^{ID} take as input security parameters k_1 and k_2 respectively.

Setup: This algorithm runs the Setup algorithm of the scheme Π^{PK} and the Setup algorithm of the scheme Π^{ID}. The message space of $\Pi^{\mathrm{CL}-1}$ will be the message space of Π^{ID}, denoted $\mathcal{M}$, while the ciphertext space of $\Pi^{\mathrm{CL}-1}$ will be the ciphertext space of Π^{ID}.

Partial-Private-Key-Extract: This algorithm is defined to be the Extract algorithm of Π^{ID}. So the partial private key D_A of ID_A in $\Pi^{\mathrm{CL}-1}$ is set to be the private key d_A of ID_A in the scheme Π^{ID}.

Set-Secret-Value and Set-Public-Key: These algorithms are obtained from the Key-Generation algorithm of Π^{PK}. Algorithm Key-Generation is run, and the output of Set-Secret-Value algorithm, x_A, is defined to be the private key K_{priv} for Π^{PK}, while the output of the Set-Public-Key algorithm, P_A, is defined to be the public key K_{pub} for Π^{PK}.

Set-Private-Key: This algorithm outputs $S_A = \langle D_A, x_A \rangle$, where, as above D_A is the private key corresponding to identifier ID_A in the scheme Π^{ID} and x_A is a private key obtained from the scheme Π^{PK}.

Encrypt: To encrypt $M \in \mathcal{M}$ for identifier ID_A and public key P_A, perform the following steps:

1. Check that P_A is a valid public key for Π^{PK}, if not output $\perp$.

2. Compute and output the ciphertext:

$$C = \mathcal{E}^{\mathrm{ID}}(\mathcal{E}^{\mathrm{PK}}(M, P_A), \mathsf{ID}_A).$$

   Here, $\mathcal{E}^{\mathrm{ID}}$ denotes the encryption algorithm of the scheme Π^{ID} and $\mathcal{E}^{\mathrm{PK}}$ denotes the encryption algorithm of the scheme Π^{PK}.

Decrypt: Suppose $C \in \mathcal{C}$. To decrypt this ciphertext using the private key $S_A = \langle D_A, x_A \rangle$, firstly compute $\mathcal{D}^{\mathrm{ID}}(C, D_A)$. If the result is equal to $\perp$, then output $\perp$ and reject the ciphertext. Otherwise output $\mathcal{D}^{\mathrm{PK}}(\mathcal{D}^{\mathrm{ID}}(C, D_A), x_A)$. Here, $\mathcal{D}^{\mathrm{ID}}$ denotes the decryption algorithm of Π^{ID} and $\mathcal{D}^{\mathrm{PK}}$ denotes the decryption algorithm of Π^{PK}.

An alternative serial encryption scheme to $\Pi^{\mathrm{CL}-1}$ is one which reverses the order of encryption, such that $C = \mathcal{E}^{\mathrm{PK}}(\mathcal{E}^{\mathrm{ID}}(M, \mathsf{ID}_A), P_A)$. This scheme will be labelled $\Pi^{\mathrm{CL}-2}$. Here, of course, we require that the ciphertexts output by $\mathcal{E}^{\mathrm{ID}}$ can be used as plaintext for the encryption algorithm of $\mathcal{E}^{\mathrm{ID}}$.

The scheme denoted $\Pi^{\mathrm{CL}-3}$ is a parallel encryption scheme. As we shall see, details in the algorithms differ. For $\Pi^{\mathrm{CL}-3}$, we need to assume that Π^{PK} and Π^{ID} are compatible in the sense that they both have the same plaintext space, denoted $\mathcal{M}$. We also assume that $\mathcal{M}$ consists of the set of strings of some length n. The seven algorithms needed to define $\Pi^{\mathrm{CL}-3}$ are described next. As with $\Pi^{\mathrm{CL}-1}$, here $\mathcal{E}^{\mathrm{ID}}/\mathcal{D}^{\mathrm{ID}}$ denotes the encryption/decryption algorithm of the scheme Π^{ID} and $\mathcal{E}^{\mathrm{PK}}/\mathcal{D}^{\mathrm{PK}}$ denotes the encryption/decryption algorithm of the scheme Π^{PK}.

Setup: This algorithm runs the Setup algorithm of the scheme Π^{PK} and the Setup algorithm of scheme Π^{ID}.

Partial-Private-Key-Extract: Identical to Partial-Private-Key-Extract of $\Pi^{\mathrm{CL}-1}$.

Set-Secret-Value and Set-Public-Key: Identical to Set-Secret-Value and Set-Public-Key

of $\Pi^{\mathrm{CL}-1}$.

Set-Private-Key: Identical to Set-Private-Key of $\Pi^{\mathrm{CL}-1}$.

Encrypt: To encrypt $M \in \mathcal{M}$ for identifier ID_A and public key P_A, perform the following steps:

1. Check that P_A is a valid public key for Π^{PK}, if not output $\perp$.
2. Choose a random M_{A} with the same bit length as M.
3. Set $M_{\mathrm{B}} = M_{\mathrm{A}} \oplus M$.
4. Compute and output the ciphertext:
$$C = \langle \mathcal{E}^{\mathrm{ID}}(M_{\mathrm{A}}, \mathsf{ID}_A), \mathcal{E}^{\mathrm{PK}}(M_{\mathrm{B}}, P_A) \rangle.$$

Decrypt: Suppose $C = \langle c_{\mathrm{A}}, c_{\mathrm{B}} \rangle \in \mathcal{C}$. To decrypt this ciphertext using the private key $S_A = \langle D_A, x_A \rangle$, firstly compute $\mathcal{D}^{\mathrm{ID}}(c_{\mathrm{A}}, D_A)$ and $\mathcal{D}^{\mathrm{PK}}(c_{\mathrm{B}}, x_A)$. If either result is equal to $\perp$, then output $\perp$ and reject the ciphertext. Otherwise output $M = \mathcal{D}^{\mathrm{ID}}(c_{\mathrm{A}}, D_A) \oplus \mathcal{D}^{\mathrm{PK}}(c_{\mathrm{B}}, x_A)$.

This concludes the description of $\Pi^{\mathrm{CL}-3}$.

Notice that if the BF ID-PKE scheme [32] and the ElGamal PKE scheme [68] are used directly in the generic construction $\Pi^{\mathrm{CL}-1}$, the resulting construction is computationally rather inefficient: in $\Pi^{\mathrm{CL}-1}$ both $\mathcal{E}^{\mathrm{ID}}$ and $\mathcal{E}^{\mathrm{PK}}$ are run independently using different plaintexts, random values and redundancies. The scheme FullCL-PKE in Chapter 6 can be regarded as an optimisation of $\Pi^{\mathrm{CL}-1}$ where the components of the scheme are FullIdent of [32] and ElG-HybridPub of Section 6.5.1. Our proof of security for that scheme utilised a particular knowledge extractor which decrypts ciphertexts with a high probability of success.

Given the proof techniques developed in previous chapters, the main obstacle in proving the security of these generic constructions in the security model developed

in Chapter 6 appears to be the construction of a *general* knowledge extractor that is appropriate to the Type I adversary setting (which is very different to existing settings). This knowledge extractor is required to decrypt ciphertexts with high probability of success for an entity whose public key may have been replaced.

7.3 Analysis of CBE

This section will clarify the relationship between the definition and security model of CBE [76] and those of CL-PKE. This is done because, as we pointed out, some similarities do exist between CBE and CL-PKE(B) schemes. Formalising this relationship is beneficial as it will elucidate both independently developed models and highlight any overlaps between them. This process will allow the reader to understand how CBE is related to our contribution.

When we started to examine the CBE definition and security model, we found them deficient in many ways. It is not our intention to criticize CBE, even though the results of our analysis point out some major shortcomings.

7.3.1 Gentry's Definition for CBE

First we provide the formal definition of CBE:

Definition 7.1 ([76]) A certificate-updating certificate-based encryption scheme is defined by six algorithms (Gen_{IBE}, Gen_{PKE}, Upd1, Upd2, $\mathcal{E}$, $\mathcal{D}$) such that:

1. Algorithm Gen_{IBE} is a probabilistic ID-PKE key generation algorithm that takes a security parameter k_1 and (optionally) the total number of time periods t as input. It returns SK_{IBE} (the certifier's master-key) and public parameters params that include a public key PK_{IBE}, and the description of a string space Λ.

2. Algorithm Gen_{PKE} is a probabilistic PKE key generation algorithm that takes a security parameter k_2 and (optionally) the total number of time periods t as input. It returns a private key SK_{PKE} and public key PK_{PKE}.

3. At the start of time period τ, the deterministic certifier update algorithm Upd1 takes as input SK_{IBE}, params, τ, $\lambda \in \Lambda$ and PK_{PKE}. It returns Cert'_τ.

4. At the start of time period τ, the deterministic update algorithm Upd2 takes as input params, τ, Cert'_τ, and (optionally) $\text{Cert}_{\tau-1}$. It returns Cert_τ.

5. Algorithm $\mathcal{E}$ is a probabilistic encryption algorithm that takes (params, τ, λ, PK_{PKE}, M) as input, where M is a message. It returns a ciphertext C on message M intended for the entity to decrypt using Cert_τ and SK_{PKE} (and possibly λ).

6. Algorithm $\mathcal{D}$ is a deterministic decryption algorithm that takes (params, Cert_τ, SK_{PKE}, C) as input in time period τ. It returns either M or the special symbol $\perp$ indicating failure. We require $\mathcal{D}_{\text{Cert}_\tau, SK_{PKE}, \lambda}(\mathcal{E}_{\tau,\lambda,PK_{IBE},PK_{PKE}}(M)) = M$ for the given params.

The algorithm Gen_{IBE} is similar to the algorithm setup in an ID-PKE scheme. The algorithm Gen_{PKE} is similar to the combined algorithm setup and $\mathcal{K}$ in a standard PKE scheme, with input to setup and output from $\mathcal{K}$.

Analysing the CBE Definition

We highlight some weaknesses in Definition 7.1:

1. Property (1) of Definition 7.1 requires PK_{IBE} to be an identifiable element of the ID-PKE scheme's parameters that can be labelled as a public key (notice that PK_{IBE} is also used during encryption). In the BF ID-PKE scheme presented in Section 3.2.4.1, PK_{IBE} corresponds to P_0. Given that not every

ID-PKE scheme need have this property, Definition 7.1 limits the ID-PKE schemes that can be used in generating CBE schemes.

2. As we can see from property (1) of Definition 7.1, combining an ID-PKE scheme with a standard PKE scheme is *explicitly* required for building a CBE scheme. This is another limitation in the way CBE schemes are constructed and means that a general CL-PKE scheme does not necessarily give rise to a CBE scheme, even though it can have all the functionality of a CBE scheme. This limitation leads to the discrepancy explained next.

3. None of the concrete schemes in [76] fit the definition of a CBE scheme given in [76] and reproduced here as Definition 7.1. This is because Gentry's concrete CBE schemes are all set up using a single generation algorithm with a single security parameter and all the key pairs $\langle SK_{PKE}, PK_{PKE}\rangle$ are computed based on system wide parameters. Thus, there is a major incompatibility in [76] between the definition of CBE on the one hand and the concrete CBE schemes on the other.

4. According to property (5) of Definition 7.1, only publicly available values are required to run algorithm $\mathsf{Upd2}$. Algorithm $\mathsf{Upd2}$ lacks any secret or private input such as SK_{IBE}. Hence, any entity can run algorithm $\mathsf{Upd2}$ to update all the certificates in the system. This of course defeats the purpose of an update algorithm.

5. In property (6) of Definition 7.1 and the description of encryption schemes in [76], the public key used in CBE is always assumed to be valid. In no circumstance does the encryption algorithm output fail and reject the public key. The encryption algorithms of the concrete CBE schemes in [76, §3] do not perform any certificate verification, and checking the public key for these schemes is crucial. We regard this as a weakness which could lead to practical attacks.

6. The encryption algorithm $\mathcal{E}$ of Definition 7.1 appears to make use of elements of both an ID-PKE scheme and a standard PKE scheme. However, the required relationship between the ID-PKE scheme, the standard PKE scheme and $\mathcal{E}$ is

not specified. Recall that in Section 7.2 we described some examples which explain why schemes, which are in some sense compatible, are required to construct a single scheme.

7. All the concrete schemes in [76] and all the algorithms in Definition 7.1 depend explicitly on time. Although it was noted [76, p.278] that the CBE scheme need not be used for certificate updating, Gentry [76] did not investigate in any detail the applications of such a scheme.

It is clear from this analysis that Definition 7.1 has many problems. Therefore in Section 7.4.1 we provide the reader with an alternative definition for CBE. The concrete schemes in [76] meet our modified definitions.

7.3.2 Gentry's Security Model for CBE

Security for CBE is defined using two different games in [76]. The adversary chooses which game to play. A CBE scheme is secure if no adversary can win either game. In Game 1 the adversary models an uncertified entity and in Game 2 the adversary models the certifier.

We now describe these IND-CCA aversarial games in more detail, following [76, §2.2].

CBE Game 1 Adversary: The challenger runs $Gen_{IBE}(k_1, t)$, and gives params to the adversary $\mathcal{A}_1$. The adversary then interleaves certification and decryption queries with a single challenge query. These queries are answered as follows:

- On certification query $\langle \tau, \lambda, PK_{PKE}, SK_{PKE} \rangle$, the challenger checks that $\lambda \in \Lambda$ and that SK_{PKE} is the private key corresponding to PK_{PKE}. If so, it runs Upd1 and returns Cert'_τ, else it returns $\perp$.
- On decryption query $\langle \tau, \lambda, PK_{PKE}, SK_{PKE}, C \rangle$, the challenger checks that $\lambda \in \Lambda$ and that SK_{PKE} is the private key corresponding to PK_{PKE}. If so, it

generates Cert_τ and outputs $\mathcal{D}_{Cert_\tau, SK_{PKE},\lambda}(C)$, else it returns $\perp$.

- On challenge query $(\tau_{\text{ch}}, \lambda_{\text{ch}}, PK_{PKE,\text{ch}}, SK_{PKE,\text{ch}}, M_0, M_1)$, the challenger checks that $\lambda_{\text{ch}} \in \Lambda$ and that $SK_{PKE,\text{ch}}$ is the private key corresponding to $PK_{PKE,\text{ch}}$. If so, it chooses random bit b and returns

$$C^* = \mathcal{E}_{\tau_{\text{ch}},\lambda_{\text{ch}},PK_{IBE},PK_{PKE,\text{ch}}}(M_b),$$

else it returns $\perp$.

Finally, $\mathcal{A}_1$ outputs a guess $b' \in \{0,1\}$. The adversary wins the game if $b = b'$ and $\langle \tau_{\text{ch}}, \lambda_{\text{ch}}, PK_{PKE,\text{ch}}, SK_{PKE,\text{ch}}, C^* \rangle$[7.1] was not the subject of a decryption query after the challenge, and $\langle \tau_{\text{ch}}, \lambda_{\text{ch}}, PK_{PKE,\text{ch}}, SK_{PKE,\text{ch}} \rangle$ was not the subject of any certification query. We define $\mathcal{A}_1$'s advantage in this game to be $\text{Adv}(\mathcal{A}_1) := 2|\Pr[b = b'] - \frac{1}{2}|$.

CBE Game 2 Adversary: The challenger runs $Gen_{PKE}(k_2, t)$, and gives PK_{PKE} to the adversary $\mathcal{A}_2$. The adversary then interleaves decryption queries with a single challenge query. These queries are answered as follows:

- On decryption query $\langle \tau, \lambda, \mathsf{params}, SK_{IBE}, C \rangle$, the challenger checks that $\lambda \in \Lambda$ and that SK_{IBE} is the master-key corresponding to params. If so, it generates Cert_τ and outputs $\mathcal{D}_{\text{Cert}_\tau, SK_{PKE},\lambda}(C)$, else it returns $\perp$.
- On challenge query $\langle \tau_{\text{ch}}, \lambda_{\text{ch}}, \mathsf{params}_{\text{ch}}, SK_{IBE,\text{ch}}, M_0, M_1 \rangle$, the challenger checks that $\lambda_{\text{ch}} \in \Lambda$ and that $SK_{IBE,\text{ch}}$ is the master-key corresponding to $\mathsf{params}_{\text{ch}}$. If so, it chooses random bit b and returns $C^* = \mathcal{E}_{\tau_{\text{ch}},\lambda_{\text{ch}},PK_{IBE,\text{ch}},PK_{PKE}}(M_b)$, else it returns $\perp$.

Finally, $\mathcal{A}_2$ outputs a guess $b' \in \{0,1\}$. The adversary wins the game if $b = b'$ and $\langle \tau_{\text{ch}}, \lambda_{\text{ch}}, \mathsf{params}_{\text{ch}}, SK_{IBE,\text{ch}}, C^* \rangle$ was not the subject of a decryption query after the challenge. We define $\mathcal{A}_2$'s advantage in this game to be $\text{Adv}(\mathcal{A}_2) := 2|\Pr[b = b'] - \frac{1}{2}|$.

[7.1] Note that $SK_{PKE,\text{ch}}$ was omitted here in [76].

Definition 7.2 ([76]) **??** A certificate-updating certificate-based encryption scheme is secure against adaptive chosen ciphertext attack (IND-CBE-CCA) if no probabilistic polynomial time adversary has non-negligible advantage in either CBE Game 1 or CBE Game 2.

Analysing the CBE Security Model

In this section, we present an analysis of the CBE security model, and compare it to the security model for CL-PKE that was developed in Chapters 5 and 6. Notice that our CL-PKE security model assumes an adversary who can extract partial private keys and change public keys even for the challenge identity, whereas Gentry's model, in which the equivalent of partial private keys are publicly available and bind the public keys (and time periods) to identities, simulates the adversary differently. Unfortunately, some major weaknesses exist in the CBE security model of [76]:

1. Game 1 does not capture an adversary obtaining a 'certificate' for an existing public key that the adversary intends to attack. This method of attack is natural for an uncertified client. The reason this restriction arises is because the challenger initially controls the setting of public keys for entities in the CBE system. We do not have such a restriction in the CL-PKE security model: our Type I adversaries are truly adaptive in nature.

2. A Game 1 adversary must provide the private key along with the corresponding public key. This is done by giving private keys to the challenger when making any query involving public keys (even the challenge query). In CL-PKE, we allow our Type I adversary to change an entity's public key without needing to show the private key. This gives the adversary more flexibility, for example, the adversary can replace the public key of an entity with that of another (without knowing the corresponding private key). We are able to handle this in our proofs by the use of special purpose knowledge extractors.

3. A Game 2 adversary does *not* get to choose a public key to attack and is

given a specific public key by the challenger. This is unlike a CL-PKE Type II adversary.

4. A Game 2 adversary proves knowledge of the master-key corresponding to params by giving every master-key to the challenger. This adversary can alter the CBE scheme by choosing new parameters for the ID-PKE scheme in *each* query. This unnecessarily complicates the way the adversary is modelled. In CL-PKE, a Type II adversary is given the master-key to allow it to 'break' part of the scheme which the KGC is always able to break. Handling only one master-key is more natural because the aim of the proof is to examine the security of a system with a pre-specified set of parameters, that is, a system which has been set up.

It can be argued that the unusual constraint expressed in weakness (2) above can be removed for the proof of the first concrete scheme in [76]. However, the proof of security for that scheme suffers from a deficiency. The decryption queries in the proof of [76, Lemma 1], do not work as defined unless the hash queries are modified in the simulation by setting '$P'_j = b_j P$' for $coin_j = 0$. Unlike this simulation problem, the above differences are significant enough to illustrate that the CBE definition and security model inadequately capture the concept we explored in CL-PKE: a concept which represents a shift in how public keys are managed and used. CBE is a very interesting concept for an encryption scheme which is suitable for solving a particular problem: that of efficient revocation in traditional public key infrastructures.

7.3.3 Gentry's Concrete CBE Scheme

The scheme BasicCBE of [76] is described using the notation established in Chapter 2 as follows:

- Setup: The CA (i) runs $\mathcal{IG}$ on input k to generate $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$; (ii) chooses $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^n$ for some n; (iii) chooses $P \in \mathbb{G}_1$, random

$s_C \in \mathbb{Z}_q^*$ and sets $P_0 = s_C P \in \mathbb{G}_1$ and params$= \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_0, H_1, H_2 \rangle$. The CA's secret is $s_C \in \mathbb{Z}_q^*$ which is used to issue certificates. The message space is $\mathcal{M} = \{0,1\}^n$. The ciphertext space is $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^n$.

- Set-Key-Pair: Entity A selects a private and public key pair $\langle s_A, s_A P \rangle$.
- Certify: (i) Entity A sends $\mathsf{ID}_A \| s_A P$ to the CA. (ii) The CA produces and distributes certificate $\text{Cert}_{A,\tau} = s_c H_1(P_0 \| \tau \| \mathsf{ID}_A \| s_A P) \in \mathbb{G}_1$ in time period τ. Notice that the private key that can be computed by A in time period τ is $S_A = \text{Cert}_{A,\tau} + s_A H_1(\mathsf{ID}_A \| s_A P) \in \mathbb{G}_1$.
- Encrypt: To encrypt $M \in \mathcal{M}$ for A, entity B (i) computes $H_1(\mathsf{ID}_A \| s_A P) \in \mathbb{G}_1$; (ii) computes $H_1(P_0 \| \tau \| \mathsf{ID}_A \| s_A P) \in \mathbb{G}_1$; (iii) chooses a random value $r \in \mathbb{Z}_q^*$; and (iv) computes:

  $$C = \langle rP, M \oplus H_2((\hat{e}(P_0, H_1(P_0 \| \tau \| \mathsf{ID}_A \| s_A P)) \cdot \hat{e}(s_A P, H_1(\mathsf{ID}_A \| s_A P)))^r) \rangle.$$

  Notice that steps (i) and (ii) require only public information.
- Decrypt: To decrypt $C = \langle U, V \rangle \in \mathcal{C}$, entity A computes:

  $$M = V \oplus H_2(\hat{e}(U, S_A)).$$

The semantically secure scheme FullCBE of [76] applies the hybridisation technique described in Section 6.4 to the scheme BasicCBE above.

Notice that the public key, $s_A P$, is simply computed according to the parameters issued by the CA. We described in Section 7.3.1 why the concrete CBE schemes in [76] fail to meet Definition 7.1. Also notice that the encryption algorithm of scheme BasicCBE uses the public key $s_A P$ without first checking the validity of the public key.

Analysis of the Concrete Scheme

Gentry's schemes BasicCBE and FullCBE use a form of double encryption and have some overlap in properties with CL-PKE schemes. The CBE schemes BasicCBE

and FullCBE were designed to use H_1 with two different inputs, '$P_0\|\tau\|\mathsf{ID}_A\|s_AP$' and '$\mathsf{ID}_A\|s_AP$', to ensures the separation of both the standard PKE and ID-PKE schemes. In designing the schemes which we described in Chapters 5 and 6, we make the separation more explicit by using two hash functions, H_2 and H_5. This allows us to run the simulations required for our proofs.

Notice that the scheme BasicCBE has some mathematical similarities with BasicCL-PKE, however, as we have seen in Sections 7.3.1 and 7.3.2 the fit between the CBE and CL-PKE definition and security model are not as natural as one would hope. Nevertheless, it would be beneficial to be able to translate CL-PKE schemes such as FullCL-PKE into IND-CBE-CCA secure CBE schemes. In addition to producing an alternative CBE scheme, the translation could reduce the computational overhead of the CBE schemes of Gentry [76].

We will show how to translate CL-PKE schemes to a CBE schemes in the next section.

7.4 Secure CBE from Secure CL-PKE

A very important functional distinction between CBE and CL-PKE is that CL-PKE allows for an entity to use multiple public keys for the same partial public key. Furthermore, the public keys in a CL-PKE scheme need not be generated before the partial private key, whereas the CBE definition requires the public key to be generated before certification. Nevertheless, in this section, we are able to show how to construct a CBE scheme using a CL-PKE scheme. After providing the construction, we prove that the resulting CBE scheme is IND-CBE-CCA secure (according to Definition **??**), provided the CL-PKE scheme is IND-CCA secure (in the model of Chapter 6).

We begin by providing the following simplified definition of CBE. This definition is consistent with the concrete CBE schemes of [76] and fixes most of the problems that

we have identified above.

Definition 7.3 A certificate-based encryption scheme is defined by five algorithms (Setup, Set-Key-Pair, Certify, $\mathcal{E}^{\mathrm{CBE}}$, $\mathcal{D}^{\mathrm{CBE}}$) such that:

1. Algorithm Setup is a probabilistic algorithm that takes a security parameter k. It returns SK_{IBE} (the certifier's master-key) and public parameters params that include the description of a string space Λ.

2. Algorithm Set-Key-Pair is a probabilistic algorithm that takes params as input. It returns a private key SK_{PKE} and public key PK_{PKE}[7.2]

3. Algorithm Certify is a deterministic certifier update algorithm that takes as input SK_{IBE}, params, τ, $\lambda \in \Lambda$ and PK_{PKE}. It returns Cert_τ.

4. Algorithm $\mathcal{E}^{\mathrm{CBE}}$ is a probabilistic encryption algorithm that takes (params, τ, λ, PK_{PKE}, M) as input, where M is a message. It returns a ciphertext C on message M intended for the entity to decrypt using Cert_τ and SK_{PKE}.

5. Algorithm $\mathcal{D}^{\mathrm{CBE}}$ is a deterministic decryption algorithm that takes (params, Cert_τ, SK_{PKE}, C) as input in time period τ. It returns either M or the special symbol $\perp$ indicating failure. We require $\mathcal{D}^{\mathrm{CBE}}_{\mathrm{Cert}_\tau, SK_{PKE}, \lambda}(\mathcal{E}^{\mathrm{CBE}}_{\tau,\lambda,\mathsf{params},PK_{PKE}}(M)) = M$.

The existing model of security described in Section 7.3.2 also applies to Definition 7.3. Although one can strengthen the existing CBE security model, an alternative security model will not be produced. We have hinted how the CBE security model can be strengthened in Section 7.3.2. Additionally, in the proof which we present in Section 7.4.2, we point to how the principles of proving security for CL-PKE can be used to provide improvements to the CBE security proofs and model.

[7.2] Even though labels PKE and IBE are used to identify components of the scheme, our definition of CBE does not require the use of PKE and ID-PKE schemes.

7.4.1 CBE Schemes from CL-PKE Schemes

In this section we sketch how to construct a CBE scheme, Π^{CBE}, from a CL-PKE scheme Π^{CL} by defining the five algorithms (Setup, Set-Key-Pair, Certify, $\mathcal{E}^{\mathrm{CBE}}$, $\mathcal{D}^{\mathrm{CBE}}$) of the CBE scheme in terms of those of the CL-PKE scheme.

1. Setup: This algorithm takes a security parameter k and returns SK_{IBE} and public parameters params that includes the description of a string space Λ. We use algorithm Setup of Π^{CL} to define Setup of Π^{CBE}, setting SK_{IBE} and params of Π^{CBE} to be master-key and params of Π^{CL}.

2. Set-Key-Pair: This algorithm runs the Set-Secret-Value and Set-Public-Key algorithm of the scheme Π^{CL}. It takes params as input and should output SK_{PKE} and PK_{PKE}. The output SK_{PKE} is defined to be the output x_A of Set-Secret-Value and the output PK_{PKE} is defined to be the output P_A of Set-Public-Value.

3. Certify: This algorithm takes as input SK_{IBE}, params, τ, $\lambda \in \Lambda$ and PK_{PKE}. It returns Cert_τ. We use algorithm Partial-Private-Key-Extract of Π^{CL} to define Certify, setting Cert_τ to be the partial private key for identity

$$\mathsf{params}\|\tau\|\lambda\|PK_{PKE}.$$

4. $\mathcal{E}^{\mathrm{CBE}}$: This algorithm takes (params, τ, λ, PK_{PKE}, M) as input, where M is a message. It returns a ciphertext C on message M intended for the entity to decrypt using Cert_τ and SK_{PKE}. We use the Encrypt algorithm of Π^{CL} to define $\mathcal{E}^{\mathrm{CBE}}$, setting

$$C = \mathcal{E}^{\mathrm{CL}}(M, P_A, \mathsf{ID}_A),$$

where $\mathsf{ID}_A = \mathsf{params}\|\tau\|\lambda\|PK_{PKE}$.

5. $\mathcal{D}^{\mathrm{CBE}}$: This algorithm takes (params, Cert_τ, SK_{PKE}, C) as input in time period τ. It returns either M or the special symbol $\perp$ indicating failure. We use the Decrypt algorithm of Π^{CL} to define $\mathcal{D}^{\mathrm{CBE}}$, setting

$$\mathcal{D}^{\mathrm{CL}}(C, \langle x_A, D_A\rangle),$$

where D_A is the partial private key for identity $\mathsf{ID}_A = \mathsf{params}\|\tau\|\lambda\|PK_{PKE}$.

7.4.2 Security of CBE schemes from CL-PKE Schemes

Next is our main theorem about the IND-CBE-CCA security of CBE schemes constructed using CL-PKE schemes.

Theorem 7.1 Suppose that Π^{CL} is an IND-CCA secure CL-PKE scheme, and suppose that Π^{CL} is used to build a CBE scheme Π^{CBE} as in Section 7.4.1. Then Π^{CBE} is IND-CBE-CCA secure.

Proof. We begin this proof by considering a Game 1 adversary against Π^{CBE}.

Let $\mathcal{A}_1$ be a Game 1 IND-CCA adversary against Π^{CBE} with advantage ϵ. We show how to construct from $\mathcal{A}_1$ a Type I IND-CCA adversary $\mathcal{B}$ against Π^{CL}.

Let $\mathcal{C}$ denote the challenger against our IND-CCA adversary $\mathcal{B}$ for Π^{CL}. The challenger $\mathcal{C}$ begins by supplying $\mathcal{B}$ with the parameters of Π^{CL}. Adversary $\mathcal{B}$ mounts an IND-CCA attack on Π^{CL} using help from $\mathcal{A}_1$ as follows.

Adversary $\mathcal{B}$ simulates the algorithm Setup of Π^{CBE} for $\mathcal{A}_1$. This is done by $\mathcal{B}$ setting SK_{IBE} to be master-key of Π^{CL} and params of Π^{CBE} to be params of Π^{CL}. Now $\mathcal{B}$ forwards params to $\mathcal{A}_1$.

As we shall see, all $\mathcal{A}_1$ queries can be handled by $\mathcal{B}$ with help from $\mathcal{C}$. In $\mathcal{A}_1$'s environment string of the form $\mathsf{params}\|\tau\|\lambda\|PK_{PKE}$ are used in certification queries and decryption queries. Adversary $\mathcal{B}$ can translate such strings $\mathsf{params}\|\tau\|\lambda\|PK_{PKE}$ into identifier strings for the certificateless scheme. Notice that in this setting, each public key PK_{PKE} has a unique identifier string, and the same public key PK_{PKE} can have multiple identifier strings.

Phase 1: Now $\mathcal{A}_1$ launches Phase 1 of its attack. Requests made by $\mathcal{A}_1$ are answered by $\mathcal{B}$ as follows:

- All certification queries (which will include PK_{PKE}) are answered by $\mathcal{B}$. On certification query $\langle \tau_i, \lambda_i, PK_{PKE,i}, SK_{PKE,i} \rangle$, adversary $\mathcal{B}$ checks that $\lambda \in \Lambda$ and that $SK_{PKE,i}$ is the private key corresponding to $PK_{PKE,i}$. If so $\mathcal{B}$ sends $\mathsf{ID}_i = \mathsf{params} \| \tau_i \| \lambda_i \| PK_{PKE,i}$ and $P_i = PK_{PKE,i}$ to $\mathcal{C}$ in a replace public key query. Then $\mathcal{B}$ forwards $\mathsf{ID}_i = \mathsf{params} \| \tau_i \| \lambda_i \| PK_{PKE,i}$ to $\mathcal{C}$ in a partial private key extraction query. $\mathcal{C}$ responds with a partial private key D_i. Adversary $\mathcal{B}$ forwards D_i to $\mathcal{A}_1$ as the certification response Cert_{τ_i}.

  Otherwise $\mathcal{B}$ returns $\perp$.

- All decryption queries are answered by $\mathcal{B}$. On decryption query

$$\langle \tau_i, \lambda_i, PK_{PKE,i}, SK_{PKE,i}, C_i \rangle,$$

  adversary $\mathcal{B}$ checks that $\lambda \in \Lambda$ and $SK_{PKE,i}$ is the private key corresponding to $PK_{PKE,i}$. If these test fail, then $\mathcal{B}$ returns $\perp$. Otherwise, $\mathcal{B}$ performs the following steps:

  1. first $\mathcal{B}$ sends $\mathsf{ID}_i = \mathsf{params} \| \tau_i \| \lambda_i \| PK_{PKE,i}$ and $P_i = PK_{PKE,i}$ to $\mathcal{C}$ in a replace public key query. Then $\mathcal{B}$ forwards $\mathsf{ID}_i = \mathsf{params} \| \tau_i \| \lambda_i \| PK_{PKE,i}$ to $\mathcal{C}$ in a partial private key extraction query. $\mathcal{C}$ responds with a partial private key D_i.
  2. $\mathcal{B}$ performs this decryption himself. $\mathcal{B}$ sets x_i of Π^{CL} to be $SK_{PKE,i}$. Adversary $\mathcal{B}$ computes and forwards the output of $\mathcal{D}^{\mathrm{CL}}(C_i, \langle x_i, D_i \rangle)$ to $\mathcal{A}_1$ as the decryption response for $\mathcal{D}^{\mathrm{CBE}}_{\mathrm{Cert}_{\tau_i}, SK_{PKE,i}, \lambda_i}(C_i)$.

  After Phase 2, we will describe how one can remove the need of $SK_{PKE,i}$ in all these queries.

Challenge: At some point, $\mathcal{A}_1$ should decide to end Phase 1 and picks a challenge string of the form:

$$\langle \tau_{\mathrm{ch}}, \lambda_{\mathrm{ch}}, PK_{PKE,\mathrm{ch}}, SK_{PKE,\mathrm{ch}}, M_0, M_1 \rangle.$$

$\mathcal{B}$ checks that $\lambda_{\mathrm{ch}} \in \Lambda$ and $SK_{PKE,\mathrm{ch}}$ is the private key corresponding to $PK_{PKE,\mathrm{ch}}$. $\mathcal{B}$ also checks that $\langle \tau_{\mathrm{ch}}, \lambda_{\mathrm{ch}}, PK_{PKE,\mathrm{ch}}, SK_{PKE,\mathrm{ch}} \rangle$ is not the subject of a valid certification query. If any check fails it returns $\perp$. Now $\mathcal{B}$ checks that the partial private

key for $\mathsf{ID}_{\text{ch}} = \mathsf{params}\|\tau_{\text{ch}}\|\lambda_{\text{ch}}\|PK_{PKE,\text{ch}}$ has not been extracted, if it has, then $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ sends $\mathsf{ID}_{\text{ch}} = \mathsf{params}\|\tau_{\text{ch}}\|\lambda_{\text{ch}}\|PK_{PKE,\text{ch}}$ and $P_{\text{ch}} = PK_{PKE,\text{ch}}$ to $\mathcal{C}$ in a replace public key query. Then $\mathcal{B}$ forwards $\langle \mathsf{ID}_{\text{ch}}, P_{\text{ch}}, M_0, M_1\rangle$ to $\mathcal{C}$ in a challenge query. The challenger $\mathcal{C}$ chooses a random bit b and responds with the challenge ciphertext $C' = \mathcal{E}^{\text{CL}}(M_b, P_{\text{ch}}, \mathsf{ID}_{\text{ch}})$. Algorithm $\mathcal{B}$ sets $C^* = C'$ and delivers C^* to $\mathcal{A}_1$ as the challenge ciphertext. It is easy to see that C^* is the CBE encrypion of M_b for $\langle \tau_{\text{ch}}, \lambda_{\text{ch}}, PK_{IBE,\text{ch}}, PK_{PKE,\text{ch}}\rangle$.

Phase 2: $\mathcal{B}$ continues to respond to requests in the same way as it did in Phase 1. We now restrict $\mathcal{A}_1$ to not make a certification query on τ_{ch}, λ_{ch} and $PK_{PKE,\text{ch}}$. We also restrict $\mathcal{A}_1$ to not make a decryption query which relays a ciphtertext equal to $C' = C^*$ with ID_{ch} and P_{ch} to the challenger. $\mathcal{B}$ aborts if needs to relay to $\mathcal{C}$ either the extract partial private key query for ID_{ch} or the decryption query for $\langle C', \mathsf{ID}_{\text{ch}}, P_{\text{ch}}\rangle$.

The value $SK_{PKE,i}$ can be ommited from certification queries, decryption queries and challenge queries. When it is omitted $\mathcal{B}$ must (i) check whether $PK_{PKE,i}$ is valid because the CBE encryption algorithm does not verify public keys and (ii) change step (2) of decryption as follows: $\mathcal{B}$ forwards C_i and $\mathsf{ID}_i = \mathsf{params}\|\tau_i\|\lambda_i\|PK_{PKE,i}$ to $\mathcal{C}$ in a decryption query. $\mathcal{C}$ responds with a message output from a decryption of the form $\mathcal{D}^{\text{CL}}(C_i, \langle x_i, D_i\rangle)$. Adversary $\mathcal{B}$ forwards the message output by $\mathcal{C}$ to $\mathcal{A}_1$ as the decryption response for $\mathcal{D}^{\text{CBE}}_{\text{Cert}_{\tau_i}, SK_{PKE,i}, \lambda_i}(C_i)$.

Guess: Eventually, $\mathcal{A}_I$ should make a guess b' for b. The $\mathcal{B}$ outputs b' as its guess for b.

Analysis: We now analyse the behaviour of $\mathcal{B}$ and $\mathcal{A}_1$ in this simulation. We claim that if algorithm $\mathcal{B}$ does not abort during the simulation then algorithm $\mathcal{A}_1$'s view is identical to its view in the real attack. Moreover, if $\mathcal{B}$ does not abort then $2|\Pr[b = b'] - \frac{1}{2}| \geq \epsilon$.

We justify this claim as follows. $\mathcal{B}$'s responses to decryption and certification queries are as in the real attack. All responses to $\mathcal{A}_1$'s queries are valid, provided of course

that $\mathcal{B}$ does not abort. Furthermore, the challenge ciphertext C^* is a valid Π^{CBE} encryption of M_b where $b \in \{0,1\}$ is random. Thus, by definition of algorithm $\mathcal{A}_1$ we have that $2|\Pr[b = b'] - \frac{1}{2}| \geq \epsilon$.

The probability that $\mathcal{B}$ does not abort during the simulation remains to be calculated. Examining the simulation, we see that $\mathcal{B}$ can abort for two reasons: (i) because $\mathcal{B}$ both replaced the public key and extracted the partial private key for entity ID_{ch} at some point, or (ii) because $\mathcal{B}$ relayed a decryption query on C' to $\mathcal{C}$ for the combination ID_{ch} and P_{ch} in Phase 2.

The first event happens only if $\mathcal{A}_1$ performs a certification queries on the combination $\langle \tau_{\text{ch}}, \lambda_{\text{ch}}, PK_{PKE,\text{ch}}, SK_{PKE,\text{ch}} \rangle$. Because only then would $\mathcal{B}$ replace the public key and extract the partial private key for entity $\mathsf{ID}_{\text{ch}} = \mathsf{params} \| \tau_{\text{ch}} \| \lambda_{\text{ch}} \| PK_{PKE,\text{ch}}$. This is exactly the certification query on which $\mathcal{A}_1$ is forbidden from making. So this event never occurs in $\mathcal{B}$'s simulation. Now let us examine the last event. Because of the way that $\mathcal{B}$ relays ciphertexts, this last event happens only if $\mathcal{A}_1$ queries $\mathcal{B}$ on the combination $\langle \tau_{\text{ch}}, \lambda_{\text{ch}}, PK_{PKE,\text{ch}}, SK_{PKE,\text{ch}}, C^* \rangle$ in Phase 2. However, this is exactly $\mathcal{A}_1$'s challenge ciphertext on which $\mathcal{A}_1$ is forbidden from making a decryption query. So this event never occurs in $\mathcal{B}$'s simulation.

Algorithm $\mathcal{B}$ never relays a forbidden query to $\mathcal{C}$ and so never aborts during the simulation. Algorithm $\mathcal{B}$ produces a perfect simulation of Game 1 for CBE and so by definition of algorithm $\mathcal{A}_1$, it would output $b' = b$ with probability at least ϵ. Thus, since the CL-PKE scheme is secure against Type I adversaries, the CBE scheme is secure against Game 1 adversaries. This completes the proof for the Game 1 adversary.

We showed how a CBE Game 1 adversary is related to a CL-PKE Type I adversary. We can also show how a CBE Game 2 adversary is related to a CL-PKE Type II adversary. This can easily be achieved by modifying Game 2 by giving the adversary the master-key, instead of allowing the adversary to pick a master-key (the motivation for this modification is given in item (4) in p.184). We omit the routine details of

this modification and the security proof which results.

□

This result is valuable because it shows that all IND-CCA secure CL-PKE schemes can be modified as specified in Section 7.4.1 to create IND-CBE-CCA secure CBE schemes. Therefore, FullCL-PKE and FullCL-PKE2 (of Chapter 8) can be altered to create IND-CBE-CCA secure CBE schemes. The Scheme FullCL-PKE when modified to form a CBE scheme is more efficient than the scheme FullCBE of [76]. The encryption scheme of FullCL-PKE requires only one pairing computation instead of two and it does not require a multiplication in $\mathbb{G}_2$. Recall that $\mathbb{G}_2$ is a subgroup of a large finite field, hence, both the pairing evaluations and computations in $\mathbb{G}_2$ are expensive. The encryption scheme FullCL-PKE replaces these expensive operations with a multiplication in $\mathbb{G}_1$ and an XOR operation.

7.5 Summary

Three generic transformations, which may generate IND-CCA secure CL-PKE schemes using IND-CCA secure ID-PKE and IND-CCA secure standard PKE schemes have been discussed. The tool required to produce a semantically secure generic CL-PKE scheme under an IND-CCA adversary is a general knowledge extractor, whose foundations remain the subject of our ongoing research. Thus, proving the security of these generic constructions remains an open problem.

We analysed CBE [76] and demonstrated several of its weaknesses. More importantly, we provided an alternative method to construct IND-CCA secure CBE schemes. The alternative method makes use of only IND-CCA secure CL-PKE schemes. This is advantageous for both CBE and CL-PKE. On one hand it is beneficial for CBE because a wider range of CBE schemes can be easily constructed from existing CL-PKE schemes. On the other hand, it is beneficial for CL-PKE because CL-PKE algorithms are naturally interoperable with CBE algorithms and because it demonstrates how to extend the potential use of CL-PKE to produce schemes suitable for streamlining certificate-based environments. A major benefit of our model is that it is robust

enough to remain applicable to other models, such as CBE [76].

Chapter 8

Further CL-PKC Schemes

Contents

In this chapter other certificateless public key cryptography (CL-PKC) schemes including a certificateless public key encryption (CL-PKE) scheme whose security rests on the Generalised Bilinear Diffie-Hellman Problem (GBDHP) are demonstrated. The CL-PKE scheme is developed to demonstrates how certificateless hierarchical and proxy schemes can be supported. In addition, we show a certificateless public key signature (CL-PKS) scheme and a certificateless authenticated key agreement protocol. As with Chapters 5 and 6, our certificateless schemes are all built from bilinear maps on groups.

8.1 Introduction

In this chapter we present an alternative CL-PKE scheme, originally published in [7]. The scheme is obtained by modifying the ID-PKE scheme of Boneh and Franklin [32]. We have seen in Chapter 6 how an appropriate adversarial model with proofs is provided to ensure that the full scheme is secure in the face of both an IND-CCA adversary who replaces public keys, and an IND-CCA adversary who has access to the master key but cannot replace public keys. The detailed development of this model in Chapters 5 and 6 provides some insight into the adaptations required to existing models that are needed to produce adversarial models for the other certificateless primitives in this chapter. Note that the aim of this chapter is to demonstrate other CL-PKE schemes and not to provide security proofs.

We will sketch a number of other CL-PKC primitives: a signature scheme based on the identity-based scheme of [83]; a key exchange protocol which improves on the security offered by the schemes of [50, 142]; a hierarchical scheme based on [77] and proxy encryption schemes. All these schemes can be easily implemented in conjunction with existing ID-PKC schemes. They use a structured public key and have desirable security properies when compared with their ID-PKC scheme counterparts.

The schemes in this chapter share structured public keys and many parameters in the set up procedure. The structure of a public key is used to construct Diffie-Hellman tuples. The tuples are verified using two pairing computations because the public key comprises of two elements in a gap group, where the DDHP is easy and the CDHP is hard. Next we will study the properties of this set up.

8.2 A General Set Up for CL-PKC

In this chapter, all the CL-PKC schemes we describe can share many parameters in their set up procedures. Our first scheme, BasicCL-PKE2, is a CL-PKE scheme which is analogous to the scheme BasicIdent of [32], and is included to help understand the setting for the remaining schemes in this chapter. The full scheme which we build using BasicCL-PKE2, is in turn an analogue of the scheme FullIdent of [32] and is IND-CCA secure in the model of Chapter 6, assuming the hardness of the GBDHP.

The full CL-PKE scheme, FullCL-PKE2, which corresponds to BasicCL-PKE2, is in many ways superseeded by the scheme presented in Section 6.3. It is superseeded in the following ways:

1. The number of pairing computations is higher in FullCL-PKE2 because we check the structure of the public key.
2. The security of FullCL-PKE2 is based on the hardness of the GBDHP (not the harder and more studied BDHP).
3. Public keys consist of two points on the elliptic curve (not one as is the case with FullCL-PKE).

The public keys in the general set up that we describe below are for a specific KGC. Therefore, the CL-PKE schemes in this chapter do not allow the encryptor to choose which KGC will authenticate the decryptor. This allows the decryptor to mandate a particular centralised point of control (that is, a particular KGC) for authentication. The merits and shortcomings of this property were discussed in Section 4.6.2. This same propery holds true for the other schemes in this chapter which share a similar set up. The algorithms of the CL-PKE schemes BasicCL-PKE2 and FullCL-PKE2 share much in common with the ID-PKE scheme of [32] when compared to the schemes of Chapters 5 and 6. This can be advantageous for deployment, since a single infrastructure can be used to support both identifier based and certificateless schemes.

8.3 CL-PKE Schemes

In this section, we describe a pair of CL-PKE schemes BasicCL-PKE2 and FullCL-PKE2.

8.3.1 A Basic CL-PKE Scheme

We describe the seven algorithms needed to define BasicCL-PKE2. As before, we let k be a security parameter given to the Setup algorithm and $\mathcal{IG}$ be a BDH parameter generator with input k.

Setup: This algorithm runs as follows:

1. Run $\mathcal{IG}$ on input k to generate output $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ where $\mathbb{G}_1$ and $\mathbb{G}_2$ are groups of some prime order q and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a pairing.
2. Choose an arbitrary generator $P \in \mathbb{G}_1$.
3. Select a master-key s uniformly at random from $\mathbb{Z}_q^*$ and set $P_0 = sP$.
4. Choose cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1^*$ and $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^n$. Here n will be the bit-length of plaintexts.

The system parameters are params$= \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_0, H_1, H_2 \rangle$. The master-key is $s \in \mathbb{Z}_q^*$. The message space is $\mathcal{M} = \{0,1\}^n$ and the ciphertext space is $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^n$.

Partial-Private-Key-Extract: This algorithm takes as input an identifier $\mathsf{ID}_A \in \{0,1\}^*$, and carries out the following steps to construct the partial private key for entity A with identifier ID_A:

1. Compute $Q_A = H_1(\mathsf{ID}_A) \in \mathbb{G}_1^*$.

2. Output the partial private key $D_A = sQ_A \in \mathbb{G}_1^*$.

The reader will notice that the partial private key of entity A here is identical to that entity's private key in the schemes of [32] and the partial private key of the schemes BasicCL-PKE and FullCL-PKE in Sections 5.4 and 6.3. As for those schemes, the correctness of the Partial-Private-Key-Extract algorithm output can be verified by checking $\hat{e}(D_A, P) = \hat{e}(Q_A, P_0)$.

Set-Secret-Value: This algorithm takes as inputs params and an entity A's identifier ID_A. It selects $x_A \in \mathbb{Z}_q^*$ at random and outputs x_A as A's secret value.

Set-Private-Key: This algorithm takes as inputs params, an entity A's partial private key D_A and A's secret value $x_A \in \mathbb{Z}_q^*$. It transforms partial private key D_A to private key S_A by computing $S_A = x_A D_A = x_A s Q_A \in \mathbb{G}_1^*$.

Set-Public-Key: This algorithm takes params and entity A's secret value $x_A \in \mathbb{Z}_q^*$ as inputs and constructs A's public key as $P_A = \langle X_A, Y_A \rangle$, where $X_A = x_A P$ and $Y_A = x_A P_0 = x_A s P$.

Encrypt: To encrypt $M \in \mathcal{M}$ for entity A with identifier $\mathsf{ID}_A \in \{0,1\}^*$ and public key $P_A = \langle X_A, Y_A \rangle$, perform the following steps:

1. Check that $X_A, Y_A \in \mathbb{G}_1^*$ and that the equality $\hat{e}(X_A, P_0) = \hat{e}(Y_A, P)$ holds. If not, output $\perp$ and abort encryption.
2. Compute $Q_A = H_1(\mathsf{ID}_A) \in \mathbb{G}_1^*$.
3. Choose a random value $r \in \mathbb{Z}_q^*$.
4. Compute and output the ciphertext:

$$C = \langle rP, M \oplus H_2(\hat{e}(Q_A, Y_A)^r) \rangle.$$

Notice that this encryption operation is identical to the encryption algorithm of the

ID-PKE scheme in Section 3.2.4.1 (the scheme BasicIdent of [32]), except for the check on the structure of the public key in step 1 and the use of Y_A in place of P_0 in step 4.

Decrypt: Suppose $C = \langle U, V \rangle \in \mathcal{C}$. To decrypt this ciphertext using the private key S_A, compute and output:

$$V \oplus H_2(\hat{e}(S_A, U)).$$

Notice that if $\langle U = rP, V \rangle$ is the encryption of M for entity A with public key $P_A = \langle X_A, Y_A \rangle$, then we have:

$$\begin{aligned} V \oplus H_2(\hat{e}(S_A, U)) &= V \oplus H_2(\hat{e}(x_A s Q_A, rP)) \\ &= V \oplus H_2(\hat{e}(Q_A, x_A s P)^r) \\ &= V \oplus H_2(\hat{e}(Q_A, Y_A)^r) \\ &= M. \end{aligned}$$

Thus decryption is the inverse of encryption. Again, the similarity to the decryption operation of BasicIdent should now be apparent. This completes our description of BasicCL-PKE2.

We have presented this scheme to help the reader understand our remaining schemes, and in particular the next scheme and teh schemes in Section 8.6 and 8.7. We do not analyse its security in detail. It can be shown that BasicCL-PKE2 is secure in the OWE model of Section 5.3.

8.3.2 A Full CL-PKE Scheme

Now that we have described our basic CL-PKE scheme, we add chosen ciphertext security to it, adapting the Fujisaki-Okamoto hybridisation technique described in Section 6.4. The reader should now be very familiar with this adaptation, as it is similar to the adaptation which generates the scheme FullCL-PKE from BasicCL-PKE in Chapter 6. The algorithms for FullCL-PKE2 are as follows:

Setup: Identical to Setup for BasicCL-PKE2, except that we choose two additional cryptographic hash functions $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$ and $H_4 : \{0,1\}^n \to \{0,1\}^n$.

Now the system parameters are params$= \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_0, H_1, H_2, H_3, H_4 \rangle$. The master-key and message space $\mathcal{M}$ are the same as in BasicCL-PKE2. The ciphertext space is now $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^{2n}$.

Partial-Private-Key-Extract: Identical to BasicCL-PKE2.

Set-Secret-Value: Identical to BasicCL-PKE2.

Set-Private-Key: Identical to BasicCL-PKE2.

Set-Public-Key: Identical to BasicCL-PKE2.

Encrypt: To encrypt $M \in \mathcal{M}$ for entity A with identifier $\mathsf{ID}_A \in \{0,1\}^*$ and public key $P_A = \langle X_A, Y_A \rangle$, perform the following steps:

1. Check that $X_A, Y_A \in \mathbb{G}_1^*$ and that the equality $\hat{e}(X_A, P_0) = \hat{e}(Y_A, P)$ holds. If not, output $\perp$ and abort encryption.
2. Compute $Q_A = H_1(\mathsf{ID}_A) \in \mathbb{G}_1^*$.
3. Choose a random $\sigma \in \{0,1\}^n$.
4. Set $r = H_3(\sigma, M)$.
5. Compute and output the ciphertext:

$$C = \langle rP, \sigma \oplus H_2(\hat{e}(Q_A, Y_A)^r), M \oplus H_4(\sigma) \rangle.$$

Decrypt: Suppose $C = \langle U, V, W \rangle \in \mathcal{C}$. To decrypt this ciphertext using the private key S_A:

1. Compute $V \oplus H_2(\hat{e}(S_A, U)) = \sigma'$.

2. Compute $W \oplus H_4(\sigma') = M'$.
3. Set $r' = H_3(\sigma', M')$ and test if $U = r'P$. If not, output $\perp$ and reject the ciphertext.
4. Output M' as the decryption of C.

When C is a valid encryption of M using P_A and ID_A, it is easy to see that decrypting C will result in an output $M' = M$. This concludes the description of FullCL-PKE2.

We have the following result about the security of FullCL-PKE2 from [7, Theorem 1].

Result 8.1 Let hash functions H_1, H_2, H_3 and H_4 be random oracles. Suppose further that there is no polynomially bounded algorithm that can solve the GBDHP in groups generated by $\mathcal{IG}$ with non-negligible advantage. Then FullCL-PKE2 is IND-CCA secure.

This security result relies on the hardness of the GBDHP and assumes H_1 and H_2 are random oracles. In essence, the detailed analysis shows that security against Type II adversaries can be reduced to the difficulty of computing the value $\hat{e}(Q_A, x_A sP)^r$. Given that a Type II adversary has s but not x_A, this is equivalent to the BDHP on input $\langle P, Q_A, U, X_A \rangle$. Likewise, security against a Type I adversary who does not know s but who might replace Y_A by a new value Y'_A can be reduced to the GBDHP on input $\langle P, Q_A = aP, U = rP, P_0 = sP \rangle$, with solution Y'_A, $\hat{e}(P, Y'_A)^{sra}$. For more details see [6, 7].

8.4 A Certificateless Signature Scheme

We will describe a certificateless public-key signature (CL-PKS) scheme that is based on a provably secure ID-PKC signature scheme of [83]. Note that we have not developed a security model for CL-PKS, and we do not prove our scheme to be secure.

In general, a CL-PKS scheme can be specified by seven algorithms:
Setup, Partial-Private-Key-Extract, Set-Secret-Value, Set-Private-Key, Set-Public-Key, Sign and Verify. These are similar to the algorithms used to define a CL-PKE scheme: Setup and params are modified to include a description of the signature space $\mathcal{S}$, Partial-Private-Key-Extract, Set-Secret-Value, Set-Private-Key and Set-Public-Key are just as before and Sign and Verify are as follows:

Sign (Σ^{CL}): This algorithm takes as inputs params, a message $M \in \mathcal{M}$ to be signed and a private key S_A. It outputs a signature $Sig \in \mathcal{S}$. We write $Sig \leftarrow \Sigma^{\mathrm{CL}}(M, S_A)$.

Verify ($\mathcal{V}^{\mathrm{CL}}$): This algorithm takes as inputs params, a message $M \in \mathcal{M}$, the identifier ID_A and public key P_A of an entity A, and $Sig \in \mathcal{S}$ as the signature to be verified. It outputs valid, invalid or $\perp$. We write $\{\mathsf{valid}, \mathsf{invalid}, \perp\} \leftarrow \mathcal{V}^{\mathrm{CL}}(M, Sig, P_A, \mathsf{ID}_A)$.

Given this general description, we now define a CL-PKS scheme using a similar basic set up procedure as the schemes BasicCL-PKE2 and FullCL-PKE2.

Setup: This is identical to Setup for our scheme BasicCL-PKE2, except that now we have hash function $H : \{0,1\}^* \times \mathbb{G}_2 \rightarrow \mathbb{Z}_q^*$ instead of H_2, hence the params is $\langle \mathbb{G}_1, \mathbb{G}_2, n, \hat{e}, P, P_0, H_1, H \rangle$. The signature space is defined as $\mathcal{S} = \mathbb{G}_1 \times \mathbb{Z}_q^*$.

Partial-Private-Key-Extract: Identical to BasicCL-PKE2.

Set-Secret-Value: Identical to BasicCL-PKE2.

Set-Private-Key: Identical to BasicCL-PKE2.

Set-Public-Key: Identical to BasicCL-PKE2.

Sign: To sign $M \in \mathcal{M}$ using the private key S_A, perform the following steps:

1. Choose a random value $a \in \mathbb{Z}_q^*$.

2. Compute $r = \hat{e}(aP, P) \in \mathbb{G}_2$.
3. Set $v = H(M, r) \in \mathbb{Z}_q^*$.
4. Compute $U = vS_A + aP \in \mathbb{G}_1$.
5. Output as the signature $\langle U, v \rangle \in \mathcal{S}$.

Verify: To verify a purported signature $\langle U, v \rangle$ on a message $M \in \mathcal{M}$ for identity ID_A and public key $\langle X_A, Y_A \rangle$:

1. Check that the equality $\hat{e}(X_A, P_0) = \hat{e}(Y_A, P)$ holds. If not, output $\perp$ and abort verification.
2. Compute $r = \hat{e}(U, P) \cdot \hat{e}(Q_A, -Y_A)^v$.
3. Check if $v = H(M, r)$ holds. If it does, output valid, otherwise output invalid.

For a valid signature, it is easy to check that the Verify algorithm will output valid. This completes the description of the CL-PKS scheme.

This scheme is related to the first scheme in [83, p.312], which is secure against existential forgery in the random oracle model. Our verification operation is identical to the verification algorithm in [83], except for the check on the structure of the public key in step 1 and the use of Y_A in place of P_0 in step 2. Our signature algorithm, however, is identical to that in [83].

8.5 A Certificateless Authenticated Key Agreement Protocol

A number of identity-based two party key-agreement protocols have been described [50, 142]. All the session keys created in Smart's protocol [142] can trivially be recovered by the TA. The protocol of [142] was later modified by Chen and Kudla

[50] to eliminate this escrow capability. However, the TA in the protocol of [50] can still perform a standard man-in-the-middle attack by replacing one short-term value with a value of its choice, and can thus impersonate any entity in an undetectable way.

Here we introduce a certificateless key agreement protocol which is only vulnerable to such a man-in-the-middle attack if, in addition to replacing a short-term value, a user-specific long-term public key is also replaced. If keys are produced using our binding technique (that is, CL-PKC (B)), then such a man-in-the-middle attack mounted by the KGC will leave evidence exposing the KGC's actions. The evidence is not the replaced public key but is the KGC's impersonation of a communicating entity which can only occur if a working public key (that is, an appropriate partial private key) exists.

The initialization for our certificateless key agreement scheme is formally specified using five algorithms: Setup, Partial-Private-Key-Extract, Set-Secret-Value, Set-Private-Key and Set-Public-Key. These are the same as in BasicCL-PKE2. Algorithms Setup, Set-Secret-Value and Set-Public-Key must be run by entities A and B before exchanging protocol messages. Algorithms Partial-Private-Key-Extract and Set-Private-Key can be run by entities A and B after exchanging protocol messages and are required for computing a shared key.

Entities A and B who wish to agree a key first choose random values $a, b \in \mathbb{Z}_q^*$ respectively. Given the initializations described in the previous paragraph, the protocol is as follows:

Protocol description: After the above messages are exchanged in Figure 8.1, both users check the validity of each other's public keys in the usual way. So entity A checks $\hat{e}(X_B, P_0) = \hat{e}(Y_B, P)$ and entity B checks $\hat{e}(X_A, P_0) = \hat{e}(Y_A, P)$. Then A computes $K_A = \hat{e}(Q_B, Y_B)^a \cdot \hat{e}(S_A, bP)$ and B computes $\hat{e}(Q_A, Y_A)^b \cdot \hat{e}(S_B, aP)$. It is easy to see that $K = K_A = K_B$ is a key shared between A and B; to ensure forward security, A and B can instead use the shared key $H(K\|abP)$ where H is a suitable

Protocol Messages

1. $A \rightarrow B: \quad aP \| \langle X_A, Y_A \rangle$
2. $B \rightarrow A: \quad bP \| \langle X_B, Y_B \rangle$

Figure 8.1: Certificateless authenticated key agreement protocol.

hash function.

The protocol uses only two passes and is bandwidth-efficient. Bandwidth usage can be reduced further if the same entities agree many keys: then transmission of only fresh aP, bP is needed in each protocol run. Each side computes four pairings; this can be reduced to one pairing each if the same entities agree many keys. The protocol is therefore competitive with those of [50, 142]. Key confirmation can be added with extra protocol passes. In a CL-PKC (B) setting, key confirmation creates the evidence required to implicate a cheating KGC.

The key generation method in our protocol is based on the protocol $1'$ in [50] – a modification which adds forward secrecy to Smart's protocol [142]. Informally, even if the public key of entity A is replaced with $\langle X'_A, Y'_A \rangle$, an adverary cannot impersonate entity A because the shared session key cannot be computed without an appropriate partial private key, D_A.

8.6 Hierarchical CL-PKE

Recall that in the survey of Section 3.6, we discussed how Gentry and Silverberg [77] improved the work of [85] by introducing a totally collusion-resistant, hierarchical, ID-based infrastructure for encryption and signatures. Such an infrastructure spreads the workload of master servers and produces levels which can be used to support

short lived keys, for example. However, the hierarchical schemes of [77] still have an undesirable escrow property. Here, we adapt the hierarchical encryption scheme of [77] to our certificateless setting and eliminate key escrow to a certain extent.

In general, a hierarchical CL-PKE (HCL-PKE) scheme has a root KGC and a hierarchy of entities. Each entity other than the KGC is associated with a level $t \geq 1$ in the hierarchy and with a string ID-tuple which identifies that entity's ancestors in the hierarchy. The ID-tuple string for an entity at level t with identity ID_t is $\langle \mathsf{ID}_1, \mathsf{ID}_2, \ldots, \mathsf{ID}_t \rangle$. An HCL-PKE scheme is specified by seven algorithms: Setup, Partial-Private-Key-Extract, Set-Secret-Value, Set-Private-Key, Set-Public-Key, Encrypt and Decrypt. Rather than outline the general function of each algorithm, we present a concrete scheme, BasicHCL-PKE, whose description should make the general operation of an HCL-PKE scheme clear. The algorithms for BasicHCL-PKE are as follows.

Setup: This algorithm is identical to Setup for BasicCL-PKE2, except that now the ciphertext space for a level t ciphertext is $\mathcal{C}_t = \mathbb{G}_1^t \times \{0,1\}^n$. The system parameters are params$= \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_0, H_1, H_2 \rangle$. For ease of presentation, we denote the master-key by x_0 instead of s (so we have $P_0 = x_0 P$).

Partial-Private-Key-Extract: This algorithm is usually executed by a level $t-1$ entity ID_{t-1} for a child entity ID_t at level t. When $t = 1$, this algorithm is executed by the root KGC for ID_1. It takes as input the ID-tuple $\langle \mathsf{ID}_1, \mathsf{ID}_2, \ldots, \mathsf{ID}_t \rangle$ and carries out the following steps to construct the partial private key for ID_t:

1. Compute $Q_t = H_1(\mathsf{ID}_1 \| \mathsf{ID}_2 \| \ldots \| \mathsf{ID}_t) \in \mathbb{G}_1^*$.
2. Output ID_t's partial private key D_t for $t \geq 2$ where

$$D_t = D_{t-1} + x_{t-1} Q_t = \sum_{i=1}^{t} x_{i-1} Q_i.$$

If $t = 1$, then output $D_1 = x_0 Q_1$. The key D_t must be transported to ID_t over a confidential and authentic channel.

Set-Secret-value: This algorithm takes as inputs params and level t entity's ID-tuple $\langle \mathsf{ID}_1, \mathsf{ID}_2, \ldots, \mathsf{ID}_t \rangle$ as inputs. It selects $x_t \in \mathbb{Z}_q^*$ at random and outputs x_t as ID_t's secret value.

Set-Private-Key: As for BasicCL-PKE2, except that the private key for ID_t is denoted by S_t. So $S_t = x_t D_t$.

Set-Public-Key: As for BasicCL-PKE2, except that the public key for ID_t is denoted by $P_t = \langle X_t, Y_t \rangle$. So $Y_t = x_0 X_t = x_0 x_t P$.

Encryption: To encrypt $M \in \mathcal{M}$ for identity ID_t at level $t \geq 1$ with ID-tuple $\langle \mathsf{ID}_1, \mathsf{ID}_2, \ldots, \mathsf{ID}_t \rangle$, perform the following steps:

1. For each $1 \leq i \leq t$, check that the equality $\hat{e}(X_i, P_0) = \hat{e}(Y_i, P)$ holds. If any check fails, output $\perp$ and abort encryption.
2. Compute $Q_i = H_1(\mathsf{ID}_1 \| \mathsf{ID}_2 \| \ldots \| \mathsf{ID}_i) \in \mathbb{G}_1^*$ for each $2 \leq i \leq t$.
3. Choose a random $r \in \mathbb{Z}_q^*$.
4. Compute and output the ciphertext:

$$C = \langle U_0, U_2, \ldots, U_t, V \rangle = \langle rP, rQ_2, rQ_3, \ldots, rQ_t, M \oplus H_2(\hat{e}(Q_1, Y_t)^r) \rangle \in \mathcal{C}_t.$$

Notice that to encrypt a message for a level t entity ID_t, the values Q_i and hence identities ID_i of all the ancestors of ID_t are needed. Moreover, to perform the checking in Step 1 ,all the public keys of these entities are also needed.

Decryption: Suppose $C = \langle U_0, U_2, \ldots, U_t, V \rangle \in \mathcal{C}_t$ is a BasicHCL-PKE ciphertext for a level t entity with ID-tuple $\langle \mathsf{ID}_1, \mathsf{ID}_2, \ldots \mathsf{ID}_t \rangle$. Let the public keys of ID_i's ancestors be $P_i = \langle X_i, Y_i \rangle$ $(1 \leq i < t)$. Then to decrypt the ciphertext C using the private key S_t, compute and output:

$$V \oplus H_2 \left(\frac{\hat{e}(S_t, U_0)}{\prod_{i=2}^{t} \hat{e}(x_t X_{i-1}, U_i)} \right).$$

Using properties of the bilinear map $\hat{e}$, we have:

$$\begin{aligned}
\frac{\hat{e}(S_t,U_0)}{\prod_{i=2}^{t}\hat{e}(x_t X_{i-1},U_i)} &= \hat{e}(x_t\sum_{i=1}^{t} x_{i-1}Q_i, rP)\cdot\prod_{i=2}^{t}\hat{e}(x_t x_{i-1}P, rQ_i)^{-1}\\
&= \hat{e}(x_t\sum_{i=1}^{t} x_{i-1}Q_i, rP)\cdot\hat{e}(-\sum_{i=2}^{t} x_t x_{i-1}Q_i, rP)\\
&= \hat{e}(x_t x_0 Q_1, rP)\\
&= \hat{e}(Q_1, x_t x_0 P)^r\\
&= \hat{e}(Q_1, Y_t)^r
\end{aligned}$$

so that decryption is the inverse of encryption.

This completes our description of BasicHCL-PKE. Using the hybridization technique described in Section 6.4, it is straightforward to adapt this scheme to obtain a scheme that is based on the identifier-based hierarchical scheme which is secure against fully-adaptive chosen ciphertext attackers [77, §3.2]. We must assume here that no ancestor ID_u of our level t entity ID_t replaces the public key of ID_t. Even with the extra binding step in place, our hierarchical schemes do not offer a true equivalent of trust level 3: although it is then possible to detect that a public key has been replaced by an ancestor, it is not possible to pinpoint exactly which ancestor is responsible. The attack by an ancestor at level u runs as follows:

1. An ancestor ID_u where $0 \leq u \leq t-1$, where the root KGC has identifier ID_0, selects $x'_u \in \mathbb{Z}_q^*$ and replaces the public key P_t of ID_t at level t with $P'_t = \langle X'_t, Y'_t\rangle = \langle x'_u P, x'_u sP\rangle$. Notice that the new public key satisfies the usual structural check.

2. Encryption to entity ID_t yields a ciphertext of the form:

$$C = \langle rP, rQ_2, rQ_3, \ldots, rQ_t, M \oplus H_2(\hat{e}(Q_1, Y'_t)^r)\rangle \in \mathcal{C}_t.$$

3. If $2 \leq u \leq t-1$, then ID_u can decrypt this ciphertext by computing:

$$V \oplus H_2\left(\frac{\hat{e}(x'_u D_u, U_0)}{\prod_{i=2}^{u}\hat{e}(x'_u X_{i-1}, U_i)}\right).$$

   If $u = 0$ or $u = 1$, then ID_u can decrypt this ciphertext by computing:

$$V \oplus H_2(\hat{e}(x'_u D_1, U_0)).$$

Notice that this attack works even if the public key $P_t = \langle x_t P, x_t s P \rangle$ is bound with ID_t in D_t, since the tuple $\langle U_{u+1} \dots U_t \rangle$ is not used in this attack. Furthermore, this attack works even if the ancestors public key $P_u = \langle x_u P, x_u s P \rangle$ is bound with the ID_u in D_u. Therefore, we cannot allow partial private keys to be made public in this setting as this would enable *any* adversary to mount a successful key attack by replacing the public key of ID_t. This attack is demonstrated next.

1. An adversary obtains ID_t's ancestor's partial private key D_1 or D_u where $2 \leq u \leq t-1$.

2. The adversary selects $x'_t \in \mathbb{Z}_q^*$ and replaces the public key P_t for an entity at level t with $P'_t = \langle X'_t, Y'_t \rangle = \langle x'_t P, x'_t s P \rangle$.

3. Encryption to entity ID_t yields a ciphertext of the form:

$$C = \langle rP, rQ_2, rQ_3, \dots, rQ_t, M \oplus H_2(\hat{e}(Q_1, Y'_t)^r) \rangle \in \mathcal{C}_t.$$

4. The adversary decrypts this ciphertext by computing:

$$V \oplus H_2 \left(\frac{\hat{e}(x'_t D_u, U_0)}{\prod_{i=2}^{u} \hat{e}(x'_t X_{i-1}, U_i)} \right),$$

   or

$$V \oplus H_2(\hat{e}(x'_t D_1, U_0)).$$

As with the previous attack, this attack works even if public keys are bound with identifiers in all the partial private keys. We note that an extension of the hybrid PKI/ID-PKC scheme of [48] has stronger security guarantees. However, this approach still requires certification for intermediate entities, and our primary focus is on completely certificate-free infrastructures. The attacks presented here suggests that there is still work to be done in designing HCL-PKE schemes with stronger security properties.

8.7 Proxy Decryption

We demonstrate how our HCL-PKE scheme BasicHCL-PKE supports two kinds of proxy decryption: an entity A with identifier ID_t at level $t \geq 1$ can efficiently delegate decryption to either a proxy at level $t-1$ (if $t \geq 2$) or a proxy at level $t+1$. This is an important feature because the decryption and encryption costs in our HCL-PKE scheme grow roughly linearly with t, so that an unacceptably high computational burden may be placed on entities located low in the hierarchy, if the hierarchy has many levels.

To prepare a ciphertext $C = \langle U_0, U_2, \ldots, U_t, V \rangle$ encrypting message M for proxy decryption, entity A with identifier ID_t located at level t transforms C by appending some fixed keying information and a string proxy to it to obtain a new ciphertext:

$$C_{\mathsf{proxy}} = \langle C, \langle x_t X_1, x_t X_2, \ldots, x_t X_{t-1} \rangle, \mathsf{proxy} \rangle.$$

Here, the value of proxy depends on whether decryption is being delegated to an entity at level $t-1$ or $t+1$. So we have two cases:

Proxy at level $t-1$:

1. Entity A sets $\mathsf{proxy} = \langle x_t U_0 \rangle$ and forwards C_{proxy} to level $t-1$ entity B with identifier ID_{t-1}.

2. Entity B decrypts C_{proxy} using its partial private key by computing:

$$M' = V \oplus H_2 \left(\frac{\hat{e}(D_{t-1} + x_{t-1} Q_t, x_t U_0)}{\prod_{i=2}^{t} \hat{e}(x_t X_{i-1}, U_i)} \right).$$

   Using the properties of the bilinear map $\hat{e}$, we can see that:

$$\frac{\hat{e}(D_{t-1} + x_{t-1} Q_t, x_t U_0)}{\prod_{i=2}^{t} \hat{e}(x_t X_{i-1}, U_i)} = \frac{\hat{e}(S_t, U_0)}{\prod_{i=2}^{t} \hat{e}(x_t X_{i-1}, U_i)} \quad .$$

   Hence we have $M' = M$, and the proxy at level $t-1$ can correctly decrypt.

Proxy at level $t+1$:

1. Entity A sets $\mathsf{proxy}= \langle x_t U_0, \hat{e}(x_t Q_{t+1}, x_t U_0)\rangle$ and forwards C_{proxy} to level $t+1$ entity B with identifier ID_{t+1}.

2. Entity B decrypts C_{proxy} using its partial private key by computing:

$$M' = V \oplus H_2\left(\frac{\hat{e}(D_{t+1}, x_t U_0)}{\hat{e}(x_t Q_{t+1}, x_t U_0)\cdot \prod_{i=2}^{t}\hat{e}(x_t X_{i-1}, U_i)}\right).$$

   Using the properties of the bilinear map $\hat{e}$, we can see that:

$$\frac{\hat{e}(D_{t+1}, x_t U_0)}{\hat{e}(x_t Q_{t+1}, x_t U_0)\cdot \prod_{i=2}^{t}\hat{e}(x_t X_{i-1}, U_i)} = \frac{\hat{e}(S_t, U_0)}{\prod_{i=2}^{t}\hat{e}(x_t X_{i-1}, U_i)} \quad .$$

   Hence we have $M' = M$, and the proxy at level $t-1$ can correctly decrypt.

Notice that the proxy capability that A delegates is one-time only: in each of our two cases, to perform decryption, B needs a component $x_t U_0$ that depends both on the ciphertext and on A's secret. Of course, our proxy schemes shield A's secret x_t and private key S_t from all entities, including the proxy. Notice also that the proxy ciphertext in our level $t+1$ proxy scheme contains sufficient information to allow it to be decrypted by our level $t-1$ proxy. So proxy ciphertexts produced for A's children can also be decrypted by A's parent. As the reader will have noticed, we have not presented any formal security analysis for the schemes in this chapter.

8.8 Summary

In this chapter we have rounded off our treatment of CL-PKC by briefly presenting a number of certificateless primitives: an encryption scheme, a signature scheme, a key agreement protocol, a hierarchical encryption scheme and two proxy decryption schemes. All of these schemes share many parameters in their set up procedures. Also, they eliminate key escrow from the ID-PKC schemes from which they originated. The certificateless schemes presented yield solutions applicable to many settings.

In future work we intend to develop security models and proofs for other certificateless primitives. We fully expect that certificateless versions of yet more primitives can be devised by adapting existing identity-based schemes. The importance of an efficient certificateless signature scheme cannot be understated, since in addition to offering the benefits of identity based signature schemes [135], it provides true non-repudiation without the need to store certificates with every verified signature. In certificate-based schemes, the certificates must be stored if non-repudiation is required. Therefore, a fruitful area of research may be special-purpose signature schemes [28, 34, 150] and signcryption schemes [38, 103, 109]. Naturally, we also pursue the creation hierarchical certificateless encryption schemes with better security assurances.

It would be interesting to see if the ideas of 'double encryption' presented in Chapter 6 could be extended to improve on our certificateless authentic key agreement protocol and signature scheme. Another interesting avenue for research is designing efficient CL-PKC schemes that are secure in the standard model. We are optimistic in this front; recently an efficient selective-ID[8.1] secure ID-PKE scheme and a pairing-based secure signature scheme were presented by Boneh and Boyen in [29] and [30] respectively, both schemes did not require the use of random oracles.

[8.1] In the selective-ID model, the adversary must commit in advance to the identifier that it intends to attack.

Part III

Pairing-based Key Agreement

Chapter 9

Tripartite Authenticated Key Agreement

Contents

This chapter develops certificate-based authenticated tripartite protocols. It begins by describing relevant public key protocols and goes on to propose and examine one round authenticated protocols which do not rely on digital signatures. Finally, signature based tripartite authenticated key agreement protocols with key confirmation are presented in a non-broadcast setting.

9.1 Introduction

To recapitulate Section 3.3, *asymmetric key agreement* protocols are multi-party protocols in which entities exchange public information allowing them to create a common secret key that is known only to these entities and which cannot be determined by any other party. This secret key, commonly called a *session key*, can then be used to create a confidential or integrity-protected communications channel amongst the entities. Beginning with the famous Diffie-Hellman protocol [58], a huge number of authenticated two-party key agreement protocols have been proposed (see [27] and [114, Chapter 12.6] for surveys). This reflects the fundamental nature of key agreement as a cryptographic primitive.

A situation in which three or more parties share a secret key is called *conference keying*. The three-party (or tripartite) case is of most practical importance not only because it is the most common size for electronic conferences, but because it can be used to provide a range of services for two communicating parties. For example, a third party can be added to chair or referee a conversation for the purpose of ad hoc auditing, data recovery or escrow purposes.

One of the most exciting developments in recent years in the area of key agreement is Joux's tripartite key agreement protocol using elliptic curve pairings [90]. See Section 3.3.3 for a description of Joux's protocol and Section 2.3 for an overview of

pairings on elliptic curves. Joux's protocol (just like the raw Diffie-Hellman protocol), however, is unauthenticated and suffers from man-in-the-middle attacks as described in Section 3.3.1.

9.1.1 Contributions

In this chapter it is shown how Joux's protocol [90] can be transformed into a secure tripartite protocol that still requires only a single broadcast per entity. In fact, we present four different one round, tripartite authenticated key agreement (TAK) protocols in Section 9.2. All these protocols have the same protocol messages, but different methods for calculating session keys from those messages. Our protocols are specifically designed to avoid the use of potentially expensive signature computations. It is clear how Joux's protocol can be augmented with signatures on the short-term keys so as to prevent man-in-the-middle attacks.

The reader should *not* assume that four different session keys (TAK-1 to TAK-4) are generated from one protocol run. There exist much better ways of deriving multiple session keys than this; the protocol is not four times as efficient! Rather we present four protocols and analyze their different security properties. Our one round protocols build on Joux's protocol and draw on ideas from the Unified Model [10], the Matsumoto, Takashima and Imai (MTI) protocols [110], and Menezes, Qu and Vanstone (MQV) protocols [99].

In Section 9.3, we consider proofs of security for our protocols. Our proofs use an adaptation of the Bellare-Rogaway model [22] to the public key setting. Our model is similar to that given by Blake-Wilson, Johnson and Menezes [26] (though our model extends to the three-party situation, and our extension is different to that presented in the symmetric-key setting in [24]). Our proofs show that the first of the TAK protocols is secure and has perfect forward secrecy provided that the BDHP is hard – see Section 2.4 for more details on the BDHP. Our security model has the benefits of being relatively simple to understand and allowing the

construction of straightforward proofs of security. However, just like earlier models [22, 26], it assumes perfect public key registration processes and it makes use of random oracles. Moreover, our proof places a relatively strong technical restriction on the capabilities of an adversary, namely, that he is not allowed to make any Reveal queries – see Section 9.3 for a more detailed discussion. Despite these limitations, we believe our proof to be a useful indicator of the strength of the TAK-1 protocol. We also note that the authors of earlier work [26] made the same technical restriction, and were forced to conjecture the security of their other authenticated key agreement protocol, a protocol similar to our third TAK protocol. Finally, we do not provide a proof of security for our other protocols. But we note that the MQV protocol, on which our fourth TAK protocol is modelled, has never been proven secure in any model, though it has been widely adopted for standardisation [8, 9, 87, 89]. While we believe security proofs and models to be useful, they are not our sole focus here, rather we are also interested in providing practical and secure protocols.

In view of the incomplete analysis currently available through provable approaches, we choose to supplement our proofs of security with ad hoc analysis in Section 9.5. This allows us to consider attacks on our protocols not included in the security model. In Section 9.6 we complete the treatment of one round tripartite authenticated key agreement protocols with three parties online by presenting Shim's tripartite protocol [138]. Furthermore, we show that Shim's protocol does not make mathematical sense. In Section 9.7 the scenario in which one of three parties is off-line is examined. The protocol we give for this situation can be applied to key escrow with an off-line escrow agent.

This chapter's penultimate section, Section 9.8, examines pairing-based authenticated key agreement with key confirmation in a non-broadcast setting. The main point we make in that section is that a properly authenticated and key confirmed protocol based on pairings can be no more efficient (in terms of protocol passes) than the obvious extension of the station-to-station protocol [59] to three parties. Thus, the apparent efficiency of Joux's protocol is lost when it is made secure and when an appropriate measure of efficiency is used.

The final section, Section 9.9, contains conclusions and some ideas for future work.

9.1.2 Related Work

9.1.2.1 Two Party Authenticated Key Agreement Protocols

As we have already noted, our work in this chapter builds on that of Joux [90], and our one round protocols draw upon some key agreement protocols from the Diffie-Hellman family, namely, the Unified Model [10], MTI [110] and MQV [99] protocols. Here we provide a brief introduction to these earlier protocols.

In what follows, Cert_A and Cert_B denote the certificates issued by the CA for entities A and B. Long-term public keys $K_{pub,A}$ and $K_{pub,B}$ corresponding to the long-term private keys x and y are in Cert_A and Cert_B respectively.

In the protocols in Figure 9.1, short-term private keys a and b are selected uniformly at random by A and B, respectively. In the MQV protocol the value of a and b are selected uniformly at random from $\mathbb{Z}_q^*$, whilst in the other protocols the random values are $1 \leq a, b \leq p-2$, where p and q are large primes. In the MQV protocol the generator point $P \in E(\mathbb{F}_t)$ has order q. In the other protocols, let g be the generater of $\mathbb{Z}_p^*$, i.e. $2 \leq g \leq p-2$. Both P and g are fixed and known to all the entities.

Protocol descriptions: In each of the protocols in Figure 9.1 an entity, A, communicating to entity B, sends a fresh short-term public value along with a certificate, Cert_A, containing A's long-term public key. The fresh short-term public value is $g^a \bmod p$ for the MTI/A0 and Unified Model protocols, $(K_{pub,B})^a \bmod p$ (where $K_{pub,B} = g^y \bmod p$) for the MTI/B0 and MTI/C0 protocols, and aP for the MQV protocol. Corresponding values and certificates are sent by entity B to A. Each party verifies the authenticity of the certificate received: if any check fails, the protocol should be aborted; if no check fails, the session keys described next, corresponding to the protocol in Figure 9.1, should be computed.

Unified Model and MTI/A0 Protocol Messages

1. $A \rightarrow B: \quad g^a \bmod p \| \text{Cert}_A$
2. $B \rightarrow A: \quad g^b \bmod p \| \text{Cert}_B$

MTI/B0 and MTI/C0 Protocol Messages

1. $A \rightarrow B: \quad (g^y)^a \bmod p \| \text{Cert}_A$
2. $B \rightarrow A: \quad (g^x)^b \bmod p \| \text{Cert}_B$

MQV Protocol Messages

1. $A \rightarrow B: \quad aP \| \text{Cert}_A$
2. $B \rightarrow A: \quad bP \| \text{Cert}_B$

Figure 9.1: Two party authenticated key agreement protocols for the Unified Model, MQV and selected MTI key agreement protocols.

Two Party Authenticated Key Generation: We now explain how session keys are generated in each protocol.

1. **Unified Model:**
   The keys computed by the entities are:
   $K_A = H((g^y)^x \bmod p \| (g^b)^a \bmod p)$,
   $K_B = H((g^x)^y \bmod p \| (g^a)^b \bmod p)$.
   Both parties now share the session key $K_{AB} = H(g^{xy} \bmod p \| g^{ab} \bmod p)$. Here, $H : \mathbb{G} \times \mathbb{G} \rightarrow \{0,1\}^l$ is a cryptographic hash function whose function is to derive a suitable length, l, session key.
2. **MTI/A0:**
   The keys computed by the entities are:

$K_A = (g^y)^a \cdot (g^b)^x \bmod p$,
$K_B = (g^x)^b \cdot (g^a)^y \bmod p$.
Both parties now share the session key $K_{AB} = g^{ay+bx} \bmod p$.

3. **MTI/B0:**
   The keys computed by the entities are:
   $K_A = ((g^x)^b)^{x^{-1}} \cdot g^a \bmod p$,
   $K_B = ((g^y)^a)^{y^{-1}} \cdot g^b \bmod p$.
   Both parties now share the session key $K_{AB} = g^{a+b} \bmod p$.

4. **MTI/C0:**
   The keys computed by the entities are:
   $K_A = ((g^x)^b)^{x^{-1}a} \bmod p$,
   $K_B = ((g^y)^a)^{y^{-1}b} \bmod p$.
   Both parties now share the session key $K_{AB} = g^{ab} \bmod p$.

5. **MQV:**
   The keys computed by the entities are:
   $K_A = h((a + x\overline{aP}) \bmod n) \cdot (bP + yP \cdot \overline{bP})$,
   $K_B = h((b + y\overline{bP}) \bmod n) \cdot (aP + xP \cdot \overline{aP})$.
   The notation is described below. Both parties now share the session key $K_{AB} = h(a + x\overline{aP})(b + y\overline{bP})P$. The notations h and $\overline{Q}$ are described below.

The Unified Model protocol is a standardised protocol [8, 9, 87] that bears some similarity with the MTI protocols but utilises a hash function, H, with concatenation instead of a multiplication to combine various components. The MTI suite of protocols consists of three infinite sequences of protocols: A[κ], B[κ] and C[κ], where $\kappa \in \mathbb{Z}$. Each scheme in the sequence can 'smartly' update the key K_{AB} to the next direct scheme in the sequence using shared keying information. For example, a shared MTI/A0 key, $K_{AB} = g^{ay+bx} \bmod p$, updates into an MTI/A1 key, $K_{AB} = g^{ayx+bxy} \bmod p$, if A uses $\kappa = 1$ in the following MTI/A sequence procedure: (i) A computes $L_{a,\kappa} = x^\kappa a$; (ii) A sends to B: $g^{L_{a,\kappa}} \bmod p$; (iii) A computes $K_A = (g^y)^{L_{a,\kappa}} \cdot (g^{L_{b,\kappa}})^x \bmod p$. Of course B also has to perform a procedure similar

to A's and the shared MTI/A[κ] key becomes $K_{AB} = g^{ayx^{\kappa}+bxy^{\kappa}} \bmod p$. The MTI/C sequence has the lowest computational complexity for updating the shared key.

If Q is an elliptic curve point, then $\overline{Q}$ is a mapping from group elements to an integer range. More specifically in [99], $\overline{Q}$ is defined to be the integer $(\overline{x} \bmod 2^{\lceil f/2 \rceil}) + 2^{\lceil f/2 \rceil}$, where $f = \lfloor \log_2 q \rfloor + 1$ is the bitlength of q and $\overline{x}$ is the binary representation of the x-coordinate of Q. Notice that $\overline{Q} \bmod q \neq 0$. The cofactor h is an integer such that $h = \lfloor (\sqrt{t}+1)^2/q \rfloor$ that is used to protect against small subgroup attacks and is typically small so q is as large as possible.

The MQV protocol was initially proposed by Menezes, Qu and Vanstone [113] and later improved by eliminating the use of hash functions (to enhance its efficiency) with Law and Solinas [99]. The improved protocol, which is presented here, is the most prominent and some versions of it are standardised [8, 9, 87, 89]. A main design goal of this protocol was to avoid the use of hash functions, however, this goal lead to a vulnerability which was exposed by Kaliski [93].

9.1.2.2 The Station-to-Station Protocol

The non-broadcast protocols, which will be presented in Section 9.8, build on the station-to-station (STS) [59] protocol. The STS protocol shown in Figure 9.2 employs an encryption algorithm $\mathcal{E}^{sym}$ from a symmetric encryption scheme and a PKS scheme's signing algorithm Σ.

In this chapter we will break our notational convention to simplify the notation. Here, we let $\Sigma_A(\sigma)$ denote A's signature on the string σ (i.e. $\Sigma_A(\sigma) = \Sigma(\sigma, K_{priv,A})$) and as with Section 6.4.2, $\mathcal{E}_K^{sym}(\sigma)$ denotes the encryption of string σ using a symmetric algorithm and key K. The message flows of the STS key agreement protocol are given in Figure 9.2.

Protocol description: In Figure 9.2 entity A initiates the protocol execution by

Sequence of Protocol Messages

1. $A \rightarrow B:$ $g^a \bmod p \| \mathrm{Cert}_A$
2. $B \rightarrow A:$ $g^b \bmod p \| \mathrm{Cert}_B \| \mathcal{E}^{sym}_{K_{AB}}(\Sigma_B(g^b \bmod p \| g^a \bmod p))$
3. $A \rightarrow B:$ $\mathcal{E}^{sym}_{K_{AB}}(\Sigma_A(g^a \bmod p \| g^b \bmod p))$

Figure 9.2: The station-to-station (STS) key agreement protocol.

sending message (*1*). After receiving message (*1*), entity B is able to calculate the session key $K_{AB} = g^{ab} \bmod p$. The same session key is calculated by A after receiving message (*2*). Messages (*2*) and (*3*) contain signatures on the short-term values to provide key authenticity. Entities A and B verify each other's signatures. Only upon successful signature verification does the protocol continue and is the session key K_{AB} accepted. These signatures are transmitted in encrypted form using the session key K_{AB} which provides key confirmation. The inclusion of the intended recipient's identifier in the signature is an option, see [59, §6] for more details.

9.1.2.3 The Security Models

Our security model is inspired by Blake-Wilson, Johnson and Menezes' extension [26] of the Bellare-Rogaway model [22]. More recent work on models for secure protocols (in particular key agreement protocols) can be found in [20, 39, 45, 46, 140].

9.1.2.4 TAK Protocol History

This chapter includes the major improvements and changes to our original work [4] presented in [5]. In particular, one of the protocols of [4], namely TAK-1 is proved to have perfect forward secrecy and another, TAK-2, is no longer explored in any

detail because of a fatal man-in-the-middle attack that was discoverd by Shim [136]. Details of this attack and what can be learned from it can be found in Section 9.5.1.

9.2 One Round Tripartite Authenticated Key Agreement Protocols

The advantage of Joux's tripartite protocol over any previous tripartite key agreement protocol is that a session key can be established in just one round. The disadvantage is that this key is not authenticated, and this allows a man-in-the-middle attack.

In this section we develop protocols which also need just one round, but which provide a key which is implicitly authenticated to all entities. Our AK protocols are generalisations of the standardised [8, 9, 87] Unified Model protocol [10], the MTI family of protocols [110] and the MQV protocol [99] to the setting of pairings. In fact, we present a single protocol with four different methods of deriving a session key. The numbering sequence for these different methods is unrelated to that of the MTI protocols. Our tripartite protocols use pairings; hence we use the notation $\mathbb{G}_1$, $\mathbb{G}_2$, $\hat{e}$ and P as established in Chapter 2 and utilised in Joux's protocol in Section 3.3.3.

In order to provide session key authentication, some form of authenticated long-term private/public key pairs are needed. As with the other certificate-based protocols (see Section 3.2.3), a certification authority (CA) is used in the initial set-up stage to provide certificates, which bind users' identities to long-term keys. Entity A's long-term public key is $K_{pub,A} = xP$, where $x \in \mathbb{Z}_q^*$ is the long-term private key of A. Element P is a public value which can be included in certificate in order to specify which element is used to construct K_{pub} and the short-term public values. Similarly, Cert_B and Cert_C are the certificates for entities B and C, with $K_{pub,B} = yP$ and $K_{pub,C} = zP$ as their long-term public keys.

As usual, in the protocol message flows given in Figure 9.3, short-term private keys

$a, b, c \in \mathbb{Z}_q^*$ are selected uniformly at random by A, B and C respectively.

Protocol Messages	
1.	$A \rightarrow B, C: \quad aP \| \text{Cert}_A$
2.	$B \rightarrow A, C: \quad bP \| \text{Cert}_B$
3.	$C \rightarrow A, B: \quad cP \| \text{Cert}_C$

Figure 9.3: Tripartite authenticated key agreement (TAK) protocol.

Protocol description: In Figure 9.3, an entity A broadcasting to B and C, sends his fresh short-term public value aP along with a certificate Cert_A containing his long-term public key. Corresponding values and certificates are broadcast by B and C to A, C and A, B respectively. Notice that the protocol messages are just the same as in Joux's protocol (Figure 3.4, Page 53), except for the addition of certificates. Each party verifies the authenticity of the two certificates he receives. If any check fails, the protocol should be aborted. When no check fails, one of four possible session keys described next should be computed.

9.2.1 TAK Key Generation

1. **Type 1 (TAK-1):**
   The keys computed by the entities are:
   $K_A = H(\hat{e}(bP, cP)^a \| \hat{e}(yP, zP)^x)$,
   $K_B = H(\hat{e}(aP, cP)^b \| \hat{e}(xP, zP)^y)$,
   $K_C = H(\hat{e}(aP, bP)^c \| \hat{e}(xP, yP)^z)$.
   By bilinearity, all parties now share the session key
   $K_{ABC} = H(\hat{e}(P, P)^{abc} \| \hat{e}(P, P)^{xyz})$. Here, $H : \mathbb{G}_2 \times \mathbb{G}_2 \rightarrow \{0, 1\}^l$ is a cryptographic hash function.

2. **Type 2 (TAK-2):**

The keys computed by the entities are:
$K_A = \hat{e}(bP, zP)^a \cdot \hat{e}(yP, cP)^a \cdot \hat{e}(bP, cP)^x = \hat{e}(yP, cP)^a \cdot \hat{e}(bP, a \cdot zP + x \cdot cP)$,
$K_B = \hat{e}(aP, zP)^b \cdot \hat{e}(xP, cP)^b \cdot \hat{e}(aP, cP)^y = \hat{e}(xP, cP)^b \cdot \hat{e}(aP, b \cdot zP + y \cdot cP)$,
$K_C = \hat{e}(aP, yP)^c \cdot \hat{e}(xP, bP)^c \cdot \hat{e}(aP, bP)^z = \hat{e}(xP, bP)^c \cdot \hat{e}(aP, c \cdot yP + z \cdot bP)$.
The session key is $K_{ABC} = \hat{e}(P, P)^{(ab)z+(ac)y+(bc)x}$.

3. **Type 3 (TAK-3):**
The keys computed by the entities are:
$K_A = \hat{e}(yP, cP)^x \cdot \hat{e}(bP, zP)^x \cdot \hat{e}(yP, zP)^a = \hat{e}(bP, zP)^x \cdot \hat{e}(yP, x \cdot cP + a \cdot zP)$,
$K_B = \hat{e}(aP, zP)^y \cdot \hat{e}(xP, cP)^y \cdot \hat{e}(xP, zP)^b = \hat{e}(aP, zP)^y \cdot \hat{e}(xP, y \cdot cP + b \cdot zP)$,
$K_C = \hat{e}(aP, yP)^z \cdot \hat{e}(xP, bP)^z \cdot \hat{e}(xP, yP)^c = \hat{e}(aP, yP)^z \cdot \hat{e}(xP, z \cdot bP + c \cdot yP)$.
The session key is $K_{ABC} = \hat{e}(P, P)^{(xy)c+(xz)b+(yz)a}$.

4. **Type 4 (TAK-4):**
The keys computed by the entities are:
$K_A = \hat{e}(bP + H(bP \| yP)yP, cP + H(cP \| zP)zP)^{a+H(aP\|xP)x}$,
$K_B = \hat{e}(aP + H(aP \| xP)xP, cP + H(cP \| zP)zP)^{b+H(bP\|yP)y}$,
$K_C = \hat{e}(aP + H(aP \| xP)xP, bP + H(bP \| yP)yP)^{c+H(cP\|zP)z}$.
The session key is
$K_{ABC} = \hat{e}(P, P)^{(a+H(aP\|xP)x)(b+H(bP\|yP)y)(c+H(cP\|zP)z)}$. Here, $H : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$ is a cryptographic hash function.

9.2.2 TAK Key Generation Notes

- Joux's protocol [90] and our protocols are all vulnerable to 'small subgroup' attacks and variants of them as observed by Lim and Lee [104]. To protect against this, a verification algorithm should be applied by each entity to ensure that their received elements are actually in $\mathbb{G}_1$. This is fairly cheap to do when $\mathbb{G}_1$ is an elliptic curve group.

- In all four cases, key generation is role symmetric and each entity uses knowledge of both short-term and long-term keys to produce a unique shared secret key. No party has control over the resulting session key K_{ABC} and if any one

of a, b, c is chosen uniformly at random, then K_{ABC} is a random element of $\mathbb{G}_2$. Of course delays in receipt of messages from the last entity should not be tolerated by the other entities. This is because after the last entity sees all the other participants' messages, he is capable of fixing a small number of bits in the final shared secret key. See [115] for more details.

- Since all four keys are created after transmitting the same protocol messages, the communication overhead of each protocol version is identical. However, TAK-2 and TAK-3 key generation require slightly more computation compared to TAK-1, even when the number of pairing computations is reduced to two from three as described in the key generation equations. Better still, TAK-4 requires only a single pairing computation per entity, but in addition requires three hash function computations by each entity. All the protocols can exploit pre-computation if entities know in advance with whom they will be sharing a key. In TAK-1, for example, all entities can pre-compute the term $\hat{e}(P, P)^{xyz}$. In TAK-2, for example, entity A can pre-compute $a \cdot yP$ and $a \cdot zP$, with similar pre-computations for B and C. However, these terms cannot be re-used because fresh a, b and c should be used in each new protocol session. We present a detailed account of the efficiency of the four protocols in Table 9.1.

	Joux	TAK-1	TAK-2	TAK-3	TAK-4
Comp. Overhead $\langle \hat{e}, \times\mathbb{G}_2, \times\mathbb{G}_1, +\mathbb{G}_1, H \rangle$	1,0,2,0,0	2,0,3,0,1	2,1,4,1,0	2,1,4,1,0	1,0,4,2,3
Overhead w/ Precomp. $\langle \hat{e}, \times\mathbb{G}_2, \times\mathbb{G}_1, +\mathbb{G}_1, H \rangle$	1,0,1,0,0	1,0,1,0,1	2,1,2,1,0	2,1,2,1,0	1,0,3,2,2
Comm. Overhead $\langle \mathbb{G}_2, \mathbb{G}_1, \text{Misc.} \rangle$	0,3,0	... 0,3,3 Certificates ...			
⟨Pass,Broadcast,Round⟩	6,3,1	... 6,3,1 ...			

Table 9.1: Efficiency comparison for one round tripartite key agreement protocols.

- Table 9.1: The computational overhead is expressed as a five-tuple which represents the number of bilinear-pairings; multiplications in $\mathbb{G}_2$; multiplications in $\mathbb{G}_1$; additions in $\mathbb{G}_1$; and hash function evaluations in that order. The computational overhead with precomputation expresses the same five-tuple. The

precomputation assumes that entities know in advance who they are communicating with. Additions in $\mathbb{Z}_q$ are omitted because they are cheap compared to the other operations. The total broadcast communication overhead expressed as triples represents total message sizes, which can be measured using the number of $\mathbb{G}_2$ elements, $\mathbb{G}_1$ elements and miscellaneous elements, in that order. The final triple represents the total number of passes, broadcasts and rounds, respectively. Furthermore, in Table 9.1, some of the multiplications in $\mathbb{G}_1$ can be expressed as exponentiations in $\mathbb{G}_2$. However, we choose to do point multiplication in $\mathbb{G}_1$ followed by the pairing map, since this is more efficient than a pairing evaluation followed by an exponentiation in $\mathbb{G}_2$. This is because $\mathbb{G}_2$ is a subgroup of a large finite field (recall Section 2.3) whereas $\mathbb{G}_1$ is a subgroup of a fairly small curve.

9.2.3 Rationale for the TAK Keys' Algebraic Forms

- Protocol TAK-1 is analogous to the Unified Model protocol, whilst protocols TAK-2 and TAK-3 have their roots in the MTI/A0 protocol. The MTI-like variant of TAK-1, in which the agreed key is $K_{ABC} = \hat{e}(P,P)^{abc+xyz}$, suffers from a severe form of key-compromise impersonation attack. This attack does not require the adversary to learn a long-term private key. Rather, the adversary only needs to obtain a session key and one of the short-term secret values used in a protocol run to mount the attack. To illustrate this, suppose E has obtained a, A's short-term private key and the session key $K_{ABC} = \hat{e}(P,P)^{abc+xyz}$. Then E can calculate $\hat{e}(P,P)^{xyz}$ using a and K_{ABC} by computing $K_{ABC}/\hat{e}(bP,cP)^a$. Knowledge of this value now allows E to impersonate any entity engaging in a protocol run with A. It would be prudent to derive session (and MAC keys for key confirmation if desired) by applying a hash function to each K_{ABC}. This would prevent problems arising from the possible existence of relatively easy bits in the BDHP. Using a hash function in TAK-1 has the additional benefit that it allows a security proof to be given (assuming H is modelled by a random oracle) – see Section 9.3.

- Protocol TAK-4 is modelled on the MQV protocol but avoids that protocol's unknown key-share weakness [93] by using a hash function H to combine both the short-term *and* long-term private keys. Here H's output is assumed to be onto $\mathbb{Z}_q^*$. Note that the protocol resulting from omission of this hash function produces the key $K_{ABC} = \hat{e}(P,P)^{(a+x)(b+y)(c+z)}$. However, this version of the protocol (which avoids the use of hash functions for efficiency) suffers from an unknown key-share weakness similar to that presented for the MQV protocol in [93], wherein the attacker *does* know the private key corresponding to his registered public key. As a consequence, this attack cannot be prevented by requiring the adversary to provide a PoP for her private key as part of the registration process (we cannot just assume the PoP occurred). See Section 9.5.5 for further discussion of unknown key-share attacks.

- Other MTI-like protocols can be produced if entities A, B and C broadcast the ordered pairs $ayP \| azP$, $bxP \| bzP$ and $cxP \| cyP$ respectively (along with the appropriate certificates). This protocol can be used to produce the MTI/C0-like shared secret key $K_{ABC} = \hat{e}(P,P)^{abc}$, which for example A calculates by $K_{ABC} = \hat{e}(bxP, cxP)^{ax^{-2}}$. It can also be used to produce the MTI/B0-like shared secret key $K_{ABC} = \hat{e}(P,P)^{ab+bc+ca}$, which A can calculate by $K_{ABC} = \hat{e}(bxP, cxP)^{x^{-2}} \cdot \hat{e}(P, bxP)^{ax^{-1}} \cdot \hat{e}(P, cxP)^{ax^{-1}}$. Although these protocols produce a key with forward secrecy, we do not consider them further here because they require significantly higher bandwidth and do not offer a security advantage over our other protocols. For example, both protocols suffer from key compromise impersonation attacks and the MTI/C0 analogue is also vulnerable to known session key attacks.

Our TAK protocols include long-term private keys in the computation of each K_{ABC} in order to prevent man-in-the-middle attacks. Shim [136], however, has shown that simply involving the long-term keys is not sufficient to prevent a man-in-the-middle attack on TAK-2. Due to this severe vulnerability in the TAK-2 protocol, the discussion of its security will be limited to Section 9.5.1. The TAK-2 protocol should be avoided and merely remains in this thesis for completeness and to provide a contrast with our other protocols. For the remaining protocols, other forms of

active attack can still occur. We consider such attacks on a case-by-case basis in Section 9.5 after considering proofs of security for TAK-1.

9.3 Security Model

In this section, our security model for TAK protocols is introduced. The security of protocol TAK-1 in this model is considered in detail. Furthermore, the ways in which it differs from previous work are highlighted.

Our model is similar to those introduced by Bellare and Rogaway [22] and Blake-Wilson, Johnson and Meneze [26] with some simplifications that are possible because of the one round nature of our TAK protocols. In particular, we avoid the use of matching conversations (and session IDs introduced in later work [20]).

We let k be a security parameter, the protocol consisting of probabilistic polynomial time algorithms which take k as input. We assume a set of protocol participants, the polynomial bound in k on the number of participants is $T_1(k)$. Multiple users are required because we are modelling an open network. Letters $A, B, C, \ldots$ will be used to label protocol participants, while E is reserved for our adversary (who is not a participant). Each participant A is modelled by an oracle Π_A^s, which the adversary E can query at will. Here, s is a session number, that determines which random tape Π_A^s will use. In previous work, oracles were of the form $\Pi_{i,j}^s$ and modelled messages sent from participant i to participant j in session s. We remove this dependence on receiving parties; in all our protocols, all messages are broadcast through E to model active attacks. We omit the dependence on receiving parties since all the messages are intended to be broadcast. The adversary E is also capable of not delivering any messages.

Oracles exist in one of several possible states Accept, Reject, or *. In our protocols, an oracle accepts only after receipt of two properly formulated messages (containing different certificates to its own) and the transmission of two messages, not necessarily

in that order (and with the received messages possibly originating from the adversary and not the oracles identified in the certificates in those messages). When an oracle accepts, we assume it accepts holding a session key K that is k bits in length. We also assume there is a key generation process $\mathcal{G}$ which produces a description of groups $\mathbb{G}_1$ and $\mathbb{G}_2$ and the map $\hat{e}$, assigns random tapes and oracles as necessary, distributes long-term private keys to participants, and prepares certificated long-term public keys. Thus our model assumes a perfect certification process and does not capture attacks based on registration weaknesses like those described in Section 9.5.5. As usual, the benign adversary is defined to be one that simply passes messages to and fro between participants. An oracle can reject computing a session key if a received message is not correctly formulated, that is, not as defined in the protocol specification. When the oracle's state is '*', it has not yet made a decision to accept or reject.

Adversary E is assumed to have complete control over the network, and we allow E to make three kinds of query to an oracle Π^s_A: Send, Reveal and Corrupt. These have the usual meanings, as per [26]: Send allows the adversary E to send a message of her choice to Π^s_A and to record the response, or to induce Π^s_A to initiate a protocol run with participants of E's choosing; Reveal reveals the session key (if any) held by Π^s_A; while Corrupt produces the long-term private key of an uncorrupted oracle. Notice that when making a Send query, E does not need to specify the intended recipients of the oracle's reply. This is because, in our protocols, the oracle's replies are independent of these recipients. In fact, E can relay these messages to any party of her choosing.

In our TAK protocols, each party sends the same message to two other participants and receives two messages from those participants. In our model three oracles Π^s_A, Π^t_B and Π^u_C can be said to have *participated in a matched session* if they have received messages exclusively generated by each other (via the adversary). In other words, the two messages Π^s_A receives are those generated by Π^t_B and Π^u_C; Π^t_B receives two messages generated by Π^s_A and Π^u_C; and Π^u_C receives two messages generated by Π^s_A and Π^t_B.

In addition to the above queries, we allow E to make one further Test query of one of the oracles Π_A^s at any point during her attack. This oracle must be *fresh*, i.e. , the oracle must:

1. Be in an Accept state.
2. Not have been subject to a Reveal query. An oracle subjected to a Reveal query is called an 'opened' oracle.
3. Not have been subject to a Corrupt query. That is, the oracle should be uncorrupted.
4. Not have participated in a matched session with any opened oracle.
5. Not have received messages containing the certificate of a corrupted oracle.

The reply to this Test query is either the session key K held by the oracle, or a random k-bit string, the choice depending on a fair coin toss. The adversary's advantage, denoted $advantage^E(k)$, is the probability that E can distinguish K from the random string. Notice that we remove the unnatural restriction in earlier models [22, 26] that this Test query be the adversary's last interaction with the model.

We say that a protocol is a secure TAK protocol if:

1. In the presence of the benign adversary, and when oracles participate in a matched session, all the oracles always accept holding the same session key, which is distributed randomly and uniformly on $\{0,1\}^k$.
2. Whenever uncorrupted oracles Π_A^s, Π_B^t and Π_C^u participate in a matched session, then all three oracles accept and hold the same session key, which is again uniformly distributed on $\{0,1\}^k$.
3. $advantage^E(k)$ is negligible.

The first condition ensures that a secure TAK protocol does indeed distribute a key of the correct form. The second condition ensures that this remains true even if *all* other oracles are corrupted. The last condition says that no adversary can obtain any information about the session key held by a fresh oracle.

We can also formally model the forward secrecy properties of TAK protocols. The model is largely as before, but now the interaction with E is slightly different. In addition to the usual Send, Reveal and Corrupt queries, we allow E to make one Test query of an oracle of her choice, say Π_A^s. We assume that Π_A^s has accepted and has participated in a matched session with uncorrupted oracles Π_B^t and Π_C^u. Thus Π_A^s will have calculated a session key in common with Π_B^t, Π_C^u. We further assume that none of these three oracles has been the subject of a Reveal query. However, any or all of the oracles Π_A, Π_B, Π_C may be corrupted at any point in E's attack. In response to E's query, E is given the long-term private keys for Π_A^s, Π_B^t and Π_C^u (these oracles are now effectively corrupted). Adversary E is also given either the session key K held by Π_A^s or a random k-bit string, the choice depending on a fair coin toss. The adversary's advantage, denoted $advantage^{E,fs}(k)$, is the probability that E can distinguish K from the random string. Again, the Test query need not be the adversary's last interaction with the model.

If in the above game, $advantage^{E,fs}(k)$ is negligible, it is said that the TAK protocol has perfect forward secrecy.

9.4 Security Proofs for TAK-1

With the description of our security model in hand, we now state our first theorem:

Theorem 9.1 Protocol TAK-1 is a secure TAK protocol, assuming that the adversary makes no Reveal queries, that the BDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ is hard and provided that H is a random oracle.

Proof. **Conditions 1 and 2:** Given the assumption that H is a random oracle, these conditions follow directly from the protocol description.

Condition 3: Suppose that $advantage^E(k) = n(k)$ is non-negligible. We show how to construct from E an algorithm F which solves the BDHP with non-negligible probability. We describe F's operation. F's input is a description of the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and the map $\hat{e}$, a non-identity element $P \in \mathbb{G}_1$, and a triple of elements $x_AP, x_BP, x_CP \in \mathbb{G}_1$ with x_A, x_B, x_C chosen randomly from $\mathbb{Z}_q^*$. F's task is to compute and output the value $g^{x_Ax_Bx_C}$ where $g = \hat{e}(P, P)$.

F operates as follows. F chooses a triple $A, B, C \in \mathsf{ID}$ uniformly at random. F simulates the running of the key generation algorithm $\mathcal{G}$, choosing all participants' long-term private keys randomly itself, and computing the corresponding long-term public keys and certificates, but with the exception of Π_A, Π_B and Π_C's keys. As public values for Π_A, Π_B and Π_C, F chooses the values x_AP, x_BP, x_CP respectively. Then F starts adversary E.

In E's attack, F will simulate all the oracles Π_i, $i \in \mathsf{ID}$. So F must answer all the oracle queries that E makes. F answers E's queries as follows. F simply answers E's distinct H queries at random, maintaining a table of queries and responses as he proceeds. Note that we do not allow our adversary to make Reveal queries, so F does not need to answer any queries of this type. F answers any Corrupt queries by revealing the long-term private key and replacing the public/private key pair, except for Corrupt queries on Π_A, Π_B or Π_C. In the event of such queries, F aborts. F replies to Send queries in the usual way, with correctly formulated responses $a_{i,s}P \| Cert_{\Pi_i}$ for all oracles Π_i^s, where $a_{i,s} \in_R \mathbb{Z}_q^*$.

Finally, we consider how F responds to the Test query on oracle Π_i. F generates a random bit $b \in \{0, 1\}$. If $b = 0$, F should respond with the key held by Π_i, while if $b = 1$, F should respond with a random k-bit value. Now F is capable of answering the Test query correctly except when $b = 0$ and the tested oracle is an instance of Π_A, Π_B or Π_C. In this last case, F's response should be of the form $H(Q \| g^{x_Ax_Bx_C})$ where

$Q \in \mathbb{G}_2$, involving the invocation of the random oracle. This use of H should be consistent with previous and future uses, but of course F does not know $g^{x_A x_B x_C}$, so cannot properly simulate the oracle in this case. Instead, F responds with a random k-bit value. This potentially introduces an imperfection into F's simulation, but we will argue below that this has no effect on success probabilities.

The final stage is as follows. Let $T_2(k)$ denote a polynomial bound on the number of H queries answered by F in the course of E's attack. F picks a value ℓ uniformly at random from $\{1, \ldots, T_2(k)\}$. Now F parses the ℓ-th H query into the form $Q \| W$ where $Q, W \in \mathbb{G}_2$. If this is not possible, F aborts. If it is, then F outputs W as its guess for the value $g^{x_A x_B x_C}$ and stops.

Now we must evaluate the probability that F's output is correct. Notice that E's view of F's simulation of the oracles is indistinguishable from E's view in a real attack provided that F is not forced to abort when asked a Corrupt query and that E does not detect that F's simulation of the random oracle was deficient when responding to the Test query – further details of these situations are given below. Now E picks some accepted oracle $\Pi_{i_1}^s$ for its Test query in a real attack. Suppose that $\Pi_{i_1}^s$ has received two messages containing certificates of oracles Π_{i_2} and Π_{i_3}. The session key held by oracle $\Pi_{i_1}^s$ will be of the form $H(Q \| g^{x_{i_1} x_{i_2} x_{i_3}})$ where $Q \in \mathbb{G}_2$ and x_{i_j} is the long-term private key of Π_{i_j}, $1 \leq j \leq 3$. Since by definition, the oracles Π_{i_j} are uncorrupted, and E does not ask any Reveal queries, if E is to succeed in distinguishing this session key from a random string with non-negligible probability $n(k)$, then E must have queried H on an input of the form $Q \| g^{x_{i_1} x_{i_2} x_{i_3}}$ at some point in its attack with some non-negligible probability $n'(k)$. The probability that this event occurs in F's simulation of the oracles is therefore also $n'(k)$. Since F outputs a random query from the list of $T_2(k)$ queries, has randomly distributed public keys $x_A P$, $x_B P$, $x_C P$ amongst the $T_1(k)$ participants, and is only deemed successful if he does not abort and his output is of the form $g^{x_A x_B x_C}$, we see that F is successful with probability at least:

$$\frac{n'(k)}{T_1(k)^3 T_2(k)} \quad .$$

However, this is still non-negligible in k.

We claim that our argument is not affected by the imperfection introduced into F's simulation when E asks a Corrupt query that F cannot answer: to be successful, E must ask a Test query of an uncorrupted but accepted oracle which has received messages containing certificates of two further uncorrupted oracles. This means that for E to be successful, at least three distinct, uncorrupted oracles must remain. So F, having made a random choice of where to place public keys $x_A P, x_B P, x_C P$, has at least a $\frac{1}{T_1(k)^3}$ chance of not facing an unanswerable Corrupt query whenever E is successful. This factor is already taken into account in our analysis.

One problem remains: what effect on E's behaviour does F's deviation have in giving a random response to the "difficult" Test query? In particular, what effect does it have on success probabilities? E's behaviour can only differ from that in a true attack run if E detects any inconsistency in F's simulation of the random oracle. In turn, this can only happen if at some point in the attack, E queries H on an input of the form $Q \| g^{x_A x_B x_C}$. For otherwise, no inconsistency arises. At this point, E's behaviour becomes undefined. (In this situation, E might guess that F's response to the Test query is a random key ($b = 1$) rather than the "correct" key ($b = 0$). But we must also consider the possibility that E simply might not terminate.) So we assume that F simply aborts his simulation whenever E's attack lasts longer than some polynomial bound $T_3(k)$ on the length of a normal attack. Notice that H has been queried on an input of the form $Q \| g^{x_A x_B x_C}$ at some point in F's simulation, and that up until this point, E's view is indistinguishable from that in a real attack. Thus, the number of H queries made by E will still be bounded by $T_2(k)$ up to this point, and an input of the required type will occur amongst these. So F's usual guessing strategy will be successful with probability $1/T_2(k)$ even when E's behaviour is affected by F's inability to correctly respond to the Test query. Since this is the same success probability for guessing in the situation where everything proceeds normally, it is now easy to see that F's overall success probability is still at least:

$$\frac{n'(k)}{T_1(k)^3 T_2(k)} \quad .$$

□

Next is our theorem about the security of protocol TAK-1 in our forward security model.

Theorem 9.2 Protocol TAK-1 has perfect forward secrecy, assuming that the adversary makes no Reveal queries, that the BDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ is hard and provided that H is a random oracle.

Proof. Suppose that $advantage^{E,fs}(k) = n(k)$ is non-negligible. We show how to construct from E an algorithm F which solves the BDHP with non-negligible probability. We describe F's operation. F's input is a description of the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and the map $\hat{e}$, a non-identity element $P \in \mathbb{G}_1$, and a triple of elements $x_A P, x_B P, x_C P \in \mathbb{G}_1$ with x_A, x_B, x_C chosen randomly from $\mathbb{Z}_q^*$. F's task is to compute and output the value $g^{x_A x_B x_C}$ where $g = \hat{e}(P, P)$.

F operates as follows. F simulates the running of the key generation algorithm $\mathcal{G}$, choosing all participants' long-term private keys randomly itself, and computing the corresponding long-term public keys and certificates.

F also chooses a triple $A, B, C \in \mathsf{ID}$ uniformly at random, and three positive integers s, t, u that are all bounded above by the number $T_3(k)$ of different sessions that E enters into across all the oracles. Then F starts adversary E.

F must answer all the oracle queries that E makes. F answers E's queries as follows. F simply answers E's distinct H queries at random, maintaining a table of queries and responses as he proceeds. Note that we do not allow our adversary to make Reveal queries, so F does not need to answer any queries of this type. F answers any Corrupt queries by revealing the long-term private key that it holds. F replies to Send queries in the usual way, with correctly formulated responses $a_{i,r} P \| Cert_{\Pi_i}$ for all oracles Π_i^r, where $a_{i,r} \in_R \mathbb{Z}_q^*$, except when queried for responses for oracles Π_A^s, Π_B^t and Π_C^u. In these special cases, F responds with $x_A P \| Cert_{\Pi_A}$, $x_B P \| Cert_{\Pi_B}$ and $x_C P \| Cert_{\Pi_C}$, respectively.

Finally, we consider how F responds to the Test query on oracle Π_i. F generates a random bit $b \in \{0,1\}$. If $b = 0$, F should respond with the key held by Π_i, while if $b = 1$, F should respond with a random k-bit value. Now F is capable of answering the Test query correctly except in one special case: this is when $b = 0$, when the tested oracle is Π_A^s, Π_B^t or Π_C^u and when the tested oracle has participated in a matched session which comprises exactly these three oracles. In this last case, F's response should be of the form $H(g^{x_A x_B x_C} \| W)$ where $W \in \mathbb{G}_2$, but F cannot properly simulate the oracle in this case. Instead, F responds with a random k-bit value. This potentially introduces an imperfection into F's simulation, but this has no effect on success probabilities; this can be argued just as in the proof of Theorem 9.1.

Let $T_2(k)$ denote a polynomial bound on the number of H queries answered by F in the course of E's attack. F picks a value ℓ uniformly at random from $\{1, \ldots, T_2(k)\}$. Now F parses the ℓ-th H query into the form $Q \| W$ where $Q, W \in \mathbb{G}_2$. If this is not possible, F aborts. If it is, then F outputs Q as its guess for the value $g^{x_A x_B x_C}$ and stops. Notice that E's view of F's simulation of the oracles is indistinguishable from E's view in a real attack, provided that E does not detect that F's simulation of the random oracle was deficient when responding to the Test query. Now E picks some accepted oracle $\Pi_{i_1}^r$ for its Test query in a real attack. Suppose that $\Pi_{i_1}^r$ has received two messages containing the short-term values of oracles Π_{i_2} and Π_{i_3}. The session key held by oracle $\Pi_{i_1}^r$ will be of the form $H(g^{x_{i_1} x_{i_2} x_{i_3}} \| W)$ where $W \in \mathbb{G}_2$ and x_{i_j} is the short-term private key of Π_{i_j}, $1 \leq j \leq 3$. Since E does not ask any Reveal queries, if E is to succeed in distinguishing this session key from a random string with non-negligible probability $n(k)$, then E must have queried H on an input of the form $g^{x_{i_1} x_{i_2} x_{i_3}} \| W$ at some point in its attack with some non-negligible probability $n'(k)$. The probability that this event occurs in F's simulation is therefore also $n'(k)$. Recall that F outputs a random query from the list of $T_2(k)$ queries, has randomly distributed values $x_A P$, $x_B P$, $x_C P$ as short-term keys amongst the $T_3(k)$ sessions, and is only deemed successful if his output is of the form $g^{x_A x_B x_C}$. Recall too that E only attacks oracles that have participated in matched sessions. Combining these

facts, we see that F is successful with probability better than:

$$\frac{n'(k)}{T_3(k)^3 T_2(k)} \quad .$$

However, this is still non-negligible in k.

□

We comment on the significance of Theorems 9.1 and 9.2. We emphasise that our proofs of security do not allow the adversary to make Reveal queries of oracles. This means that our proofs do not capture known session-key attacks. In fact, protocol TAK-1 is vulnerable to a simple attack of this type. In Section 9.5.2 we describe the attack in full. The attack only works on TAK-1 because of the symmetry of the short-term components, and attacks of this type do not appear to apply to TAK-2, TAK-3 or TAK-4. The attack is analogous to known session key attacks well-understood for other protocols (see the comments following [26, Theorem 11] for an example).

In Section 9.8.1 we consider a confirmed version of Joux's protocol. This protocol is obtained in the usual way by adding three confirmation messages (which are piggy-backed on other messages in a non-broadcast environment), one for each protocol participant. The confirmation messages use encryptions and signatures which could be replaced with MACs using keys derived from the short term values exchanged during the protocol run. We expect that the techniques used to prove the security of Protocol 2 of [26] can be adapted to prove that the described confirmed version of TAK-1 is indeed a secure AKC protocol. The analysis presented in Section 9.8.3 explains why we did not further the research into pairing based AKC protocols.

Finally, the techniques used to prove Theorems 9.1 and 9.2 do not appear to extend to our other protocols. Nor has any security proof yet appeared for the MQV protocol (on which TAK-4 is based), although the MQV protocol is widely believed to be secure and has been standardised [8, 9, 87, 89]. In any case, as we shall see in the next section, current security models do not handle all the reasonable attack types and so need to be augmented by *ad hoc* analysis.

We extended the model in [26] to include forward security and proved that TAK-1 is forward secure. Other models [20, 140] also capture forward security.

Furthermore, key-compromise attacks like those described in Section 9.5.4 are not captured by our model. Finally, our proof assumes that H is a random oracle: again recent work [39] shows that security proofs for key agreement protocols can be obtained in the standard model (at some computational cost).

The apparent difficulty in obtaining security proofs for our protocols and the limitations of our security model motivates the additional *ad hoc* security analysis presented in the next section.

9.5 Heuristic Security Analysis of TAK Protocols

We present a variety of attacks on the three TAK protocols that are not captured by the security models of the previous section. These are mostly inspired by earlier attacks on the two-party MTI protocols. Following this analysis, we summarise the security attributes of our TAK protocols in Table 9.2. In this section, we use E_A to indicate that the adversary E is impersonating A in sending or receiving messages intended for or originating from A. Similarly, $E_{B,C}$ denotes an adversary impersonating both B and C. To begin with, we will examine Shim's attack on TAK-2 [136].

9.5.1 Shim's Man-in-the-Middle Attack on TAK-2

In the attack demonstrated by Shim [136], adversary E creates short-term private keys $a', b', c' \in \mathbb{Z}_q^*$ and replaces short-term public keys of A, B and C with $\mu'_A = a'P$, $\mu'_B = b'P$ and $\mu'_C = c'P$ respectively. As in Section 3.3.1, E_A indicates that E is impersonating A in sending or receiving messages intended for or originating from A. Then, entities A, B and C compute the following keys:

$$\begin{aligned} K_{AE_BE_C} &= \hat{e}(P, zP)^a \cdot \hat{e}(yP, \mu'_C)^a \cdot \hat{e}(\mu'_B, P_C)^x = \hat{e}(P, P)^{ab'z+ac'y+b'c'x}, \\ K_{BE_AE_C} &= \hat{e}(P, zP)^b \cdot \hat{e}(xP, \mu'_C)^b \cdot \hat{e}(\mu'_A, P_B)^y = \hat{e}(P, P)^{a'bz+bc'x+a'c'y}, \end{aligned}$$

$$K_{CE_AE_B} = \hat{e}(P, yP)^c \cdot \hat{e}(xP, \mu'_B)^c \cdot \hat{e}(\mu'_A, P_B)^z = \hat{e}(P, P)^{a'cy+b'cy+a'b'z}.$$

Now E with the knowledge of the triple $\langle a', b', c' \rangle$ is able to compute three different session keys using the available long-term public keys xP, yP and zP as follows:

$$\begin{aligned}
K_{E_{B,C}A} &= \hat{e}(aP, zP)^{b'} \cdot \hat{e}(yP, aP)^{c'} \cdot \hat{e}(xP, P)^{b'c'} = \hat{e}(P, P)^{ab'z+ac'y+b'c'x}, \\
K_{E_{A,C}B} &= \hat{e}(bP, zP)^{a'} \cdot \hat{e}(xP, bP)^{c'} \cdot \hat{e}(yP, P)^{a'c'} = \hat{e}(P, P)^{a'bz+bc'x+a'c'y}, \\
K_{E_{A,B}C} &= \hat{e}(cP, yP)^{a'} \cdot \hat{e}(xP, cP)^{b'} \cdot \hat{e}(zP, P)^{a'b'} = \hat{e}(P, P)^{a'cy+b'cy+a'b'z}.
\end{aligned}$$

Since $K_{AE_BE_C} = K_{E_{B,C}A}$, $K_{BE_AE_C} = K_{E_{A,C}B}$ and $K_{CE_AE_B} = K_{E_{A,B}C}$, the attack renders TAK-2 insecure against a man-in-the-middle attack. The attack is present because every pairing computation required to produce the key can be computed using two short-term values which are injected by the adversary. We did not find this attack because a two party heuristic analysis of the man-in-the-middle attack was naively used and not extended to the group setting.

9.5.2 Known Session Key Attack on TAK-1

We present a known session key attack on TAK-1 that makes use of session interleaving and message reflection. In the attack, E interleaves three sessions and reflects messages originating from A back to A in the different protocol runs. The result is that the session keys agreed in the three runs are identical, so E, upon revealing one of them, gets keys for two subsequent sessions as well. In what follows, $E_{B,C}$ denotes an adversary impersonating both B and C.

A is convinced to initiate three sessions with E:
Session $\alpha : A \rightarrow E_{B,C}$: $aP \| \text{Cert}_A$ (1^α)
Session $\beta : A \rightarrow E_{B,C}$: $a'P \| \text{Cert}_A$ (1^β)
Session $\gamma : A \rightarrow E_{B,C}$: $a''P \| \text{Cert}_A$ (1^γ)

E reflects and replays pretending to be B and C, to complete session α:
$E_B \rightarrow A: \quad a'P \| \text{Cert}_B \quad (2^\alpha)$
$E_C \rightarrow A: \quad a''P \| \text{Cert}_C \quad (3^\alpha)$

Similarly, the second session is completed by $E_{B,C}$ sending $a''P \| \text{Cert}_B$ (2^β) and $aP \| \text{Cert}_C$ (3^β) to A. In the third parallel session she sends $aP \| \text{Cert}_B$ (2^γ) and $a'P \| \text{Cert}_C$ (3^γ) to A.

If E is now able to obtain the first session key $H(\hat{e}(P,P)^{aa'a''} \| \hat{e}(P,P)^{xyz})$, she then knows the keys for the next two sessions, as these are identical to this first session key.

9.5.3 Forward Secrecy Weakness in TAK-3

As shown in the proof of Theorem 9.2, TAK-1 has perfect forward secrecy. Protocol TAK-4 also appears to have this property because the key K_{ABC} agreed also includes the component $\hat{e}(P,P)^{abc}$. However, it is straightforward to see that if an adversary obtains two long-term private keys in TAK-3, then she has the ability to obtain old session keys (assuming she keeps a record of the public values aP, bP, cP). Thus TAK-3 does not enjoy forward secrecy. The protocol can made into one which enjoys perfect forward secrecy, at extra computational cost, by using the key $K_{ABC} \cdot \hat{e}(P,P)^{abc}$ in place of the key K_{ABC}.

9.5.4 Key-Compromise Impersonation Attack on TAK-1

We present a key-compromise impersonation attack on TAK-1 that occurs when E can obtain the long-term private key of one of the entities. Suppose E has obtained x, A's long-term private key. Then E can calculate $\hat{e}(P,P)^{xyz}$ using x and public data in B and C's certificates. Knowledge of this value now allows E to impersonate any entity engaging in a TAK-1 protocol run with A (and not just A). For example

to impersonate C to both A and B, E_C with possession of x simply broadcasts $c'P\|\text{Cert}_C$ and obtains the protocol messages, $aP\|\text{Cert}_A$ and $bP\|\text{Cert}_B$ from A and B respectively. The adversary E_C can now compute the session key $K_{ABE_C} = H(\hat{e}(P,P)^{abc'}\|\hat{e}(P,P)^{xyz})$.

Due to the combinations of long-term and short-term key components used in computing K_{ABC}, these kinds of attacks do not appear to apply to TAK-3 nor to TAK-4. However, Shim [136] presented 'partial key compromise impersonation attacks' on TAK-3 and TAK-4. These attacks use A's long-term key to impersonate C to B. However, A does not compute the same session key as B and C do, and the adversary does not learn A's session key. So this attack is only meaningful in the scenario where A does not use its session key to communicate with B or C, but B and C use their version of the key to communicate with one another. In the attack, the adversary has A's long-term key and replaces A's short-term key in the protocol run. It therefore is not surprising that B will compute a key that the adversary can also compute. Indeed this attack is tantamount to a simple impersonation attack on A by the adversary. This is because E (who has possesion of x) is really just impersonating A to B (and not C to B) by replacing the short-term public key of A with $a'P$.

In fact, no implicitly authenticated conference key agreement protocol, which broadcasts only certificates and short-term values, can prevent an adversary from mounting a 'partial key compromise impersonation attack' of the type described by Shim [136]. The adversary with A's long-term private key and A's (replaced) short-term private key can of course impersonate any entity in the group (by replacing that entity's short-term public key) to any other entity except A. Hence preventing such an adversary is equivalent to preventing an adversary from impersonating an entity with knowledge of that entity's long-term and short-term keys. This is of course impossible. Given that not all the entities compute a common shared key in a partial key compromise attack, key confirmation (using MACs and the shared key, for example) is sufficient to prevent this form of attack.

9.5.5 Unknown Key-Share Attacks

A basic source substitution attack is applicable to many protocols and applies to all our TAK protocols. Unknown key-share attacks were first discussed in Diffie *et al.* [59] and utilise a potential registration weakness for public keys to create fraudulent certificates [113]. Since we assumed that the certification process in the model of Section 9.3 is perfect, the security model does not capture these attacks. Unknown key share attacks are also known as source substitution attacks because the adversary forces an entity into believing that a message is from a different source to the actual source.

In a generic form of this attack, adversary E registers A's public key $K_{pub,A}$ as her own, creating Cert_E which is equal to $(\mathsf{ID}_E \| K_{pub,A} \| \Sigma(\mathsf{ID}_E \| K_{pub,A}), K_{priv,\text{CA}})$. Then, she intercepts A's message $aP \| \text{Cert}_A$ and replaces Cert_A with her own certificate Cert_E. Note that E registered A's long-term public key xP as her own without knowing the value of x. Therefore she cannot learn the key K_{ABC}. However, B and C are fooled into thinking they have agreed a key with E, when in fact they have agreed a key with A. They will interpret any subsequent encrypted messages emanating from A as coming from E. This basic attack could be eliminated if the CA does not allow two entities to register the same long-term public key. However, this solution may not scale well to large or distributed systems. A better solution is discussed after highlighting another type of source substitution attack.

A second, less trivial source substitution attack can be found against TAK-3. In this attack E certifies a related public key without knowing the corresponding private key. Thus, even if the CA does the previous check, the adversary can still obtain a Cert_E from the CA, which contains a component $K_{pub,E}$ which is a multiple of $K_{pub,A}$. The adversary can then alter the short-term keys in subsequent protocol messages by appropriate multiples. As with the last attack, the adversary does not create the shared key. Rather, the attack gives her the ability to fool two participants, B and C, into believing the messages came from her rather then from the third (honest) participant, A. This attack is presented next.

9.5.5.1 Source Substitution Attack on TAK-3

We present in detail the second source substitution attack on TAK-3.

1. A sends $aP \| \text{Cert}_A$ to $E_{B,C}$.
2. E computes $K_{pub,E} = \delta xP$ and obtains a certificate Cert_E on $K_{pub,E}$.
3. E initiates a run of protocol TAK-3 by sending $aP \| \text{Cert}_E$ to B, C.
4. B sends $bP \| \text{Cert}_B$ to E, C; C sends $cP \| \text{Cert}_C$ to E, B.
5. B and C (following the protocol) compute $K_{EBC} = K_{AE_{B,C}} = \hat{e}(P,P)^{(\delta xy)c+(\delta xz)b+(yz)a}$.
6. E_B sends $\delta bP \| \text{Cert}_B$ to A.
7. E_C sends $\delta cP \| \text{Cert}_C$ to A.
8. A (following the protocol) computes a key $K_{AE_{B,C}} = K_{AE_{B,C}} = \hat{e}(P,P)^{(\delta xy)c+(\delta xz)b+(yz)a} = K_{EBC}$.
9. Now E forwards A's messages encrypted under key $K_{EBC} = K_{AE_{B,C}}$ to B and C, and fools them into believing that A's messages come from her.

This attack does not seem to apply to TAK-1 or TAK-4 because of the way in which long-term private key components are separated from the short-term components in K_{ABC} in TAK-1 and due to the use of a hash function in TAK-4.

The adversary in our attacks *does not* know her long term private key. Unlike the more severe attack by Kaliski [93] on the MQV protocol, all these source substitution attacks are easily prevented if the CA insists that each registering party provides a PoP of his private key when registering a public key. This can be achieved using a variety of methods. For example, one might use zero-knowledge techniques or regard the pair $\langle x, xP \rangle$ as an ECDSA signature key pair and have the registering party sign a message of the CA's choice using this key pair. As an alternative, to prevent all forms of unknown key share attacks whilst maintaining a single round protocol, the protocol could be modified using one of the two methods:

1. Identities could be included in the key derivation function such that $K'_{ABC} = \mathrm{KDF}(K_{ABC} \| \mathsf{ID}_A \| \mathsf{ID}_B \| \mathsf{ID}_C)$. This ensures that the parties will create different keys when under attack.

2. Entity A could identify B and C within the exchanged message, for example by signing the identities ID_B and ID_C. Entities B and C should verify A's signature and identify entities A, C and A, B respectively within their messages. This ensures that the misrepresented parties will be detected when under attack. This solution is unattractive due to the computational complexity it could incur. Hence, our one round protocols avoid the use of signatures.

Method (2) is used within the key confirmation protocols presented in Section 9.8, making them resistant to unknown key-share attacks.

9.5.6 Insider and Other Attacks

Certain new kinds of insider attacks must be considered when dealing with tripartite protocols. For example, an insider A might be able to fool B into believing that they have participated in a protocol run with C, when in fact C has not been active. An active A can do this easily in our TAK protocols, by choosing C's value cP and injecting it into the network along with C's certificate and stopping B's message from reaching C. This kind of attack can have serious consequences for tripartite protocols: if C acts as an on-line escrow agent, then B believes a shared key has been properly escrowed with C when in fact it has not. On the other hand, when C acts as an off-line escrow agent, as in the protocol we describe in Section 9.7, this insider attack is only significant if C is providing an auditing function. This lack of auditability is actually beneficial if protocol participants want to have a deniability property, as in protocols like IKE, IKEv2 and JFK [84].

Attacks of this type can be prevented in a number of ways. Adding a confirmation phase, as we do in Section 9.8, prevents them. An alternative requires a complete protocol re-design, but maintains a one round broadcast protocol: the long-term keys

are simply used to sign short-term values (rather than combining them with short-term keys as in our TAK protocols) and agree the key $\hat{e}(P,P)^{abc}$. This approach requires each participant to maintain a log of all short-term values used, or to use synchronized clocks and time-stamps, to prevent an attacker simply replaying an old message. This requirement along with the need to create and verify signatures for each protocol run makes this solution somewhat unwieldy.

We note that Shim [136] has also found a rather theoretical 'known-key conspiracy attack' on TAK-2. This appears to be what we have called a known session key attack, but it requires two adversaries to conspire and to reveal three session keys. A similar attack can also be found on TAK-3, but is easily prevented if TAK-3 is augmented with a key derivation function.

In addition to the above attacks, we note that TAK-3 is vulnerable to a triangle attack of the type introduced by Burmester [41]. This atack is described next. It is somewhat theoretical in nature and thus presented only for completeness.

9.5.6.1 Triangle Attack on TAK-3

The triangle attack on TAK-3 allows an adversary E (who has a certificate Cert_E containing $K_{pub,E} = \Delta P$) to compute a session key K_{ABC} previously shared by the honest parties A, B and C.

1. E eavesdrops to obtain aP, bP and cP from the session in which the session key $K_{ABC} = \hat{e}(P,P)^{(xy)c+(xz)b+(yz)a}$ is agreed between A, B, C.

2. E now initiates three protocol runs. The first one is:
   $E \rightarrow B, C$: $aP \| \text{Cert}_E$ (1^α)
   $B \rightarrow E, C$: $b'P \| \text{Cert}_B$ (2^α)
   $C \rightarrow E, B$: $c'P \| \text{Cert}_C$ (3^α)
   The session key agreed is $K_{EBC} = \hat{e}(P,P)^{(\Delta y)c'+(\Delta z)b'+(yz)a}$.

3. The second run is:
   $E \rightarrow A, C$: $bP \| \text{Cert}_E$ (1^{β})
   $A \rightarrow E, C$: $a''P \| \text{Cert}_A$ (2^{β})
   $C \rightarrow A, E$: $c''P \| \text{Cert}_C$ (3^{β})
   The session key agreed is $K_{AEC} = \hat{e}(P,P)^{(x\Delta)c''+(xz)b+(\Delta z)a''}$.

4. And lastly:
   $E \rightarrow A, B$: $cP \| \text{Cert}_E$ (1^{γ})
   $A \rightarrow B, E$: $a'''P \| \text{Cert}_A$ (2^{γ})
   $B \rightarrow A, E$: $b'''P \| \text{Cert}_B$ (3^{γ})
   The agreed session key is $K_{ABE} = \hat{e}(P,P)^{(xy)c+(x\Delta)b'''+(y\Delta)a'''}$.

5. E now obtains three session keys K_{EBC}, K_{AEC} and K_{ABE} from B or C, A or C and A or C respectively. For this reason, this attack is regarded as somewhat theoretical.

6. Finally, session key

$$\begin{aligned} K_{ABC} &= K_{EBC} \cdot \hat{e}(P,P)^{-(\Delta y)c'-(\Delta z)b'} \\ &\cdot K_{AEC} \cdot \hat{e}(P,P)^{-z\Delta c'''-\Delta z a'''} \\ &\cdot K_{ABE} \cdot \hat{e}(P,P)^{-x\Delta b'''-y\Delta a'''} \end{aligned}$$

   can now be computed by E.

This triangle attack is possible because of the algebraic relationship between the long and short term key components in K_{ABC}. It can be thwarted using appropriate key derivation. This attack does not work on TAK-1 because we cannot isolate individual short term key components (e.g. in step 2 we cannot isolate a from fresh components b' and c'). This type of attack is also eliminated in TAK-4 because of the binding of each entity's short and long-term key using a hash function.

	Joux	TAK-1	TAK-2	TAK-3	TAK-4
Implicit key authentication	No	Yes	No	Yes	Yes
Known session key secure	No	No	No	Yes$^{(i)}$	Yes
Perfect forward secrecy	n/a	Yes	No	No$^{(iii)}$	Yes
Key-compromise impersonation sec.	n/a	No	No	Yes$^{(iv)}$	Yes$^{(iv)}$
Unknown key-share secure	No	Yes$^{(v)}$	No	Yes$^{(vi)}$	Yes$^{(v)}$

Table 9.2: Comparison of security goals and attributes for one round tripartite key agreement protocols.

(i) Only when a key derivation function is used; see Section 9.5.6.

(iii) No forward secrecy when two long-term private keys are compromised, but still has forward secrecy if only one is compromised.

(iv) Note, however, Section 9.5.4 describes the inevitable 'partial key compromise impersonation attack' on this scheme.

(v) If the CA checks that public keys are only registered once, and if inconvenient use (vi).

(vi) If the CA verifies that each user is in possession of the long-term private key corresponding to his public key.

9.5.7 Security Summary

Table 9.2 compares the security attributes that we believe our protocols TAK-1, TAK-2, TAK-3 and TAK-4 to possess. We have also included a comparison with the 'raw' Joux protocol.

Based on this table and the analysis in Section 9.3, we recommend the use of protocol TAK-4 or protocol TAK-3 along with pre-computation (in the event that the use of a hash function is not desirable). If perfect forward secrecy is also required, then TAK-3 can be modified as described in Section 9.5.3. Protocol TAK-4 has the additional benefit of being the most computationally efficient of all our protocols. Of course, robust certification is needed for all of our TAK protocols in order to avoid unknown key-share attacks.

9.6 Shim's Tripartite Key Agreement Protocol

Here we will look at the shortcomings of Shim's [138] one round tripartite authenticated key agreement protocol based on pairings. We show that the protocol of [138] does not make mathematical sense.

9.6.1 Shim's Protocol

Shim's protocol [138] addresses the lack of authentication in Joux's protocol [90] by utilising certified long-term public keys. In the notation we use in this thesis, Shim's protocol can be described as follows. The certificates Cert_A, Cert_B, Cert_C as usual contain entity A, B and C's public keys $K_{pub,A} = xP$, $K_{pub,B} = yP$ and $K_{pub,C} = cP$ respectively. In Shim's protocol, which we reproduce below, short-term keys are $a, b, c \in \mathbb{Z}$ (actually, these should be chosen uniformly at random from $\mathbb{Z}_q^*$) which are selected by entities A, B and C respectively. The message flows of Shim's protocol are given in Figure 9.4.

Protocol Messages

1. $A \rightarrow B, C: \quad a \cdot xP \| \text{Cert}_A$
2. $B \rightarrow A, C: \quad b \cdot yP \| \text{Cert}_B$
3. $C \rightarrow A, B: \quad c \cdot zP \| \text{Cert}_C$

Figure 9.4: Shim's tripartite protocol.

Protocol description: An entity A computes $T_A = a \cdot K_{pub,A}$ and broadcasts it to B and C along with a certificate Cert_A containing his long-term public key $K_{pub,A}$. Corresponding values ($T_B = bK_{pub,B}$ and $T_C = cK_{pub,C}$) and certificates are broadcast by B and C to A, C and A, B respectively. According to [138], the shared

key computed by the three parties is

$$K_{ABC} = \mathrm{KDF}(\hat{e}(P,P)^{xyzabc \cdot \hat{e}(P,P)^{xyz}} \| \mathsf{ID}_A \| \mathsf{ID}_B \| \mathsf{ID}_C),$$

where KDF is a key derivation fuction. Entity A computes this K_{ABC} by first of all computing the elliptic curve component $\hat{e}(P,P)^{xyzabc \cdot e(P,P)^{xyz}}$ by calculating $K_A = \hat{e}(T_B, T_C)^{ax \cdot \hat{e}(K_{pub,B}, K_{pub,C})^x}$. Entities B and C are meant to perform similar calculations.

9.6.2 The Problem

The problem with the protocol of [138] is that the definition of the shared key K_{ABC} does not make mathematical sense. This is because calculation of the key K_{ABC} requires raising a finite field element $\hat{e}(P,P)^{xyzabc}$ (i.e. $\hat{e}(T_B, T_C)^{xa}$ for entity A) to the power of another finite field element $\hat{e}(P,P)^{xyz}$. No such operation involving field elements is possible. It is only possible to raise a finite field element to an integer power. The finite field element $\hat{e}(P,P)^{xyz}$ cannot be used for this purpose. So the protocol of [138] is mathematically nonsensical. Even though it makes no mathematical sense, this protocol has been cryptanalysed by Sun and Hsieh in [144].

Of course, a number of alternative approaches to securing Joux's protocol [90] are presented previously in this chapter.

9.7 Tripartite Protocols with One Off-line Party

As we mentioned in the introduction, there is an application of tripartite key exchange protocols to the two-party case where one of the parties acts as an escrow agent. It may be more convenient that this agent be off-line, meaning that he receives messages but is not required to send any messages. In this section, we adapt our earlier protocols to this situation. The protocol below is a modified version of TAK-4. We assume that C is the off-line party and that C's certificate Cert_C is

pre-distributed or is readily available to A and B. The message flows of this protocol are given in Figure 9.5.

Protocol Messages		
1.	$A \rightarrow B, C:$	$aP \| \text{Cert}_A$
2.	$B \rightarrow A, C:$	$bP \| \text{Cert}_B$

Figure 9.5: Off-line TAK protocol.

Protocol description: The protocol in Figure 9.5 is as in TAK-4, but without the participation of C. Entities A and B use C's long-term public key zP in place of his short-term public value cP when calculating the session key. Thus, the session key is

$$K_{ABC} = \hat{e}(P, P)^{(a+H(aP\|xP)x)(b+H(bP\|yP)y)(z+H(zP\|zP)z)}.$$

Let θ be the output of $H(zP\|zP)$, which is publicly computable and always the same value. Therefore, in the above key $(z + H(zP\|zP)z)$ can be set to $(1 + \theta)z$. Alternatively, the value $(z + H(zP\|zP)z)$ in the session key can be optimised to z without affecting the security. Thus, the session key becomes

$$K_{ABC} = \hat{e}(P, P)^{(a+H(aP\|xP)x)(b+H(bP\|yP)y)(z)}.$$

Note that C can compute the key when required.

This protocol (whether optimised or not) appears to be resistant to all the previous attacks except the simple source substitution attack which is easily prevented via robust registration procedures. It also has forward secrecy, obviously except when the private key z is compromised. Here z can be viewed as an independent master key, which can be regularly updated along with the corresponding certificate.

Interestingly, the two party Diffie-Hellman based protocols presented in Figure 9.1 can be transformed into tripartite protocols with one off-line party. This is done by:

1. Changing the protocol setting to one appropriate for elliptic curve pairings protocols.

2. Both A and B additionally including entity C as a recipient of the protocol messages.

3. Including entity C's public key in the pairing map. This is done by A replacing elliptic curve Diffie-Hellman computations of the form $K_{priv,A} \cdot K_{pub,B}$ with computations of the form $\hat{e}(K_{pub,C}, K_{pub,B})^{K_{priv,A}}$ where $K_{priv} \in \mathbb{Z}_q^*$ and $K_{pub} \in \mathbb{G}_1^*$.

This evidently can be veiwed as a method of providing escrow to many two party Diffie-Hellman based protocols. For example, it is easy to see that the key computed by participants in an 'escrowable' MTI/A0 protocol is $K_{ABC} = \hat{e}(P, P)^{(ay+bx)(z)}$, which is computable by A, B and C.

9.8 Non-Broadcast – Tripartite AKC Protocols

Up to this point we have considered protocols that are efficient in the broadcast setting: they have all required the transmission of one broadcast message per participant. As we mentioned when discussing communication complexity in Section 3.1, the number of broadcasts is not always the most relevant measure of a protocol's use of communications bandwidth. A good example is the basic broadcast Joux protocol, which offers neither authentication nor confirmation of keys and requires six passes in a non-broadcast network. In this section, we introduce a pairing based tripartite key agreement protocol that also requires six passes, but that offers both key confirmation and key authentication, thus, the protocol is an AKC protocol. We show that *any* such protocol requires at least six passes. We then compare our protocol to a tripartite version of the station-to-station protocol [59].

9.8.1 A Six Pass Pairing-Based AKC Protocol

Our notation in describing our pairing-based tripartite authenticated key agreement with key confirmation (TAKC) protocol is largely as before. Additionally, to simplify the notation $\Sigma_A(\sigma)$ denotes A's signature on the string σ (i.e. $\Sigma_A(\sigma) = \Sigma(\sigma, K_{priv,A})$). We assume now that the CA's certificate Cert_A contains A's signature verification key. Also $\mathcal{E}_K^{sym}(\sigma)$ denotes encryption of string σ using a symmetric algorithm and key K, and χ denotes the string $aP\|bP\|cP$. The message flows of this protocol are given in Figure 9.6.

Sequence of Protocol Messages

1. $A \to B: \quad aP\|\text{Cert}_A$
2. $B \to C: \quad aP\|\text{Cert}_A\|bP\|\text{Cert}_B$
3. $C \to A: \quad bP\|\text{Cert}_B\|cP\|\text{Cert}_C\|\mathcal{E}^{sym}_{K_{ABC}}(\Sigma_C(\mathsf{ID}_A\|\mathsf{ID}_B\|\chi))$
4. $A \to B: \quad cP\|\text{Cert}_C\|\mathcal{E}^{sym}_{K_{ABC}}(\Sigma_C(\mathsf{ID}_A\|\mathsf{ID}_B\|\chi))\|\mathcal{E}^{sym}_{K_{ABC}}(\Sigma_A(\mathsf{ID}_B\|\mathsf{ID}_C\|\chi))$
5. $B \to C: \quad \mathcal{E}^{sym}_{K_{ABC}}(\Sigma_A(\mathsf{ID}_B\|\mathsf{ID}_C\|\chi))\|\mathcal{E}^{sym}_{K_{ABC}}(\Sigma_B(\mathsf{ID}_A\|\mathsf{ID}_C\|\chi))$
6. $B \to A: \quad \mathcal{E}^{sym}_{K_{ABC}}(\Sigma_B(\mathsf{ID}_A\|\mathsf{ID}_C\|\chi))$

Figure 9.6: TAKC protocol from pairings.

Protocol description: In Figure 9.6, entity A initiates the protocol execution with message (*1*). After receiving message (*2*), entity C is able to calculate the session key $K_{ABC} = \hat{e}(P, P)^{abc}$. The same session key is calculated after receiving messages (*3*) and (*4*), by A and B respectively. Messages (*3*) and onwards contain signatures on the short-term values and identities in the particular protocol run. This provides key authenticity. These signatures are transmitted in encrypted form using the session key K_{ABC} and this provides key confirmation. The confirmations from C to B, A to C and B to A are piggy-backed and forwarded by the intermediate party in messages (*3*), (*4*) and (*5*) respectively. More properly, encryptions should use a key derived from K_{ABC} rather than K_{ABC} itself. The symmetric encryptions can be replaced by

appending MACs to the signatures with the usual safeguards.

If the expected recipients' identities were not included in the signatures, this protocol would be vulnerable to an extension of an attack due to Lowe [107]. This attack exploits an authentication error and allows a limited form of unknown key-share attack. To perform it, we assume adversary E has control of the network. The attack is as follows.

1. E_C intercepts message (*2*), then E forwards (*2*) replacing Cert_B with Cert_E to C as if it originated from E. Thus, C assumes he is sharing a key with A and E.
2. E_A intercepts message (*3*) en route to A. Now E_C forwards this message replacing Cert_E with Cert_B to A.
3. Entity A receives (*3*) and continues with the protocol, sending message (*4*).
4. E blocks messages (*5*) and (*6*), so C and A assume an incomplete protocol run has occurred and terminate the protocol. However, on receipt of message (*4*), entity B already thinks he has completed a successful protocol run with A and C, whilst C might not even know B exists.

As usual in an unknown key-share attack, E cannot compute the shared key. The attack is limited because A and C end up with an aborted protocol run (rather than believing they have shared a key with B). The attack is defeated in our protocol because the inclusion of identities in signatures causes the protocol to terminate after message (*3*), when A realises that an illegal run has occurred.

We claim that no tripartite AKC protocol can use fewer than six passes, that is, no fewer than six messages exchanged in the protocol. Thus our protocol in Figure 9.6 is pass optimal. Our reasoning is as follows:

1. Each of the three entities must receive two short-term keys to construct K_{ABC}, so a total of six short-term values must be received.

2. The first pass can contain only one short-term key (the one known by the sender in that pass), while subsequent passes can contain two.

3. From 1 and 2, it can be seen that a minimum of four passes are needed to distribute all the short-term values to all of the parties. Therefore, only after at least four passes is the last party (entity B in our protocol) capable of creating the key.

4. This last party needs another two passes to provide key confirmation to the other two entities.

5. From 3 and 4, we see that at least six passes are needed in total.

Note that this argument holds whether the network supports broadcasts or not.

9.8.2 A Six Pass Diffie-Hellman based AKC Protocol

The STS protocol presented in Section 9.1.2.2 is a three pass, two-party AKC protocol designed by Diffie, van Oorschot and Wiener [59] to defeat man-in-the-middle attacks. Here we extend the protocol to three parties and six passes, a pass-optimal protocol according to the argument above.

An appropriate prime p and generator g mod p are selected. In Protocol 4 below, $a, b, c \in \mathbb{Z}_p^*$ are randomly generated short-term values and χ denotes the concatenation $g^a \| g^b \| g^c$. As before, we use the notation: $\mathcal{E}^{sym}_{K_{ABC}}(\cdot)$ and $\Sigma_A(\cdot)$. Again, we assume that authentic versions of signature keys are available to the three participants. In Figure 9.7, modulo p operations are omitted for simplicity of presentation.

Protocol description: The protocol in Figure 9.7 is similar in operation to the protocol presented in Figure 9.6, with additional computations performed before steps (*2*), (*3*) and (*4*) to compute g^{ab}, g^{bc}, and g^{ac} respectively. The shared session key is $K_{ABC} = g^{abc} \bmod p$.

Sequence of Protocol Messages

1. $A \rightarrow B: \quad g^a \| \mathrm{Cert}_A$
2. $B \rightarrow C: \quad g^a \| \mathrm{Cert}_A \| g^b \| \mathrm{Cert}_B \| g^{ab}$
3. $C \rightarrow A: \quad g^b \| \mathrm{Cert}_B \| g^c \| \mathrm{Cert}_C \| g^{bc} \| \mathcal{E}^{sym}_{K_{ABC}}(\Sigma_C(\mathsf{ID}_A \| \mathsf{ID}_B \| \chi))$
4. $A \rightarrow B: \quad g^c \| \mathrm{Cert}_C \| g^{ac} \| \mathcal{E}^{sym}_{K_{ABC}}(\Sigma_C(\mathsf{ID}_A \| \mathsf{ID}_B \| \chi)) \| \mathcal{E}^{sym}_{K_{ABC}}(\Sigma_A(\mathsf{ID}_B \| \mathsf{ID}_C \| \chi))$
5. $B \rightarrow C: \quad \mathcal{E}^{sym}_{K_{ABC}}(\Sigma_A(\mathsf{ID}_B \| \mathsf{ID}_C \| \chi)) \| \mathcal{E}^{sym}_{K_{ABC}}(\Sigma_B(\mathsf{ID}_A \| \mathsf{ID}_C \| \chi))$
6. $B \rightarrow A: \quad \mathcal{E}^{sym}_{K_{ABC}}(\Sigma_B(\mathsf{ID}_A \| \mathsf{ID}_C \| \chi))$

Figure 9.7: Diffie-Hellman based TAKC protocol.

9.8.3 Analysis of AKC Protocols

Two immediate conclusions can be drawn from our work in Sections 9.8.1 and 9.8.2. Firstly, we have given a pairing-based, tripartite AKC protocol using just the same number of passes as are needed in Joux's protocol (but with the penalty of introducing message dependencies). Secondly, this AKC version of Joux's protocol is no more efficient in terms of passes than a 3-party version of the STS protocol! Thus, when one considers confirmed protocols in a non-broadcast environment, the apparent advantage that Joux's protocol enjoys disappears. Of course, there is a two round broadcast version of the TAKC protocol (requiring 6 broadcasts and 12 passes). Both of our six pass AKC protocols can be performed in 5 rounds in a broadcast environment.

9.9 Summary

We have taken Joux's one round tripartite key agreement protocol and used it to construct one round TAK protocols. We have considered security proofs and heuristic

security analysis of our protocols, as well as an off-line version of our protocols. We have preserved the innate communications efficiency of Joux's protocol while enhancing its security functionality. We described why Shim's protocol which was intended to enhance the security functionality of Joux's protocol does not work. We have also considered tripartite variants of the STS protocol, suited to non-broadcast networks, showing that in the non-broadcast case, pairing-based protocols can offer no communication advantage over more traditional Diffie-Hellman style protocols. This investigation provides the reader with some intuition into the advantages of Joux's protocol and the security difficulties which accompany it due to its three party nature.

Future work should consider the security of our protocols in more robust models, capturing a larger set of realistic attacks. Constructing multi-party key agreement protocols using our TAK protocols as a primitive might result in bandwidth-efficient protocols. Finally, it would be interesting to see if the methods of [39] could be emulated in the setting of pairings to produce TAK protocols secure in the standard model.

Bibliography

[1] C. Adams and S. Lloyd. *Understanding Public-Key Infrastructure – Concepts, Standards, and Deployment Considerations*. Macmillan Technical Publishing, Indianapolis, USA, 1999.

[2] L.M. Adleman. The function field sieve. In L.M. Adleman and M.A. Huang, editors, *Proceedings of Algorithmic Number Theory Symposium – ANTS I*, volume 877 of *Lecture Notes in Computer Science*, pages 108–121. Springer-Verlag, 1994.

[3] S.S. Al-Riyami and C.J. Mitchell. Renewing cryptographic timestamps. In B. Jerman-Blazic and T. Klobucar, editors, *Communications and Multimedia Security*, volume 228 of *IFIP Conference Proceedings*, pages 9–16. Kluwer, 2002.

[4] S.S. Al-Riyami and K.G. Paterson. Authenticated three party key agreement protocols from pairings. Cryptology ePrint Archive, Report 2002/035, 2002. `http://eprint.iacr.org/`.

[5] S.S. Al-Riyami and K.G. Paterson. Authenticated three party key agreement protocols from pairings. In K.G. Paterson, editor, *Proceedings of 9th IMA International Conference on Cryptography and Coding*, volume 2898 of *Lecture Notes in Computer Science*, pages 332–359. Springer-Verlag, 2003.

[6] S.S. Al-Riyami and K.G. Paterson. Certificateless public key cryptography. Cryptology ePrint Archive, Report 2003/126, 2003. `http://eprint.iacr.org/`, full version of [7].

[7] S.S. Al-Riyami and K.G. Paterson. Certificateless public key cryptography (extended abstract). In C.S. Laih, editor, *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473. Springer-Verlag, 2003.

[8] American National Standards Institute – ANSI X9.42. Public key cryptography for the financial services industry: Agreement of symmetric keys using discrete logarithm cryptography, 2001.

[9] American National Standards Institute – ANSI X9.63. Public key cryptography for the financial services industry: Key agreement and key transport using elliptic curve cryptography, 2001.

[10] R. Ankney, D. Johnson, and M. Matyas. The Unified Model – contribution to X9F1, October 1995.

[11] J. Baek and Y. Zheng. Identity-based threshold decryption. Cryptology ePrint Archive, Report 2003/164, 2003. `http://eprint.iacr.org/`.

[12] P.S.L.M. Barreto, H.Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In M. Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–368. Springer-Verlag, 2002.

[13] P.S.L.M. Barreto, B. Lynn, and M. Scott. Constructing elliptic curves with prescribed embedding degrees. In S. Cimato, C. Galdi, and G. Persiano, editors, *Security in communication networks – SCN 2002*, volume 2576 of *Lecture Notes in Computer Science*, pages 263–273. Springer-Verlag, 2002.

[14] R. Barua, R. Dutta, and P. Sarkar. An n-party key agreement scheme using bilinear map. Cryptology ePrint Archive, Report 2003/062, 2003. `http://eprint.iacr.org/`.

[15] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science – FOCS 1997*, pages 394–403. IEEE Computer Society Press, 1997.

[16] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO 1998*, volume 1462 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.

[17] M. Bellare and S. Goldwasser. Lecture notes on cryptography. Summer course on "Cryptography and Information Security" at MIT, 2001. `http://www.cs.ucsd.edu/users/mihir/papers/gb.html`.

[18] M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 268–286. Springer-Verlag, 2004.

[19] M. Bellare and A. Palacio. Protecting against key exposure: Strongly key-insulated encryption with optimal threshold. Cryptology ePrint Archive, Report 2002/064, 2002. `http://eprint.iacr.org/`.

[20] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer-Verlag, 2000.

[21] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[22] M. Bellare and P. Rogaway. Entity authentication and key distribution. In D.R. Stinson, editor, *Advances in Cryptology – CRYPTO 1993*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer-Verlag, 1994.

[23] M. Bellare and P. Rogaway. Optimal asymmetric encryption – how to encrypt with RSA. In A. De Santis, editor, *Advances in Cryptology – EUROCRYPT 1994*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1994.

[24] M. Bellare and P. Rogaway. Provably secure session key distribution: The three party case. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing STOC*, pages 57–66. ACM, 1995.

[25] I.F. Blake, G. Seroussi, and N.P. Smart. *Elliptic curves in cryptography*. Cambridge University Press, Cambridge, 1999.

[26] S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In *Proceedings of the 6th IMA International Conference on Cryptography and Coding*, volume 1355 of *Lecture Notes in Computer Science*, pages 30–45. Springer-Verlag, 1997.

[27] S. Blake-Wilson and A. Menezes. Authenticated Diffie-Hellman key agreement protocols. In S. Tavares and H. Meijer, editors, *5th Annual Workshop on Selected Areas in Cryptography (SAC 1998)*, volume 1556 of *Lecture Notes in Computer Science*, pages 339–361. Springer-Verlag, 1998.

[28] A. Boldyreva. Efficient threshold signature, multisignature and blind signature schemes based on the gap-Diffie-Hellman-group signature scheme. In Y. Desmedt, editor, *International Workshop on Practice and Theory in Public Key Cryptography – PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer-Verlag, 2003.

[29] D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer-Verlag, 2004.

[30] D. Boneh and X. Boyen. Short signatures without random oracles. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer-Verlag, 2004.

[31] D. Boneh, X. Ding, G. Tsudik, and M. Wong. A method for fast revocation of public key certificates and security capabilities. In *proceedings of the 10th USENIX Security Symposium*, pages 297–308. USENIX, 2001.

[32] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In J. Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer-Verlag, 2001.

[33] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM J. Computing*, 32(3):586–615, 2003. `http://www.crypto.stanford.edu/~dabo/abstracts/ibe.html`, full version of [32].

[34] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verfiably encrypted signatures from bilinear maps. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer-Verlag, 2003.

[35] D. Boneh, I. Mironov, and V. Shoup. A secure signature scheme from bilinear maps. In M. Joye, editor, *Topics in Cryptology – CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 98–110. Springer-Verlag, 2003.

[36] D. Boneh, H. Shacham, and B. Lynn. Short signatures from the Weil pairing. In C. Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer-Verlag, 2001.

[37] C. Boyd. Towards extensional goals in authentication protocols. In *Proceedings of the 1997 DIMACS Workshop on Design and Formal Verification of Security Protocols*, 1997. `http://www.citeseer.nj.nec.com/boyd97towards.html/`.

[38] X. Boyen. Multipurpose identity-based signcryption : A swiss army knife for identity-based cryptography. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 383–399. Springer-Verlag, 2003. For the full version of this paper see, `http://eprint.iacr.org/2003/163`.

[39] E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic group Diffie-Hellman key exchange under standard assumptions. In L.R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 321–336. Springer-Verlag, 2002.

[40] A. Buldas, H. Lipmaa, and B. Schoenmakers. Optimally efficient accountable time-stamping. In Y. Zheng and H. Imai, editors, *International Workshop on Practice and Theory in Public Key Cryptography – PKC 2000*, volume 1751 of *Lecture Notes in Computer Science*, pages 293–305. Springer-Verlag, 2000.

[41] M. Burmester. On the risk of opening distributed keys. In Y. Desmedt, editor, *Advances in Cryptology – CRYPTO 1994*, volume 839 of *Lecture Notes in Computer Science*, pages 308–317. Springer-Verlag, 1994.

[42] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *Proceedings of the 13th Annual ACM Symposium on the Theory of Computing*, pages 209–218. ACM, 1993.

[43] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271. Springer-Verlag, 2003.

[44] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In C. Cachin and J. Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222. Springer-Verlag, 2004. `http://eprint.iacr.org/2003/182`.

[45] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer-Verlag, 2001.

[46] R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. In L.R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 337–351. Springer-Verlag, 2002.

[47] J.C. Cha and J.H. Cheon. An identity-based signature from gap Diffie-Hellman groups. In Y. Desmedt, editor, *Public Key Cryptography – PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30. Springer-Verlag, 2002.

[48] L. Chen, K. Harrison, A. Moss, D. Soldera, and N.P. Smart. Certification of public keys within an identity based system. In A.H. Chan and V.D. Gligor, editors, *Information Security, 5th International Conference, ISC*, volume 2433 of *Lecture Notes in Computer Science*, pages 322–333. Springer-Verlag, 2002.

[49] L. Chen, K. Harrison, D. Soldera, and N.P. Smart. Applications of multiple trust authorities in pairing based cryptosystems. In G.I. Davida, Y. Frankel, and O. Rees, editors, *Infrastructure Security, International Conference, InfraSec*, volume 2437 of *Lecture Notes in Computer Science*, pages 260–275. Springer-Verlag, 2002.

[50] L. Chen and C. Kudla. Identity based authenticated key agreement from pairings. In *IEEE Computer Security Foundations Workshop – CSFW-16 2003*, pages 219–233. IEEE Computer Society Press, 2003.

[51] X. Chen, F. Zhang, and K. Kim. A new ID-based group signature scheme from bilinear pairings. Cryptology ePrint Archive, Report 2003/116, 2003. `http://eprint.iacr.org/`.

[52] Z. Chen. Security analysis on Nalla-Reddy's ID-based tripartite authenticated key agreement protocols. Cryptology ePrint Archive, Report 2003/103, 2003. `http://eprint.iacr.org/`.

[53] J.H. Cheon. A universal forgery of Hess's second ID-based signature against the known-message attack. Cryptology ePrint Archive, Report 2002/028, 2002. `http://eprint.iacr.org/`.

[54] C. Cocks. An identity based encryption scheme based on quadratic residues. In B. Honary, editor, *Proceedings of 8th IMA International Conference on Cryptography and Coding*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer-Verlag, 2001.

[55] D. Coppersmith. Evaluating logarithms in GF(2^n). In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing STOC*, pages 201–207. ACM, 1984.

[56] J. Dankers, T. Garefalakis, R. Schaffelhofer, and T. Wright. Public key infrastructure in mobile systems. *IEE Electronics and Commucation Engineering Journal*, 14(5):180–190, 2002.

[57] Y. Desmedt and J. Quisquater. Public-key systems based on the difficulty of tampering. In A.M. Odlyzko, editor, *Advances in Cryptology – CRYPTO 1986*, volume 263 of *Lecture Notes in Computer Science*, pages 111–117. Springer-Verlag, 1986.

[58] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[59] W. Diffie, P.C. van Oorschot, and M. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2:107–125, 1992.

[60] X. Ding and G. Tsudik. Simple identity-based cryptography with mediated rsa. In M. Joye, editor, *Topics in Cryptology – CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 193–210. Springer-Verlag, 2003.

[61] Y. Dodis, M. Franklin, J. Katz, A. Miyaji, and M. Yung. Intrusion-resilient public-key encryption. In M. Joye, editor, *Topics in Cryptology – CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 19–32. Springer-Verlag, 2003.

[62] Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-insulated public key cryptosystems. In L.R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 65–82. Springer-Verlag, 2002.

[63] Y. Dodis and M. Yung. Exposure-resilience for free: The hierarchical ID-based encryption case. In *Proceedings of the First International IEEE Security in Storage Workshop*, pages 45–52. IEEE Computer Society Press, 2002.

[64] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *SIAM Journal of Computing*, 30(2):391–437, 2000.

[65] R. Dupont and A. Enge. Provably secure non-interactive key distribution based on pairings. In *Proceedings of the International Workshop on Coding and Cryptography – WCC 2003*, pages 165–174, 2003. To appear in Discrete Applied Mathematics.

[66] R. Dupont, A. Enge, and F. Morain. Building curves with arbitrary small MOV degree over finite prime fields. Cryptology ePrint Archive, Report 2002/094, 2002. `http://eprint.iacr.org/`.

[67] I. Duursma and H. Lee. Tate-pairing implementations for tripartite key agreement. Cryptology ePrint Archive, Report 2003/053, 2003. `http://eprint.iacr.org/`.

[68] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithm. In G.R. Blakley and D. Chau, editors, *Advances in Cryptology – CRYPTO 1984*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer-Verlag, 1985.

[69] G. Frey, M. Müller, and H. Rück. The Tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Transactions on Information Theory*, 45(5):1717–1719, 1999.

[70] E. Fujisaki and T. Okamoto. How to enhance the security of public-key encryption at minimum cost. In H. Imai and Y. Zheng, editors, *International Workshop on Practice and Theory in Public Key Cryptography – PKC 1999*, volume 1560 of *Lecture Notes in Computer Science*, pages 53–68. Springer-Verlag, 1999.

[71] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M.J. Wiener, editor, *Advances in Cryptology – CRYPTO 1999*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer-Verlag, 1999. `http://citeseer.nj.nec.com/fujisaki99secure.html`.

[72] M. Gagné. Identity-based encryption: A survey. *RSA Laboratories Cryptobytes*, 6(1):10–19, 2003.

[73] S.D. Galbraith. Supersingular curves in cryptography. In C. Boyd, editor, *Proceedings of AsiaCrypt 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 495–513. Springer-Verlag, 2001.

[74] S.D. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In C. Fieker and D.R. Kohel, editors, *Algorithmic Number Theory 5th International Symposium, ANTS-V*, volume 2369 of *Lecture Notes in Computer Science*, pages 324–337. Springer-Verlag, 2002.

[75] S.D. Galbraith, H.J. Hopkins, and I.E. Shparlinski. Secure Bilinear Diffie-Hellman bits. Cryptology ePrint Archive, Report 2002/155, 2002. `http://eprint.iacr.org/`.

[76] C. Gentry. Certificate-based encryption and the certificate revocation problem. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 272–293. Springer-Verlag, 2003.

[77] C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In Y. Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer-Verlag, 2002.

[78] M. Girault. Self-certified public keys. In D.W. Davies, editor, *Advances in Cryptology – EUROCRYPT 1991*, volume 547 of *Lecture Notes in Computer Science*, pages 490–497. Springer-Verlag, 1992.

[79] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[80] R. Granger, A.J. Holt, D. Page, N.P. Smart, and F. Vercauteren. Function field sieve in characteristic three. In D.A. Buell, editor, *Proceedings of Algorithmic Number Theory Symposium – ANTS VI*, volume 3076 of *Lecture Notes in Computer Science*, pages 223–234. Springer-Verlag, 2004.

[81] P. Gutmann. PKI: It's not dead, just resting. *IEEE Computer*, 35(8):41–49, 2002.

[82] S. Han, K.Y. Yueng, and J. Wang. Undeniable signatures from pairings over elliptic curves. In *ACM Conference on Electronic Commerce – EC 2003*, pages 262–263. ACM, 2003.

[83] F. Hess. Efficient identity based signature schemes based on pairings. In K. Nyberg and H. Heys, editors, *Selected Areas in Cryptography 9th Annual International Workshop, SAC 2002*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324. Springer-Verlag, 2003.

[84] P. Hoffman. Features of proposed successors to IKE. Internet Draft, `ftp://ftp.ietf.org/internet-drafts/draft-ietf-ipsec-soi-features-01.txt%`, 2002.

[85] J. Horwitz and B. Lynn. Towards hierarchical identity-based encryption. In L.R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481. Springer-Verlag, 2002.

[86] D. Hühnlein, M. Jacobson, and D. Weber. Towards practical non-interactive public key cryptosystems using non-maximal imaginary quadratic orders. In D.R. Stinson and S.E. Tavares, editors, *Selected Areas in Cryptography – SAC 2000*, volume 2012 of *Lecture Notes in Computer Science*, pages 275–287. Springer-Verlag, 2000.

[87] IEEE P1363. Standard specifications for public key cryptography, 2000. `http://grouper.ieee.org/groups/1363/index.html`.

[88] International Organization for Standardization. *ISO/IEC FCD 18014-1, Information technology — Security techniques — Time-stamping services — Part 1: Framework*, September 2001.

[89] ISO/IEC 15946-3. Information technology – security techniques – cryptographic techniques based on elliptic curves – part 3: Key establishment, awaiting publication.

[90] A. Joux. A one round protocol for tripartite Diffie-Hellman. In W. Bosma, editor, *Proceedings of Algorithmic Number Theory Symposium – ANTS IV*,

volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer-Verlag, 2000.

[91] A. Joux and R. Lercier. The function field sieve is quite special. In *Algorithmic Number Theory 5th International Symposium, ANTS-V*, volume 2369 of *Lecture Notes in Computer Science*, pages 431–445. Springer-Verlag, 2002.

[92] A. Joux and K. Nguyen. Separating decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. Cryptology ePrint Archive, Report 2001/003, 2001. `http://eprint.iacr.org/`.

[93] B. Kaliski, Jr. An unknown key-share attack on the MQV key agreement protocol. *ACM Transactions on Information and Systems Security*, 4(3):275–288, 2001.

[94] J. Katz. A forward secure public-key encryption scheme. Cryptology ePrint Archive, Report 2002/060, 2002. `http://eprint.iacr.org/`.

[95] M. Kim and K. Kim. A new identification scheme based on the bilinear Diffie-Hellman problem. In L. Batten and J. Seberry, editors, *Information Security and Privacy, Seventh Australasian Conference – ACISP*, volume 2384 of *Lecture Notes in Computer Science*, pages 362–378. Springer-Verlag, 2002.

[96] M. Kim and K. Kim. A new identification scheme based on the gap Diffie-Hellman problem. *SCIS 2002: The 2002 Symposium on Cryptography and Information Security Shirahama, Japan*, 2002.

[97] N. Koblitz. *Algebraic Aspects of Cryptography*. Algorithms and Computation in Mathematics. Springer-Verlag, 1999.

[98] C. Kudla. Identity-based cryptography and related applications. Master's thesis, Royal Holloway University of London, 2002.

[99] L. Law, A. Menezes, M. Qu, J. Solinas, and S.A. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28(2):119–134, 2003. `http://www.cacr.math.uwaterloo.ca/techreports/1998/tech_reports98.html`.

[100] B. Lee and K. Kim. Self-certificate: PKI using self-certified key. *Conference on Information Security and Cryptology 2000 – CISC 2000*, 10(1):65–73, 2000. `http://citeseer.nj.nec.com/483476.html`.

[101] B. Lee and K. Kim. Self-certified signatures. In A. Menezes and P. Sarkar, editors, *Progress in Cryptology – INDOCRYPT 2002*, volume 2551 of *Lecture Notes in Computer Science*, pages 199–214. Springer-Verlag, 2002.

[102] B. Libert and J.-J. Quisquater. Efficient revocation and threshold pairing based cryptosystems. In *Symposium on Principles of Distributed Computing – PODC 2003*, pages 163–171, 2003.

[103] B. Libert and J.-J. Quisquater. Efficient signcryption with key privacy from gap diffie-hellman groups. In F. Bao, R.H. Deng, and J. Zhou, editors, *International Workshop on Practice and Theory in Public Key Cryptography – PKC 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 187–200. Springer-Verlag, 2004. See `http://eprint.iacr.org/2003/023` for the full version.

[104] C. H. Lim and P. J. Lee. A key recovery attack on discrete log-based schemes using a prime order subgroup. In B.S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 249–263. Springer-Verlag, 1997.

[105] C.-Y. Lin and T.-C. Wu. An identity-based ring signature scheme from bilinear pairings. Cryptology ePrint Archive, Report 2003/117, 2003. `http://eprint.iacr.org/`.

[106] C.-Y. Lin, T.-C. Wu, and F. Zhang. A structured multisignature scheme from the gap Diffie-Hellman group. Cryptology ePrint Archive, Report 2003/090, 2003. `http://eprint.iacr.org/`.

[107] G. Lowe. Some new attacks upon security protocols. In *PCSFW: Proceedings of The 9th Computer Security Foundations Workshop*, pages 162–169. IEEE Computer Society Press, 1996.

[108] B. Lynn. Authenticated identity-based encryption. Cryptology ePrint Archive, Report 2002/072, 2002. `http://eprint.iacr.org/`.

[109] J. Malone-Lee. Identity-based signcryption. Cryptology ePrint Archive, Report 2002/098, 2002. `http://eprint.iacr.org/`.

[110] T. Matsumoto, Y. Takashima, and H. Imai. On seeking smart public-key-distribution systems. *Transactions on IECE of Japan*, E69:99–106, 1986.

[111] U. Maurer and Y. Yacobi. Non-interactive public-key cryptography. In D.W. Davies, editor, *Advances in Cryptology – EUROCRYPT 1991*, volume 547 of *Lecture Notes in Computer Science*, pages 498–507. Springer-Verlag, 1991.

[112] A. Menezes, T. Okamoto, and S. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.

[113] A. Menezes, M. Qu, and S. Vanstone. Some new key agreement protocols providing mutual implicit authentications. *2nd Workshop on Selected Areas in Cryptography (SAC 1995)*, pages 22–32, May 1995.

[114] A. Menezes, P.C. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, 1997.

[115] C. Mitchell, M. Ward, and P. Wilson. Key control in key agreement protocols. *Electronics Letters*, 34:980–981, 1998.

[116] S. Mitsunari, R. Sakai, and M. Kasahara. A new traitor tracing. *IEICE Transactions on Fundamentals*, E85-A(2):481–484, 2002.

[117] A. Muzereau, N.P. Smart, and F. Vercauteren. The equivalence between the DHP and DLP for elliptic curves used in practical applications. *LMS Journal Computation and Mathematics*, 7:50–72, 2004. `http://www.lms.ac.uk/jcm/7/lms2003-034/`.

[118] D. Nalla and K.C. Reddy. ID-based tripartite authenticated key agreement protocols from pairings. Cryptology ePrint Archive, Report 2003/004, 2003. `http://eprint.iacr.org/`.

[119] D. Nalla and K.C. Reddy. Signcryption scheme for identity-based cryptosystems. Cryptology ePrint Archive, Report 2003/066, 2003. `http://eprint.iacr.org/`.

[120] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing STOC*, pages 427–437. ACM, 1990.

[121] E. Okamoto. Key distribution systems based on identification information. In C. Pomerance, editor, *Advances in Cryptology – CRYPTO 1987*, volume 293 of *Lecture Notes in Computer Science*, pages 194–202. Springer-Verlag, 1987.

[122] K.G. Paterson. Cryptography from pairings: a snapshot of current research. *Information Security Technical Report*, 7(3):41–54, 2002.

[123] K.G. Paterson. ID-based signatures from pairings on elliptic curves. *Electronics Letters*, 38(18):1025–1026, 2002.

[124] H. Petersen and P. Horster. Self-certified keys – concepts and applications. In *Third International Conference on Communications and Multimedia Security*, pages 102–116. Chapman and Hall, 1997. `http://citeseer.nj.nec.com/petersen97selfcertified.html`.

[125] B. Preneel, B. Van Rompay, J.-J. Quisquater, H. Massias, and J. Serret Avila. Design of a timestamping system. Technical report, TIMESEC, Katholieke Universiteit Leuven and Université Catholique de Louvain, 1998.

[126] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attacks. In J. Feigenbaum, editor, *Advances in Cryptology – CRYPTO 1991*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444. Springer-Verlag, 1991.

[127] K.C. Reddy and D. Nalla. Identity based authenticated group key agreement protocol. In A. Menezes and P. Sarkar, editors, *Advances in Cryptology – INDOCRYPT 2002*, volume 2551 of *Lecture Notes in Computer Science*, pages 215–233. Springer-Verlag, 2003.

[128] P. Rogaway and T. Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision-resistance, 2004. `http://www.cs.ucdavis.edu/~rogaway/papers/index.html`. To appear in Fast Software Encryption (FSE) 2004.

[129] A. Roscoe. Intensional specifications of security protocols. In *Proceedings 9th IEEE Computer Security Foundations Workshop*, pages 28–38. IEEE Computer Society Press, 1996.

[130] S. Saeednia. Identity-based and self-certified key-exchange protocols. In V. Varadharajan, J. Pieprzyk, and Y. Mu, editors, *Information Security and Privacy, Second Australasian Conference – ACISP*, volume 1270 of *Lecture Notes in Computer Science*, pages 303–313. Springer-Verlag, 1997.

[131] S. Saeednia. A note on Girault's self-certified model. *Information Processing Letters*, 86:323–327, 2003.

[132] R. Sakai and M. Kasahara. ID based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054, 2003. `http://eprint.iacr.org/`.

[133] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *The 2000 Symposium on Cryptography and Information Security*, Okinawa, Japan, January 2000.

[134] M. Scott. Authenticated ID-based key exchange and remote log-in with insecure token and PIN number. Cryptology ePrint Archive, Report 2002/164, 2002. `http://eprint.iacr.org/`.

[135] A. Shamir. Identity-based cryptosystems and signature schemes. In G.R. Blakley and D. Chaum, editors, *Advances in Cryptology – CRYPTO 1984*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, 1984.

[136] K. Shim. Cryptanalysis of Al-Riyami-Paterson's authenticated three party key agreement protocols. Cryptology ePrint Archive, Report 2003/122, 2003. `http://eprint.iacr.org/`.

[137] K. Shim. Efficient ID-based authenticated key agreement protocol based on Weil pairing. *Electronics Letters*, 39(8):653–654, 2003.

[138] K. Shim. Efficient one round tripartite authenticated key agreement protocol from Weil pairing. *Electronics Letters*, 39(2):208–209, 2003.

[139] K. Shim. A man-in-the-middle attack on Nalla-Reddy's ID-based tripartite authenticated key agreement protocol. Cryptology ePrint Archive, Report 2003/115, 2003. `http://eprint.iacr.org/`.

[140] V. Shoup. On formal models for secure key exchange. IBM Technical Report RZ 3120, 1999. `http://shoup.net/papers`.

[141] J. Silverman. *The Arithmetic of Elliptic Curves*. Number 106 in Graduate Texts in Mathematics. Springer-Verlag, 1986.

[142] N.P. Smart. An identity based authenticated key agreement protocol based on the Weil pairing. *Electronics Letters*, 38(13):630–632, 2002.

[143] N.P. Smart. Access control using pairing based cryptography. In M. Joye, editor, *Topics in Cryptology – CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 111–121. Springer-Verlag, 2003.

[144] H.-M. Sun and B.-T. Hsieh. Security analysis of Shim's authenticated key agreement protocols from pairings. Cryptology ePrint Archive, Report 2003/113, 2003. `http://eprint.iacr.org/`.

[145] H. Tanaka. A realization scheme for the identity-based cryptosystem. In C. Pomerance, editor, *Advances in Cryptology – CRYPTO 1987*, volume 293 of *Lecture Notes in Computer Science*, pages 341–349. Springer-Verlag, 1987.

[146] S. Tsuji and T. Itoh. An ID-based cryptosystem based on the discrete logarithm problem. *IEEE Journal on Selected Areas in Communication*, 7(4):467–473, 1989.

[147] E.R. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 195–210. Springer-Verlag, 2001.

[148] E.R. Verheul. Self-blindable credential certificates from the Weil pairing. In C. Boyd, editor, *Proceedings of AsiaCrypt 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 533–551. Springer-Verlag, 2001.

[149] ITU-T Recommendation X.509. Information technology— open systems interconnection — the directory: Public-key and attribute certificate frameworks, 2000.

[150] F. Zhang and K. Kim. ID-based blind signature and ring signature from pairings. In Y. Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 533–547. Springer-Verlag, 2002.

[151] F. Zhang, S. Liu, and K. Kim. ID-based one round authenticated tripartite key agreement protocol with pairings. Cryptology ePrint Archive, Report 2002/122, 2002. `http://eprint.iacr.org/`.

[152] F. Zhang, R. Safavi-Naini, and C.-Y. Lin. New proxy signature, proxy blind signature and proxy ring signature schemes from bilinear pairing. Cryptology ePrint Archive, Report 2003/104, 2003. `http://eprint.iacr.org/`.

[153] F. Zhang, R. Safavi-Naini, and W. Susilo. Attack on Han *et al.*'s ID-based confirmer (undeniable) signature at ACM-EC'03. Cryptology ePrint Archive, Report 2003/129, 2003. `http://eprint.iacr.org/`.

[154] Z.-F. Zhang, J. Xu, and D.-G. Feng. Attack on an identification scheme based on gap Diffie-Hellman problem. Cryptology ePrint Archive, Report 2003/153, 2003. `http://eprint.iacr.org/`.

[155] Y. Zheng. Digital signcryption or how to achieve cost (signature & encryption) << cost(signature) + cost(encryption). In B.S. Kaliski Jr., editor, *Advances*

in Cryptology – CRYPTO 1997, volume 1294 of *Lecture Notes in Computer Science*, pages 165–179. Springer-Verlag, 1997.