Certificateless Authenticated Two-Party Key Agreement Protocols

Tarjei K. Mandt

Master's Thesis
Master of Science in Information Security
30 ECTS
Department of Computer Science and Media Technology
Gjøvik University College, 2006

Institutt for
informatikk og medieteknikk
Høgskolen i Gjøvik
Postboks 191
2802 Gjøvik

Department of Computer Science
and Media Technology
Gjøvik University College
Box 191
N-2802 Gjøvik
Norway

Abstract

Certificateless public key cryptography (CL-PKC) was proposed to overcome the weaknesses of the public key infrastructure (PKI) and identity-based cryptography (ID-PKC). In PKI, certificates are used to provide the authenticity of public keys. However, a PKI faces many challenges in practice, such as the scalability of the infrastructure and certificate management (distribution, revocation, storage, and validation costs). ID-PKC does not use certificates, but employs a key generation center (KGC) that will know every user's private key. Hence, the KGC will also be able to trace each user transaction and may cause loss of privacy if it's not trusted. In CL-PKC, on the other hand, the KGC does not have this information. Thus, CL-PKC is often considered a cross between PKI and ID-PKC.

In their seminal paper on CL-PKC, Al-Riyami and Paterson (AP) proposed a certificateless authenticated key agreement protocol. Key agreement protocols are one of the fundamental primitives of cryptography, and allow two or more parties to establish secret keys securely in the presence of an eavesdropping adversary. AP's protocol, the only certificateless key agreement protocol proposed so far, essentially requires each party to compute four bilinear pairings. Such pairings can be computationally intensive to compute, and should therefore be used moderately in protocols.

In this thesis, we propose a new certificateless authenticated two-party key agreement protocol that only requires each party to compute two pairings. We perform a security analysis and heuristically argue that the protocol obtains the desired security attributes. We also show that our protocol can be used to establish keys between members of distinct domains (under different KGCs). Finally, we compare the protocol's efficiency to current identity-based and certificateless protocols.

Sammendrag

Utfordringen i dag ved å utvikle sikre systemer basert på offentlig-nøkkel kryptografi er ikke det å velge tilstrekkelig sikre algoritmer og implementere disse, men heller å utvikle en infrastruktur som forsikrer brukere om tilhørigheten av offentlige nøkler. I tradisjonell offentlig-nøkkel infrastruktur (PKI) løses dette ved bruk av sertifikater, hvor en tiltrodd tredjepart (Certificate Authority) signerer den offentlige nøkkelen. Det finnes imidlertid en rekke problemer ved bruk av sertifikater, deriblant tilbaketrekking, lagring, distribusjon og valideringskostnader. Identitetsbasert kryptografi (ID-PKC) benytter seg ikke av sertifikater, men er avhengig av en tiltrodd tredjepart (Private Key Generator) som til enhver tid kjenner til alle brukeres private nøkler. En PKG kan derfor forfalske signaturer og utgi seg for å være hvilken som helst bruker i systemet.

Sertifikatløs offentlig-nøkkel kryptografi (CL-PKC) er et relativt nytt konsept som forsøker å løse problemene ved PKI og identitetsbasert kryptografi. I likhet med ID-PKC benytter det seg ikke av sertifikater, men den tiltrodde tredjeparten (Key Generation Center) kjenner ikke til hver enkelt brukers private nøkkel. Derfor er man ikke lenger nødt til å ha like stor tillit til den tiltrodde tredjeparten som i ID-PKC.

I dette prosjektet har vi utviklet en sertifikatløs nøkkelutvekslingsprotokoll. En nøkkelutvekslingsprotokoll tillater to eller flere brukere å bli enige om en felles nøkkel over et åpent nett. Vi argumenterer for at protokollen er sikker og har bedre ytelse enn den som ble utviklet av Al-Riyami og Paterson [1]. Vi viser dessuten hvordan protokollen kan fungere i et miljø der brukere er underlagt forskjellige tiltrodde tredjeparter (KGCer).

Aknowledgements

I would like to thank my supervisor Dr. Chik How Tan, who provided excellent guidance and offered me many hours every month of his precious time. Without the many interesting discussions and his help, I can honestly say that this thesis would not have been the same. I would also like to thank my opponent, Sjur Ringheim Lid, for providing valuable feedback on the preliminary draft of the thesis.

Moreover, I would like to thank my family and friends for being supportive throughout the course of the thesis. I would also like to thank my girlfriend Hilde for being patient with me during the rather hectic project period.

Tarjei K. Mandt, June 30, 2006

Contents

List of Figures

List of Tables

1 Introduction

1.1 Topic

The challenge today in developing secure systems based on public key cryptography is not choosing appropriately secure algorithms and implementing these, but rather developing an infrastructure to support the authenticity of a user's public key. In the traditional public key infrastructure (PKI), certificates are used to provide an assurance of the relationship between public keys and the identities that hold the corresponding private keys. However, a PKI faces many challenges in practice, such as the scalability of the infrastructure and certificate management. To address the shortcomings of PKI and to simplify key management, Shamir [54] proposed the notion of identity-based public key cryptography (ID-PKC) in which the public keys are derived from the users' identities, such as a username or an e-mail address. Private keys are generated by a trusted third party called a Private Key Generator (PKG), and thus ID-PKC eliminates the need for certificates.

Unfortunately, ID-PKC is not without problems. The dependence on a PKG that uses a system-wide master key to generate private keys introduces problems such as key escrow and trust. For instance, the PKG can decrypt any ciphertext from any user to which it has issued a key. Moreover, the PKG can forge any signature and masquarade as any user in the identity-based setting. In [1], Al-Riyami and Paterson proposed the concept of certificateless public key cryptography (CL-PKC) to address the key escrow limitation of ID-PKC. Yet, CL-PKC does not require the use of certificates and can thus be considered a cross between ID-PKC and PKI.

This thesis focuses on certificateless authenticated two-party key agreement protocols. Key agreement protocols are one of the fundamental primitives of cryptography, and allow two or more parties to establish secret keys securely in the presence of an eavesdropping adversary. A key agreement protocol is said to be authenticated if it offers the assurance that only the participating parties of the protocol can compute the agreed key.

1.2 Problem Description

There is always a need to improve the efficiency or security of a key agreement protocol. It is important to undertand that protocols are never perfect. Many times, proposed protocols are found to lack certain desirable properties or to be inefficient in some way. Over time, authors will always find new and clever ways to improve the efficiency or the security of protocols.

In their seminal paper on CL-PKC, Al-Riyami and Paterson (AP) proposed a certificateless authenticated key agreement protocol. Their protocol essentially requires each party to compute four bilinear pairings. Such pairings can be computationally intensive to compute (for instance, in low-power devices), and should therefore be used moderately in protocols. Moreover, their protocol also requires users to exchange public keys comprising two group elements. Ideally, public keys should only comprise one group element as in identity-based cryptography.

Due to these apparent shortcomings, it would be desirable to propose a new certificateless key agreement protocol that offers essentially the same security as AP's protocol, but with improved efficiency.

1.3 Justification, Motivation, and Benefits

The advantage and benefits of using a key agreement protocol based on CL-PKC is that there is no PKI and will therefore save communication costs. The solution may therefore be ideal in a wireless environment or in low-power devices where resources are limited. Moreover, a certificateless key agreement protocol does not have the property of key escrow inherent of ID-PKC. Thus, it may be more suited in a distributed environment (in which privacy is a requirement), whereas ID-based protocols seems more suited for smaller networks and closed groups.

1.4 Research Questions

In order to design a certificateless authenticated two-party key agreement protocol, certificateless public key cryptography and existing ID-based key agreement protocols must be carefully studied. It is furthermore needed to study the vulnerabilites of a number of key agreement protocols, including the Diffie-Hellman key exchange, and assess the attacks possible. Finally, known vulnerabilites such as man-in-the-middle attacks, can be prevented by using proper authentication methods. We will look at how this can be achieved in key agreement.

Ultimately, this leads us to the following research questions:

1. How are key agreement protocols designed in certificateless public key cryptography, and can existing identity-based schemes be adopted?
2. What are the possible attacks on a certificateless authenticated two-party key agreement protocol?
3. How is proper authentication achieved in certificateless key agreement protocols?
4. How does the efficiency and security of certificateless key agreement measure up against identity-based key agreement?

1.5 Method

In order to solve the research questions, we have used a qualitive approach in which existing literature has been studied. We have studied the concept of certificateless public key cryptography, as well as existing key agreement schemes (both identity-based and certificateless) and security models.

1.6 Summary of Claimed Contributions

The contributions of this thesis are as follows:

- A new certificateless authenticated key agreement protocol that is more efficient than the protocol of [1].
- A certificateless authenticated key agreement protocol that allows users of distinct domains (under different KGCs) to establish a shared secret.

- An assessment of security properties in certificateless key agreement and a suggestion for a certificateless key agreement security model.

1.7 Outline of Chapters

Chapter 2 provides an introduction to the algebra and definitions used throughout the thesis. Chapter 3 provides an introduction to modern cryptography, the use for certificateless public key crypgraphy, and key agreement protocols. In Chapter 4 we examine similar pairing-based key agreement protocols that relate to protocol proposed in this thesis. In Chapter 5 we propose a new certificateless authenticated two-party key agreement protocol and discuss its properties. Chapter 6 suggests further work and Chapter 7 gives a conclusion of the thesis.

2 Definitions

This chapter defines the mathematical basis and provides an introduction to elliptic curve theory and pairings and their properties. It also defines the cryptographic primitives used throughout this thesis.

2.1 Abstract Algebra

This section provides a basic introduction to groups and finite fields.

2.1.1 Groups

Definition 2.1.1. *A binary operation $*$ on a set $\mathbb{G}$ is a function that assigns to each pair of elements a and b in $\mathbb{G}$ a unique $a * b$ in $\mathbb{G}$. Binary operators may be of the form $*, \cdot, +, \circ, \oplus, \otimes$. An operation $a * b$ is said to be written in the multiplicative notation. A brinary operation written in the additive notation is denoted by a $+$.*

Definition 2.1.2. *A group $(\mathbb{G}, *)$ is a non-empty set $\mathbb{G}$ and a binary operation $*$ of which $G_1 * G_1 = G_2$ satisfies the following axioms.*

- *Associativity:* $\forall a, b, c \in \mathbb{G}, a * (b * c) = (a * b) * c$
- *Neutral element: There is an element e in $\mathbb{G}$ such that* $\forall a \in \mathbb{G}, e * a = a * e = a$
- *Inverse element: For each $a \in \mathbb{G}$, there is an inverse element a^{-1} such that $a * a^{-1} = e$*
- *Commutativity: If $\mathbb{G}$ is an Abelian group, then* $\forall a, b \in \mathbb{G}, a * b = b * a$.

The order of a group $\mathbb{G}$, usually denoted by $|\mathbb{G}|$, is the number of elements in the set $\mathbb{G}$. If $\mathbb{G}$ is a finite set, it is called a finite group.

Definition 2.1.3. *A group $\mathbb{G}$ is cyclic if there is an element $\alpha \in \mathbb{G}$ such that for each $b \in \mathbb{G}$ there is an integer i with $b = \alpha^i$. If α generates all elements of the group $(\mathbb{G}, *)$, then α is a generator of $\mathbb{G}$. The order of α equals to the order of the group it generates.*

Example 2.1.4. *$\mathbb{G}_7 = \{1, 2, 3, 4, 5, 6\}$. It can then be shown that 3 is a generator of $\mathbb{G}_7$ by $\langle 3 \rangle = \{3^0, 3^1, 3^2, 3^3, 3^4, 3^5\} = \{1, 3, 2, 6, 4, 5\} = \mathbb{G}_7$. The group $\mathbb{G}_7$ is therefore cyclic.*

The groups used in this thesis are represented by $\mathbb{Z}_n$, $\mathbb{G}_1$, and $\mathbb{G}_2$. The group $\mathbb{Z}_n$ denotes a set of integers $\{0, 1, ..., n-1\}$ with modulo-n addition such that n represents the number of elements in the group. A group donated by $\mathbb{Z}_n^*$ is a set of all elements in $\mathbb{Z}_n$ that have a multiplicative inverse such that $\mathbb{Z}_n^* = \{x \in \mathbb{Z}_n : \gcd(x, n) = 1\}$ (containing only non-identity elements). The additive group $\mathbb{G}_1$ and the multiplicative group $\mathbb{G}_2$ are cyclic groups of a large prime order related to elliptic curves over finite fields.

2.1.2 Finite Fields

A *finite field* $(F, +, \times)$ is a finite set of elements F and two binary operations with integer addition and multiplication. It has been proven by Galois that the size of the finite field (the number of elements it contains) must be a power m of a prime number q. There is exactly one finite field for any given size q^m, and this field is denoted by $\mathbb{F}_{q^m}$.

If $p = q^m$ where q is a prime and $m \in \mathbb{Z}_n$, then q is called the characteristic of $\mathbb{F}_q$ and m is called the extension degree of $\mathbb{F}_q$. Most schemes restrict the order of the field to be of an odd prime ($q = p$) or a power of 2 ($p = 2^m$).

Definition 2.1.5. *Let $\mathbb{F}_q$ be a finite field with a prime q. The field $\mathbb{F}_{q^m}$ with an integer $m > 1$ is known as an extension field of the subfield $\mathbb{F}_q$.*

Elements of an extension field are polynomials of degree less than m over $\mathbb{F}_q$. $\mathbb{F}_{q^m}$ is represented by a polynomial $a_{m-1}x^{m-1} + ... + a_1x + a_0$ where $a_0, a_1, ..., a_{m-1}$ are elements of the finite field $\mathbb{F}_q$.

2.2 Elliptic Curves

Many key agreement schemes and other cryptographic primitives build on pairings of elliptic curve groups. This section explains briefly the concept of elliptic curves and their properties. Most of the results come from [51].

An elliptic curve is traditionally defined as

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

The equation above generates elliptic curves over real numbers as points in coordinates. Calculations made using real numbers are slow and inaccurate due to the round-off error, and are therefore unsuited for cryptographic work. By limiting the coefficients of the curves to the element of finite fields (such as $\mathbb{F}_p$ and $\mathbb{F}_{2^m}$), one may generate curves using only integer points (Figure 2).

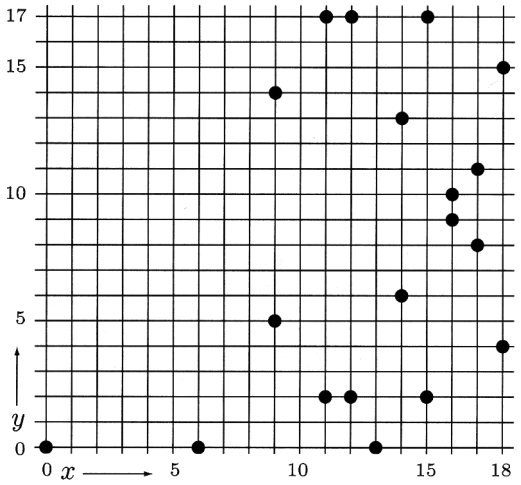

Figure 1: An Elliptic Curve over a Finite Field [51]

The finite field $E(\mathbb{F}_p)$ exists for any prime p. If $p > 3$ is an odd prime, the *short Weierstrass form* may be used

$$y^2 = x^3 + ax + b \tag{2.2}$$

To determine whether the elliptic curve E generates a group over a finite field $\mathbb{F}_p$, one

verifies that the curve contains no repeated factors or that the discriminant of the curve is nonzero such that

$$4a^3 + 27b^2 \pmod{p} \neq 0 \tag{2.3}$$

There are several differences between elliptic curves over real numbers and finite fields. Obviously, finite fields only contain a finite number of points in contrast to real elliptic curves that contain an infinite number of points. Moreover, the geometry of elliptic curves using real numbers cannot be applied to curves over finite fields. However, the algebraic rules for the arithmetic can be adapted for elliptic curves over finitie fields.

Definition 2.2.1. *For an extension field $\mathbb{K}$ of $\mathbb{F}$, a set $\{(x, y) \in \mathbb{K} \times \mathbb{K} : E(\mathbb{K})\} \cup \{\mathcal{O}\}$ can be used to form an elliptic curve group $E(\mathbb{K})$ under some group operation.*

The group operation called "point addition" of any two points $P \in E(\mathbb{K})$ and $Q \in E(\mathbb{K})$, selects either a point in $E(\mathbb{K})$ denoted by $P + Q$ or $\mathcal{O}$ (the point at infinity). The following properties hold:

1. $P + \mathcal{O} = \mathcal{O} + P = P$ for all $P \in E(\mathbb{K})$. Thus, $\mathcal{O}$ is the additive identity of the group.
2. Let $P = (x, y)$ and $Q = (x, -y)$. Then $Q = -P$ and $P + Q = P - P = \mathcal{O}$. Thus, the *inverse* of P is $-P$.
3. Let $P = (x, y)$ and $Q = (x', y')$. The sum of P and Q, denoted $-R$, is defined as follows. Let a line pass through P and Q and intersect the curve in the third point R. $P + Q$ is then the reflection of this point in the x-axis.
4. Let $P = (x, y)$ and $Q = (x', y')$. If $x = x'$, but $y \neq y'$, then $P + Q = \mathcal{O}$.
5. Let $P = (x, y)$. Then, the *point doubling* of P results in a point $-R$, defined as follows. Draw P's tangent and let it intersect the curve in a point R. The double of P then becomes R reflected in the x-axis, denoted by $-R$.
6. $E(\mathbb{K})$ is *commutative* because $(P + Q) + R = P + (Q + R)$ and *associative* because $P + Q = Q + P$.

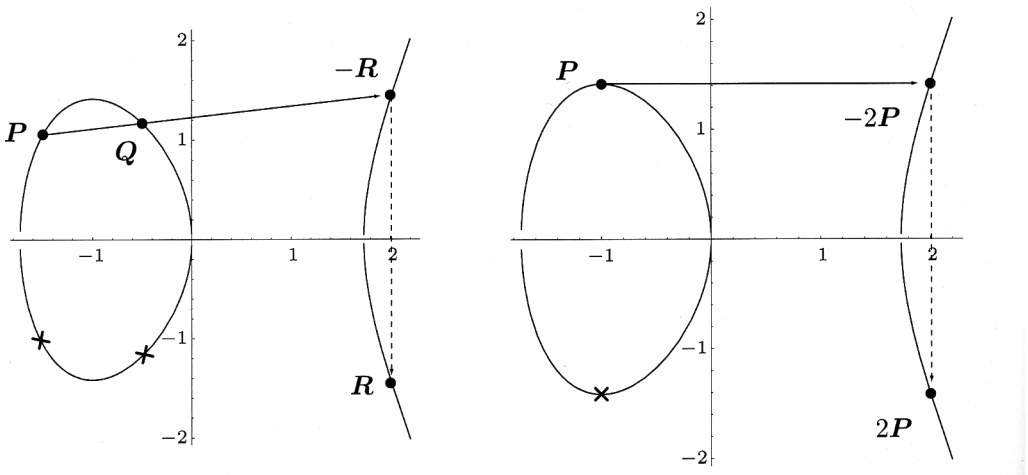

Figure 2: Point Addition on Elliptic Curves [51]

Scalar multiplication of a point P is denoted by mP, such that $P + ... + P = mP$. It is believed to be computaionally infeasible to reverse the operation (compute m from mP, P).

The number of points on an elliptic curve $\mathbb{E}(\mathbb{F}_{q^k})$ is called the order of the curve over the field $\mathbb{F}_{q^k}$.

Most work in this thesis makes use of pairings 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 $n \in \mathbb{Z}^*$.

2.3 Bilinear Maps on Elliptic Curve Groups

Bilinear maps are often called elliptic curve pairings because they associate pairs of elements from $\mathbb{G}_1$ with elements in $\mathbb{G}_2$. Let $\mathbb{G}_1 = \langle P \rangle$ be an additive group (identity $\mathcal{O}$) with prime order q and let $\mathbb{G}_2$ be a multiplicative group (identity 1) of the same order. A bilinear map on $(\mathbb{G}_1, \mathbb{G}_2)$ is then a function $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ that must satisfy the following properties.

1. *Bilinearity*: Given any $P, Q, R \in \mathbb{G}_1$, we have
   $\hat{e}(P, Q + R) = \hat{e}(P, Q) \cdot \hat{e}(P, R)$ and $\hat{e}(P + Q, R) = \hat{e}(P, R) \cdot \hat{e}(Q, R)$.
   Thus, for any $a, b \in \mathbb{Z}_q$:
   $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab} = \hat{e}(abP, Q) = \hat{e}(P, abQ)$.
2. *Non-degeneracy*: $\hat{e}(P, P) \neq 1$. If P is a generator for $\mathbb{G}_1$, then $\hat{e}(P, P)$ is a generator for $\mathbb{G}_2$.
3. *Computability*: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P, Q \in \mathbb{G}_1$.

Note that the bilinearity of pairings also implies that $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is symmetric. Thus, for any $Q, R \in \mathbb{G}_1$, the equality $\hat{e}(Q, R) = \hat{e}(R, Q)$ holds. Both $Q, R \in \mathbb{G}_1$ can be represented by some generator P such that $Q = aP$ and $R = bP$ where $a, b \in \mathbb{Z}$. Then it's followed that $\hat{e}(Q, R) = \hat{e}(aP, bP) = \hat{e}(P, P)^{ab} = \hat{e}(bP, aP) = \hat{e}(R, Q)$.

The map $\hat{e}$ may be computed using a Weil pairing [41] or a Tate pairing [25] on an elliptic curve over $\mathbb{F}_q$. In principle, the antisymmetry of the Weil pairing forces the two subgroups to be distinct. However, given a supersingular curve[1] one may define a modified Weil pairing on a single subgroup of order q using distortion maps introduced by Verheul [59]. Distortion maps (also called endomorphisms) makes it possible to send points from one subgroup of the l-torsion to another.

Of the two, the Weil pairing has simpler mathematic properties. However, it does not always reach the optimal value for r. Tate on the other hand, always reaches its optimal value.

2.4 Bilinear Diffie-Hellman and Related Problems

The computational problems introduced in this section provide the basis of security for pairing-based key agreement schemes and the proposed protocol. Many cryptographic primitives are based on number-theoretic problems. Two terms are frequently used in complexity theory in describing cryptographic problems and assumptions, namely *negligible function* and *polynomial time algorithm*.

Definition 2.4.1. *A negligible function* $g : \mathbb{N} \rightarrow \mathbb{R}$ *approaches zero faster than the reciprocal*

[1] A curve is called supersingular if $k \leq 6$ in the extension field $\mathbb{F}_{q^k}$.

of any polynomial. That is, for every $k \in \mathbb{N}$ *there is an integer* k_c *such that* $g(k) \leq k^{-c}$ *for all* $k \geq k_c$.

Cryptographic protocols require the adversary's advantage to be insignificant in guessing the solution to some problem. For instance, one might say that the adversary's success probability in recovering a session key is a negligible function of the *security parameter*. The security parameter, denoted by k in many cases, represents the complexity of the input problem. The value of k is important because it can adjust parameters such as the size of cryptographic groups and key lengths. The larger k is, the more computation is required by the algorithm.

Definition 2.4.2. *A polynomial time algorithm is an algorithm whose execution time of a computation* $m(k)$ *is no more than a polynomial function of the security parameter,* k. *More formally,* $m(k) = \mathcal{O}(k^c)$[2] *where* c *is a constant.*

A polynomial time algorithm is said to be efficient. If a problem can be solved by an algorithm $\mathcal{A}$ in polynomial time for at least a non-negligible fraction of all possible inputs, it is said that the problem is *tractable*. If no such algorithm exists, it is assumed that the problem is *intractable* and protocols that base on the problem are computationally secure.

The problems of the following sections are treated as intractable as their true computational complexities are unknown up to present.

2.4.1 The Classic Diffie-Hellman Problems

When using standard cryptographic groups, the security relies on three assumptions: the discrete logarithm problem, the computational Diffie-Hellman problem, and the decisional Diffie-Hellman problem. These problems can easily be applied to elliptic curve cryptography. Thus, the security of elliptic curve cryptosystems base on the intractibility of the following problems.

Definition 2.4.3 (Discrete Logarithm Problem)**.** *Given* $Q \in \mathbb{G}_1$ *where* P *is a generator of* $\mathbb{G}_1$*, find an element* $a \in \mathbb{Z}_q^*$ *such that* $aP = Q$.

Definition 2.4.4 (Computational Diffie-Hellman Problem)**.** *Given* $\langle P, aP, bP \rangle$ *in* $\mathbb{G}_1$ *where* $a, b \in \mathbb{Z}_q^*$*, compute* abP.

Definition 2.4.5 (Decisional Diffie-Hellman Problem)**.** *Given* $\langle P, aP, bP, cP \rangle$ *in* $\mathbb{G}_1$ *where* $a, b, c \in \mathbb{Z}_q^*$*, determine if* $abP = cP$.

If there exists a polynomial time algorithm $\mathcal{A}$ that can solve the discrete logarithm problem (DLP), then it can be used to solve the CDHP in polynomial time. The algorithm first computes a from aP and then computes $a(bP) = abP$. Moreover, if the algorithm can solve CDH, it can also solve the DDHP. Given a group element cP, $\mathcal{A}$ can determine whether $cP = abP$. Thus, the DDHP reduces to the CDHP which again reduces to the DLP. Hence, the security of a protocol is strongest if it reduces to the discrete logarithm problem.

2.4.2 The Bilinear Diffie-Hellman Problems

Pairing based cryptography (which may be considered as an extension of elliptic curve cryptography) introduces a new problem known as the Bilinear Diffie-Hellman Problem (BDHP), first formalized in [9].

[2]The big-$\mathcal{O}$ notation is used to represent the order of the asymptotic upper bound as the exact running time of the input algorithm is usually unknown.

Definition 2.4.6 (Bilinear Diffie-Hellman Problem)**.** *Let* $\hat{e}$ *be a bilinear pairing on* $(\mathbb{G}_1, \mathbb{G}_2)$ *and* P *be a generator of* $\mathbb{G}_1$*. Given* $\langle P, aP, bP, cP\rangle \in \mathbb{G}_1$ *with* $a, b, c \in \mathbb{Z}_q^*$*, compute* $\hat{e}(P,P)^{abc} \in \mathbb{G}_2$.

Similarily, the BDH assumption states that there exists no algorithm in expected polynomial time that can solve the BDH problem for $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}\rangle$ with non-negligible probability. It is easy to show that if the CDH problem is easy, then it is just as easy to solve BDHP. Given $aP, bP, cP \in \mathbb{G}_1$, the shared key abP may be computed using the CDH algorithm. It is then possible to compute $\hat{e}(abP, cP) = \hat{e}(P,P)^{abc}$ which is the solution to the BDH problem. It is also easy to solve the BDH problem if the CDH problem in $\mathbb{G}_2$ is easy. By first computing $g = \hat{e}(P,P)$ followed by $g^{ab} = \hat{e}(aP, bP)$ and $g^c = \hat{e}(P, cP)$, the shared key $g^{abc} = \hat{e}(P,P)^{abc}$ may be computed using the CDH algorithm.

Definition 2.4.7 (Decisional Bilinear Diffie-Hellman Problem)**.** *Let* $\hat{e}$ *be a bilinear pairing on* $(\mathbb{G}_1, \mathbb{G}_2)$ *and* P *be a generator of* $\mathbb{G}_1$*. Given* $\langle P, aP, bP, cP\rangle \in \mathbb{G}_1$ *with* $a, b, c \in \mathbb{Z}_q^*$ *and a random element* $Q \in G_2^*$*, determine if* $Q = e(P,P)^{abc}$.

If the BDHP is solved, then the DBDHP can also easily be solved. The solution of the BDH instance $\langle P, aP, bP, cP\rangle$ yields $R = \hat{e}(P,P)^{abc}$, and thus the solution of the DBDHP can be obtained by checking if $R = Q$ holds.

2.4.3 Implications of Bilinear Maps

The following are consequences of bilinear pairings, as pointed out by [40].

1. The discrete logarithm problem (DLP) in $\mathbb{G}_1$ is no harder than in $\mathbb{G}_2$. Given $P, Q \in \mathbb{G}_1$, find $a \in [0, n-1]$ such that $Q = aP$. Similarily in $\mathbb{G}_2$, find $a \in [0, n-1]$ such that $\hat{e}(P,Q) = \hat{e}(P, aP) = \hat{e}(P,P)^a$. Thus, the DLP of $\mathbb{G}_1$ can be reduced to the DLP of $\mathbb{G}_2$.
2. The decisional Diffie-Hellman problem (DDHP) in $\mathbb{G}_1$ is easy. Given $\langle P, aP, bP, cP\rangle$ in $\mathbb{G}_1$, it is possible to decide if $c \equiv ab \pmod n$. If $g = \hat{e}(P,P)$, then $\hat{e}(aP, bP) = \hat{e}(P,P)^{ab} = g^{ab}$ and $\hat{e}(P, cP) = \hat{e}(P,P)^c = g^c$. One may then prove $c \equiv ab \pmod n$ by verifying $g^{ab} = g^c$. The groups where DDH becomes easy while CDH remains hard are called gap groups.
3. Let $Q \in \mathbb{G}_1^*$. Then $f_Q : \mathbb{G}_1 \to \mathbb{G}_2$ defined by $f_Q(R) = \hat{e}(Q,R)$ is a group isomorphism. An algorithm to invert any f_Q will make the DDHP in $\mathbb{G}_2$ easy.
4. If the DDHP in $\mathbb{G}_2$ is hard, then the DLP in $\mathbb{G}_2$ may be harder than the DLP in $\mathbb{G}_1$.

2.5 Cryptographic Primitives

This section reviews the cryptographic primitives used throughout this thesis.

2.5.1 Hash Functions

One-way functions, or hash functions, are one of the fundamental primitives of cryptography. The hash function accepts as input an arbitrary-length message and scrambles the individual bits to generate a fixed-size output called the hash value. The basic idea of the hash function is to produce a unique fingerprint for any given input message.

Definition 2.5.1. *A hash function is a function* $h : \mathcal{D} \to \mathcal{R}$ *where the domain* $\mathcal{D} = \{0,1\}^*$ *and the range* $\mathcal{R} = \{0,1\}^n$ *for some* $n \geq 1$.

Hash functions are required to be one-way and collision resistant. A hash function H is said to be one-way if it's computationally infeasable to recover the message x from a

hash value $H(x)$. A collision resistant hash function implies that no two messages should generate the same output.

Definition 2.5.2 ([51]). *A secure hash function H has two properties: (1) if $z = H(x)$, then it is computationally infeasable to find a $y \neq x$ such that $z = H(y)$; (2) collisions are extremely rare (it is computationally infeasable to find any two arguements x and y that hash to the same z).*

Secure hash functions are commonly used in cryptographic protocols for digital signatures and data integrity. Signing hash values of documents is also more efficient than signing the document itself. As the flip of a bit in a message x will output a different hash, it is also easy to verify the integrity.

In key agreement, hash functions are often used as key derivation functions, denoted KDF, to enhance the security properties of a session key. Hash functions are also used in converting a user's indentifying information to a point on the underlying elliptic curve in identity-based and certificateless cryptography.

2.5.2 Message Authentication Codes

A message authentication code, or MAC, is a short piece of information used to protect a message's integrity and authenticity. While anyone can generate a hash of a given value, a MAC assumes that the generator and the verifier share a common secret. The MAC algorithm accepts as input a secret key and an arbitrary-length message to be authenticated, and generates a MAC as output.

Definition 2.5.3. *A message authentication code (MAC) is a function $h : \mathcal{K} \times \mathcal{M} \to \mathcal{R}$ where the key space $\mathcal{K} = \{0,1\}^k$, the message space $\mathcal{M} = \{0,1\}^*$, and the range $\mathcal{R} = \{0,1\}^n$ for some $n \geq 1$.*

To authenticate a message m, an entity with a pre-shared key k' computes $(m, a) = MAC_k(m)$ where a is the tag (a checksum) on m. To verify (m, a), a different entity checks that $MAC_{k'}(m)$ does indeed equal (m, a) using the same pre-shared key k'.

The main idea of the MAC is that an adversary without the knowledge of the key should be unable to forge a valid tag for a given message that has not yet been authenticated. A MAC must therefore be able to resist adaptive chosen-plaintext attacks in order to be considered secure. This implies that no two messages m and m' should yield the same MAC under some unknown key.

Message authentication codes share some similarities with conventional encryption, for instance in the way that communicating parties need a prior established shared key. However, the key is only used in a one-way function which build on the difficulty of computing certain mathematical primitives. This makes the MAC less vulnerable to attacks than regular encryption.

For a more comprehensive review of cryptographic MAC algorithms and hash functions, see [48].

2.6 Other Notation

The symbol $\|$ is often used to denote the concatenation of two strings. For instance, a hash function $H(k\|m) = H(k, m)$ gives the hash value of a message m computed under a key k. Functions denoted $\{0,1\}^* \to \mathbb{G}_1$ map an arbitrary-length binary message to an element of the group $\mathbb{G}_1$, and is often used by hash functions to represent the mapping

of a user identity to an elliptic curve integer point. $\{0,1\}^k$ denotes a binary message that is k bits in length.

3 Preliminary Topics

This chapter begins by discussing the properties and shortcomings of the public key infrastructure and identity-based cryptography, and moves on to explaining the concept of certificateless public key cryptography and its advantages. It then provides an introduction to key agreement protocols, and reviews the primitives used in such protocols today. Furthermore, the chapter provides a list of security goals for key agreement protocols and assesses potential vulnerabilities.

3.1 Public Key Cryptography

In modern cryptography, there's a difference between symmetric and asymmetric cryptosystems. In symmetric cryptography, there is only one secret key used between the sender and the receiver. Thus, the same key is used for both encryption and decryption. Symmetric primitives include block ciphers, stream ciphers, cryptographic hash functions, and message authentication codes (MACs). Common to all symmetric cryptosystems is that the parties who wish to communicate need some prior secret established before distributing keys. This is usually achieved by establishing a secure channel to a trusted authority (TA) who then issues a common secret key to both parties.

In comparison to symmetric-key cryptosystems, the main idea of asymmetric or public key cryptosystems is to make key distribution easier. It is asymmetric in the sense that each party has a key pair, a public key and a private key. Respectively, these are used for encryption and decryption. Although the keys have some cryptographic relation, the public key can be widely distributed without compromising the private key. Thus, any party who wish to communicate with another party can encrypt the message using the recipient's public key who then can decrypt it using his or her private key. It is also possible to sign data, in which the private key is used for signing and the public key is used for verification. Unlike symmetric-key cryptography, there is no need for having established a secret prior to interaction.

3.1.1 Public Key Infrastructure (PKI)

The notion of public key cryptography was first introduced by Whitfield Diffie and Martin Hellman in their 1976 seminal paper [21]. Realizing that the proposed public key directory had its shortcomings (both in regard to performance and availability), Loren Kohnfelder introduced the concept of certificates in 1978 [35]. The idea was to allow a certificate authority (CA) to bind a name to a key through a digital signature and store it in a repository. A few years later, certificates were incorporated into X.500, a hierarchical database model for the public key infrastructure (PKI). These certificates (X.509) were designed to address the access control issues of the X.500 directory.

The original motivation for PKI was to provide mechanisms for issuing, storing, and distributing public key certificates. Over the years, however, a number of problems [30, 29] with PKI have been discovered. One concerns the identity of the X.509 certificate and how to properly retrieve the desired key should the repository hold certificates with identical names (DN). It is possible to disambiguate names by adding uniquely identifi-

able strings or digits such as a user's Social Security number to the DN, but this again makes it trivial to perform name lookups for third parties. The fact that certificates are based on owner identity also becomes a problem if the owner changes affiliation, e-mail address, or name. Usually, an owner will have several certificates with the same identity. X.509 certificates are far less flexible than IDs used in the real world, and need to be replaced if the slighest change is made.

Another "flaw" of the public key infrastructure is the concept of certificate revocation lists (CRL), which basically are lists that hold revoked certificates. Unfortunately, the method of revocation can no longer meet the demands of today's real-time applications. Currently, CRLs must be issued by the CA to each underlying party who then is responsible to check if a certificate is revoked. Today, this process does not occur frequently enough to be effective against a compromised key. Furthermore, distributing CRLs may become an expensive task as each party must download the updated list, often every minute in order to provide a timely revocation. In other words, timeliness comes at the cost of computing resources. It can also be cumbersome if the CRL is very large, which is often the case for large PKIs. Downloading large CRLs over low-speed links may use excessive bandwidth and thus can cause network congestion. Checking for CRLs may also be time-consuming, and a denial-of-service-attack can easily render them ineffective.

Another problem relates to certificate chains and cross-certification. Certificates in the public key infrastructure are often arranged into a hierarchical trust model. This allows an end entity to be signed by a single CA depending on the degree of trust or privileges granted to that specific entity. To check the legitimacy of a certificate, a user obtains the signer's public key and verifies that the provided signature is valid. However, in order to check that a certificate wasn't simply forged, a user must also check the signer certificate and in turn all the certificates above up to an implicitly trusted root CA. Thus, we say that a user creates a certificate chain. The situation complicates when the end entity of a CA in one hierarchy wishes to authenticate the end entity of a CA in a different hierarchy. For this to be possible, the CAs cross-certify each other by signing each other's certificates. Not only does this lead to multiple paths from leaf to roots, but the semantics of entities change depending on the chosen path.

Apparently, these are just a few of the problems with PKI that have been brought up over the years. Fortunately, there are alternatives that may prove to be far more efficient and flexible than the current standard.

3.1.2 Identity-Based Cryptography

Identity-based cryptography (ID-PKC) was introduced by Shamir [54] in 1984, and enables communicating users to verify signed data without exchanging private or public keys, without managing certificates, and without having to rely on services provided by a third party. It assumes the existance of a private key generator (PKG) from which users are issued their private keys. Once all private keys have been issued, the PKG can be closed for an indefinite period while the network can continue to function as normal (as long as no additional users are introduced). This is because the system does not introduce key revocation as in traditional PKI, and therefore always assumes that keys are valid.

In the identity-based system, public keys are derived from a known identity such as the username or e-mail address, and thus may be generated by anyone. In order to

obtain the private key, an entity needs to present itself to a private key generator. The PKG combines its master key with the identity value of the challenging entity and generates the private key. It is crucial to the identity-based scheme that the PKG is trusted as it will know every user's private key, and thus be able to decrypt any message sent in its domain. This property is called *key escrow* and is by many considered a shortcoming of identity-based cryptography. However, there are also cases in which key escrow may be a needed property, such as in the health care profession where an audit trail to transactions may be a legal requirement.

Shamir's motivation in developing the identity-based cryptosystem was originally to simplify key management in e-mail systems. Because a user generally knows the e-mail address of the recipient, it implies that the user also would know the public key. Encrypting the message using the public key would require the recipient to obtain the corresponding private key from the PKG. The sender may also sign the message using the appropriate private key. Upon reciept, the receiver may easily verify a signature only by knowing the identity of the sender. It is also important to note that public and private keys are generated independently, unlike in traditional PKI.

Although the notion of identity-based cryptography is quite old and has co-existed with PKI for many years, it wasn't until recently when Boneh and Franklin [9] demonstrated the construction of very efficient and provably secure identity-based primitives using elliptic curve pairings that ID-PKC truly gained popularity. Since then, a host of new primitives have been proposed, including encryption schemes, key agreement protocols [52, 58, 15, 60, 39, 49], and signature schemes [10]. Currently, ID-PKC is a very active area of research.

Although identity-based cryptography makes certificates obsolete and has many desirable attributes, it has also its weaknesses. The inherent key escrow allows a PKG to decrypt any message and therefore forces its users to delegate an almost unacceptable amount of trust. Several schemes [15] have proposed solutions on how to remove the escrow property, but the methods used are inefficient and usually result in additional computation and communication overhead. ID-PKC also does not offer non-repudiation as the PKG may forge any signature. Furthermore, the compromise of the PKG master key would be disasterous in the identity-based setting. An adversary who is able to attain the PKG master key is able to masquarade as any entity. As trust is of utmost importance to ID-PKC, it seems more suited for small groups or closed environments rather than large infrastructures.

3.1.3 Certificateless Public Key Cryptography

In 2003, Al-Riyami and Paterson proposed the concept of certificateless public key cryptography (CL-PKC) [1]. In a way, CL-PKC combines the best of both worlds by still operating in a certificateless environment like ID-PKC, but using a trust model similar to that of PKI. Thus, CL-PKC does not inherit the escrow property of ID-PKC, making the system ideal for networks where privacy or user anonymity is preferred. Furthermore, the absence of certificates removes the cost incurred by certificate storage, distribution, and verification which makes CL-PKC far more efficient than traditional PKI.

CL-PKC still makes use of a trusted authority, but in contrast to ID-PKC, the *key generation center* (KGC) does not have access to the entities' private keys. Instead, the KGC generates a partial private key that the user then combines with a secret value. Together,

these values make up the actual private key, and thus the KGC cannot recover the shared secret established between entities. This change to the scheme also makes it impossible for the KGC to forge any signatures. The public key is generated in a similar way by letting the user combine its secret value with a public parameter selected by the KGC. However, since the secret value is only known to a specific user, public keys can no longer be generated by anyone as in ID-PKC. Thus, the scheme loses the benefit of identity-based key derivation. Consequently, public keys must be provided in some other way, such as through a public directory or by attatching them to messages in a protocol run.

Since the introduction of CL-PKC, many new papers have proposed improvements and fixes to the original scheme. However, most of these concern certificateless public key encryption (CL-PKE) and thus few new primitives (such as signature schemes and key agreement protocols) have been proposed. In [2], the original CL-PKE scheme of [1] was improved both in terms of efficiency and security. Later, [62] discovered an adaptive chosen ciphertext vulnerability and proposed a countermeasure to overcome the flaw. In [20], Dent and Kudla argues against a claim that the certificateless schemes cannot be proven secure in the standard model.

3.1.4 Trust Model

Girault [27] shows that public key cryptosystems essentially can be classified into three different *trust levels* depending on the trust assumption of the trusted third party (TTP).

- At trust level 1, the TTP knows the users' private keys and can therefore impersonate any user at any time in an undetectable way.
- At trust level 2, the TTP does not know the users' private keys, but can still impersonate users by generating false public keys.
- At trust level 3, the TTP does not know the users' private keys, and generating false public keys will expose the TTP's actions.

Due to the escrow property, it's easy to see that the trust level of CL-PKC is greater than that of ID-PKC. In PKI, whenever a CA tries forge a certificate, it can be identified by the fact that there are two working certificates for the same user. In CL-PKC, however, the TTP will still be able to replace public keys without the entities realizing that these are invalid. To address this and achieve trust level 3, CL-PKC also proposes an alternative key generation technique that binds a user identifier to a public key. Thus, the corresponding private key will be bound to the public key, and if the KGC replaces a public key it will easily be noticed. A minor drawback of this technique is that the public key must be generated before the private key is issued by the KGC.

3.2 Cryptographic Key Agreement Protocols

A *key agreement protocol* is a series of steps used by two or more parties in order to securely agree on a shared secret, such as a session key, in an unprotected network. Key agreement protocols differ from *key transport protocols* in which the whole key is transmitted over a secure channel from one entity to another. For instance, key transport and secure channels are used by trusted third parties such as a CA or a KGC in issuing private data to users. In key agreement, on the other hand, entities contribute information

jointly to establish a shared secret.

A protocol that establishes a shared key between two entities is called a *two-party* key agreement protocol. Sometimes it's also useful to consider three parties, and thus the protocol is called a *tripartite* key agreement protocol. If a protocol has more than three participants, it is called a *group* or *conference* key agreement protocol. Examples of tripartite and group key agreement protocols can be found in [32] and [22, 55] respectively.

Furthermore, if a key agreement protocol exchanges information between its participants, it is said to be *interactive*. It is also possible for a protocol to be *non-interactive*, although more usual in the identity-based setting in which public keys of entities are always known. Identity-based non-interactive protocols will always suffer from the session key escrow property and are thus rarely used in practice. For the remainder of this thesis, only interactive two-party key agreement protocols will be discussed.

3.2.1 Goals of Key Agreement

The fundamental goal of any key agreement protocol is to securely establish a common secret key by distributing keying data between two entities. Both entities should influence the outcome of the key, thus preventing an undesired third-party from injecting any weak keys on the agreeing parties. A key agreement protocol should be able to withstand both *active* attacks (in which an adversary injects, deletes, alters, or replays a message) and *passive* attacks (in which an adversary simply observes the protocol exchange and prevents it from acheiving its goals).

In [12], Boyd proposed a classification of design goals divided into intentional and extensional goals. Intentional goals are generally concerned with ensuring that the protocols run correctly as specified, while extensional goals are concerned about what a protocol is designed to achieve for its participants. In key agreement, the following extensional goals are desired.

- **Implicit Key Authentication**. A key agreement protocol provides implicit key authentication if an entity can be assured that none other than the intended entities *can* obtain the value of the secret key. A protocol in which this assurance is given to all participants is called an *authenticated key agreement protocol* (AK).
- **Explicit Key Authentication**. A protocol that provides explicit key authentication assures each participating entity that the intended other entities have actually computed the key. Such a protocol is also called an *authenticated key agreement with key confirmation* (AKC) protocol.
- **Good Key**. A *good key* should be selected uniformly at random from the key space and must esentially be non-predictable. No adversary should be able to guess the outcome of the key established by two entities using public information. Ensuring key freshness and the use of key derivation functions can help achieve the goal of good key.

Note that implicit authentication is used in key agreement instead of the term entity authentication. Implicit authentication does not only imply authentication of entities, but also assures that a key can be efficiently computed by these.

3.2.2 The Diffie-Hellman Key Exchange

The Diffie-Hellman (DH) key distribution algorithm [21] became the breakthrough of modern cryptography. Its security rests on the discrete logarithm assumption, which assumes that it's computationally difficult to solve discrete logarithms modulo very large primes. An eavesdropper who monitors the key exchange will not be able to predict the outcome of the shared key. This is also known as the Diffie-Hellman problem and thus the DH key exchange fulfills the goal of good key.

The algorithm has two public parameters; a prime number p and an integer less than p known as the generator, g. The generator may generate any element in $[1, p-1]$ when multiplied by itself enough times, modulo p. $\mathbb{G}$ is the finite cyclic group with prime order $|\mathbb{G}|$ generated by g. If Alice and Bob wish to agree on a secret key, Alice first chooses a number a (the private key) at random from $[1, p-1]$ and keeps it secret. She then computes the public key, P_a.

$$P_a = g^a \pmod{p}$$

Bob also chooses a value b in the same fashion and computes P_b.

$$P_b = g^b \pmod{p}$$

Alice and Bob now exchange public keys and may then compute the shared secret key using their private keys.

$$K = (P_a)^b = (P_b)^a \pmod{p}$$

If an eavesdropper, Eve, is to compute the key using the public values, she must solve the equation

$$K = P_a^{(\log_g P_b)} \pmod{p}$$

The above equation refers to what is known as the computational Diffie-Hellman (CDH) problem which states that it's hard to compute g^{ab} even with the knowledge of p, g, g^a, and g^b. However, CDH alone is not sufficient to ensure the security of Diffie-Hellman. Eve may still be able to predict a large amount of bits of g^{ab} with some confidence. If a shared secret key is to be derived from a block of bits from g^{ab}, it is necessary to assume that Eve cannot predict these bits using the known values g^a and g^b. Formally, this is known as the Decisionial Diffie-Hellman problem [8]. No algorithm should efficiently be able to distinguish between the two distributions $\langle g^a, g^b, g^{ab} \rangle$ and $\langle g^a, g^b, g^c \rangle$, in which g^c is randomly distributed in $\mathbb{G}$. Today, the most efficient method for solving the DDH problem is by computing descrete log to test that a triplet $\langle x, y, z \rangle$ satisfies the Diffie-Hellman relation.

Shared Key

The shared key generated by the Diffie-Hellman algorithm is rarely used directly for encryption. The key might not satisfy the amount of bits required for the encryption key and it's unclear how secure the bits in the DH key actually are. The most significant bits are provably secure [11], but it's unknown to what degree the rest of the bits are. A KDF may be used to destroy any algebraic relationship between keys. A KDF also prevents the use of static keys, which have been proven vulnerable to the Burmester triangle attack [13]. A pseudorandom function such as a secure hash function (i.e. SHA-1) or a MAC are ideal KDFs.

Authentication

As the Diffie-Hellman key exchange does not provide any authentication of parties or the exchanged information, the scheme is vulnerable to a *man-in-the-middle* attack. An adversary can thus easily break the protocol by intercepting g^x and g^y and replacing them with $g^{x'}$ and $g^{y'}$ respectively. Alice will then think the key is $g^{y'x}$ and Bob will believe the key is $g^{x'y}$. Although Alice and Bob now have different keys, both are known to the attacker. If Alice encrypts a message using her key in a secret-key cryptosystem and passes it along to Bob, the adversary can then intercept and decryt the message. The adversary then encrypts it using Bob's key, and passes it along to Bob. Effectively, the adversary has managed to break the encryption scheme.

The DH key exchange can be improved to provide implicit authentication by signing all the communication sent between parties. This is sometimes referred to as Signed Diffie-Hellman. Although the solution may sound ideal, signatures on DH keys present a number of problems. Moreover, signatures cause the messages to increase considerably in size, and are thus in many cases inconvenient.

Public Parameters

It is important that the public parameters (p, g) as well as the respective secret keys x and y are chosen with care in order to avoid attacks such as the *degenerate message attack*. If either g^x or g^y equals 1 (hence, the shared secret key also becomes 1), the protocol may be broken. Similarily, if x or y use simple values such as 1, the protocol may also be broken as g^x or g^y equals g. Furthermore, if a protocol is designed carelessly, an attacker may intercept g^x and g^y and replace them with 1. This will cause the shared secret key to also be 1. An attacker may also try to fool participants into using weak system parameters (p, g) if there is no authentication of these.

In order to thwart degenerate messages, protocol participants need to make sure that their key agreement peer does not send $g^z = 1$. This may be avoided by using an interval such as $[2, p-2]$. Primes should also be large enough to render current algorithms insufficient, such as the Pohlig-Hellman algorithm [45] which may compute the discrete log of g^x if the prime factorization of g's order consists of small primes. Strong primes may be acheived by using safe primes[1] or the slightly more efficient Lim-Lee primes[2]. If the exponent range is limited to a certain interval, for instance to improve efficiency, one should also be aware of the Pollard Lambda algorithm [46].

3.2.3 Protocol Attacks

In addition to the attacks listed above, a key agreement protocol may be susceptible to a number of attacks depending on how a session key is constructed and the power of the adversary. The following attacks have been identified in literature [47, 31].

- *Source substitution attack*. If the attacker has access to another user's public key and manage to obtain a certificate in his or her name for that key, the attacker may then masquerade as the other user in a number of different situations. This attack may be prevented by the CA requesting the private key before issuing a signed certificate, but it's recommended to avoid exposure to such attacks.

[1] $p = Rq + 1$, where R is a small positive value and q is a large prime
[2] $p = 2q_1 \cdot ... \cdot q_n + 1$, where q_i are large primes; see [38]

- *Key separation attack*. In the *key separation attack*, entity A may during an authentication protocol with entity B encrypt a message using the symmetric key K. If A also uses the same key in communicating with C in a different protocol, it is possible that C could exploit the lack of key separation to replace messages sent during the first protocol with messages from the second. This kind of attack may be avoided if the key K is replaced by a key derived from K, known as a *key derivation function*.
- *Time-memory trade off attack*. If the hashed version of a data stream is available, the attacker can determine the data by comparing the hash with pre-computed values. If we assume that data *K* contains *k* bits, and *h(K)* has been observed, the attacker pre-computes and stores 2^r values of *h(K)*. During the protocol run, the attacker compares h(K) with the pre-computed values, in which the probability of success is 2^{k-r}. The attacker will therefore need to compare equally many values for every capture of *K*.
- *Known key attacks*. In a known key attack, old session keys that are compromised will also compromise future session keys. Should the attacker acquire a session key from a past session, it may be used to exchange messages in a different session. Known key attacks come in a few flavors:

  - In a *key reveal attack* [61], the adversary exploits the algebraic relationship between keys by using the session key of one session to obtain the session key of another session. Specifically, the attacker has access to a key reveal oracle which can reveal an old session key that has been previously accepted, and from it, derive something from the other established session key.
  - In a *key-replication attack*, the attacker manages to obtain the key of a session by finding a different session that generates a key identical to that session. A protocol may be vulnerable to such an attack even if it uses a collision-free hash function to derive its keys.

- *Forgery attack*. It should not be possible for an adversary to forge a session key using known protocol parameters or information exchanged between entities. Forgery in DH-based protocols, requires the adversary to solve a computational DH problem which is believed to be hard.

3.2.4 Security Attributes and Considerations

In order for key agreement protocols to be able to withstand the attacks previously mentioned, it is desirable that these protocols possess the following security attributes [7, 6, 17].

1. *Known session key security*. Each run of the key agreement protocol should result in a unique secret session key. An adversary who learns a session key should not be able to recover data from past or future sessions.
2. *Forward secrecy*. If long-term private keys of one or more entities are compromised, the secrecy of previously established session keys should not be affected. There's a difference between *partial forward secrecy* in which one or more parties' private keys are compromised and *perfect forward secrecy* in which all participating parties' pri-

vate keys are compromised. Sometimes it's also relevant to include *TA forward secrecy* in which a compromised TA master key does not reveal past session keys. In [36], Krawczyk shows that no 2-pass AK protocol can achieve perfect forward secrecy unless the adversary is not actively involved in the choice of ephemeral keys in a session. Thus, we say that a protocol achieves *weak* perfect forward secrecy (wPFS).

3. *Key-compromise impersonation*. If A's long-term private key is compromised, the adversary can imporsonate A, but should not enable the adversary to impersonate other entities to A. A typical example of KCI can be found in the identity-based key agreement scheme of [49], as pointed out by [61], in which the symmetry of $\hat{e}(Q_A, Q_B)^s$ allows an attacker to use either S_A or S_B to carry out the impersonation.
4. *Unknown key-share*. Entity A should not be coerced into sharing a key with entity C when in fact A thinks she is sharing a key with entity B.
5. *Key control*. Neither party should be able to influence the outcome of the key more than the other. While this is an ideal attribute for key agreement schemes, it is very difficult to design a method which has *perfect key control*. This is because it's necessary for one party to choose its input key first, thus granting the other the possibility of estimating a certain number of bits by trying different input combinations.
6. *Known session-specific temporary information security*. Many protocols use some randomized private input to produce a unique session key in each run of the protocol. The compromise of this private temporary information should not compromise the secrecy of the generated session key. Although overlooked in many security analyses, exposure of such information can occur in practical implementations if ephemeral keys are precomputed or stored insecurely. It is sometimes also necessary to define *weak* known session-specific temporary information security, which assumes that a TA cannot obtain any short-term keys. Identity-based protocols can only achieve this weaker form of security.

Other desirable attributes include using a *minimal number of passes* (i.e., 2 for AK and 3 for AKC) and ensuring *low communication overhead* by limiting the amount of bits sent in each message. It is also desirable to limit the number of arithmetical operations required, thus ensuring *low computation overhead*. This is not only important to promote efficiency, but also to avoid the use of costly operations that may be targeted in a denial-of-service attack. Implementations should therefore avoid the use of costly functions in the first pass of a protocol. For this reason, it would also be preferable with the possibility of *precomputation* in order to reduce the computational overhead during on-line interaction.

3.2.5 Key Confirmation

In authenticated protocols (AK), A will merely get the assurance of that B is the only party that is able the compute the shared key. However, in many cases it is also desirable to know if B has actually computed the shared key. Similarily, B would also want the assurance of that A has computed the shared key. This can be achieved by implementing key confirmation methods.

Key confirmation ensures explicit authentication of messages. Krawczyk identified in [36] a generic attack against forward secrecy on any interactive two-party key agreement protocol that only implicitly authenticates messages. In the attack, the adversary first

masquerades as A and sends the inital message T_A to B. B then replies with T_B and completes the session with some key K. Once the session key expires at B, the adversary corrupts A, obtains the private key S_A, and finally computes the agreed session key.

In AKC protocols, key confirmation is usally provided by adding an extra pass and adding the MAC of the flow number, identities, and the ephemeral public keys. The MACs are computed under a shared key k', which is different from the session key k. If the same key was used in both cases, a passive adversary would be able to learn some information about k. The adversary could then distinguish k from a key selected uniformly at random from the key space.

[6] presents a provable secure AKC scheme (unified model) which many AK schemes [44, 42, 15, 58] has adopted. In this scheme, message authentication codes (MAC) are used in the Diffie-Hellman scheme to provide key confirmation. MACs are computationally efficient and very easy to employ in protocols.

Protocol (Unified Model with key confirmation)

Let A and B be two entities that wish to establish a shared secret key. A first selects $a \in \mathbb{Z}_q^*$ uniformly at random and sends g^a to B. On reciept, B verifies that $g^a \in [2, p-1]$ and $(g^a)^q = 1$, and then in the same way selects $b \in \mathbb{Z}_q^*$ and computes g^b. B then computes the MAC key $\kappa' = \mathcal{H}_1(g^{ab})$ and uses κ' to compute $MAC_{\kappa'}(2, B, A, g^b, g^a)$, and sends the authenticated message to A. On reciept, A checks the validity of g^b, computes κ' and verifies the authenticated message. If the MAC is ok, A sends back $MAC_{\kappa'}(3, A, B, g^a, g^b)$ to B, who then checks and verifies the message. Both parties now compute the agreed session key using a different hash function, $\kappa = \mathcal{H}_2(g^{ab})$.

$$A \xrightarrow{g^a} B$$
$$A \xleftarrow{g^b, MAC_{\kappa'}(2,B,A,g^b,g^a)} B$$
$$A \xrightarrow{MAC_{\kappa'}(3,A,B,g^a,g^b)} B$$

UNIFIED MODEL

Alternatively, k and k' may be computed under both the ephemeral and long-term keys, such that $\kappa' = \mathcal{H}_1(g^{ab} \| g^{s_a s_b})$ and $\kappa = \mathcal{H}_2(g^{ab} \| g^{s_a s_b})$. Should any of these values be compromised, the effect would then be less clear. Note, however, that compromise of long-term secret keys (such as $g^{s_a s_b}$) could reveal information about κ, and thus need to be guarded carefully.

3.3 Provable Security

Provable security was invented in the 1980's by Goldwasser and Micali [28], and originally applied to encryption schemes and signature schemes. A scheme is provable secure if there is a polynomial reduction proof from a known hard computational problem (such as those of Section 2.4) to an attack against the security of the scheme. Thus, if there is a polynomially bounded adversary that breaks the scheme, then the problem assumed to be hard can be solved in polynomial time. Provided that the assumption regarding the hardness of the problem is true, then no such adversary exists.

The process in proving security comes in four stages [3],

1. Provide a formal definition of the goals of the protocol;

2. Provide a formal adversarial model (define the capabilities of the adversary);
3. Define what it means by the protocol being secure (define attacks it should withstand);
4. Provide a security proof of the protocol by reducing a known hard computational problem to an attack on the protocol.

The security proofs of many authenticated key agreement protocols require that the hash functions used are modelled by random oracles. This approach, commonly referred to as the random oracle model, was first formulated by Bellare and Rogaway [5] and further streamlined by Blake-Wilson *et al.* [6]. In this model, ideal hash functions were introduced in which any arbitrary input generates an output selected uniformly at random. When queried with the same input more than once, the oracle is defined to respond with the output responded with previously (as a hash function would in the real world). Essentially, in the random oracle model, no adversary can make use of the underlying structure of the hash function.

Although this model allows for simple and efficient protocols to be proven secure, critics argue that no good implementation exists for a random oracle hash function. A model where no such random oracles exist is known as the standard model.

4 Related Work

This chapter provides an overview of the literature relating to this thesis. This includes an introduction to pairing-based cryptography and its use in cryptographic key agreement protocols. Furthermore, it surveys some widely discussed identity-based key agreement protocols, as well as the certificateless scheme of [1]. These protocols are featured in the security and efficiency analysis of the proposed protocol.

4.1 Pairing-Based Cryptography

Bilinear pairings of algebraic curves were originally used as cryptographic tools for reducing the discrete logarithm problem in weak elliptic curves [41, 24]. In 2000, however, Joux [32] showed that pairings also could have a positive use in cryptography by constructing a tripartite Diffie-Hellman key agreement protocol based on the Weil pairing. The message flows of the protocol are identical to the elliptic curve-based Diffie-Hellman protocol, but use the Weil pairing in computing the session key. As there is no way of efficiently computing discrete logarithms on elliptic curves, it is possible to maintain a high level of security while using short keys.

Boneh and Franklin [9] extended the idea by Joux and constructed an identity-based encryption scheme based on the properties of bilinear pairings on elliptic curves. The scheme was the first fully functional, efficient and provably secure identity-based encryption scheme, and was shortly followed by the BLS short signature scheme [10]. Since the introduction of these two important applications of pairings, many new pairing-based cryptographic protocols have been designed and analyzed (see [23] for a survey).

A lot of the work concerning pairing-based cryptography is on the realization and efficient implementation of pairings. Such implementations may, for instance, be done using the Weil pairing or the Tate pairing. Although both involve fairly complex mathematics, they can be dealt with abstractly. Most key agreement protocols use a modified pairing in order to avoid sending two points per participant. For more detailed information about pairings and their use in cryptographic protocols, see [33, 26].

4.2 Identity-Based Authenticated Key Agreement

Identity-based public key cryptography has been criticized by many for its lack of privacy in that a PKG knows every user's private key. For instance, armed with the PKG master key, it is possible to masquarade as any entity and forge its signature. In fact, this was the main motivation for the development of certificateless public key cryptography. Although many identity-based schemes have managed to defeat the inherent key escrow, they still offer very limited security against an adversary armed with the master key. On the other hand, identity-based schemes can be very efficient as no long-term public keys need to be exchanged. We will in the following sections survey a selection of escrowless identity-based authenticated key agreement protocols proposed in literature. These will be included in the security and efficiency analysis of the proposed protocol in order to properly assess the value of certificateless key agreement protocols.

In the following protocols, it is assumed that the PKG has chosen and distributed the tuple (P, P_0) in which P is a public generator and $P_0 = sP$ is the PKG public key (s is the PKG master key). The public key for a user with identity ID is given by $Q_{ID} = H(ID)$ where $H : \{0,1\}^* \to \mathbb{G}_1$ is a hash function that maps the identity to a group element. The corresponding private key is issued by the PKG, denoted by $S_{ID} = sQ_{ID}$.

4.2.1 Smart's Protocol

Smart's pairing-based scheme [58] combines the ideas of Boneh and Franklin [9] and Joux's tripartite Diffie-Hellman protocol [32]. If A and B wish to agree on a session key, they each select a private ephemeral key $a, b \in \mathbb{Z}_q^*$ and generate the corresponding public ephemeral key $aP, bP \in \mathbb{G}_1$ respectively. A then sends $T_A = aP$ to B, who sends $T_B = bP$ back to A. User A then computes $k_A = \hat{e}(aQ_B, P_0) \cdot \hat{e}(S_A, T_B)$ and user B computes $k_B = \hat{e}(bQ_A, P_0) \cdot \hat{e}(S_B, T_A)$. The scheme is consistent because $k_A = k_B = \hat{e}(aQ_B + bQ_A, P_0)$.

A		B
$a \in \mathbb{Z}_q^*$	$\xrightarrow{T_A}$	$b \in \mathbb{Z}_q^*$
$k_A = \hat{e}(aQ_B, P_0) \cdot \hat{e}(S_A, T_B)$	$\xleftarrow{T_B}$	$k_B = \hat{e}(bQ_A, P_0) \cdot \hat{e}(S_B, T_A)$
$K = k_A = k_B = \hat{e}(aQ_B + bQ_A, P_0)$		
$FK = H_1(K \Vert abP)$		

Figure 3: Smart's Protocol (modified by Chen and Kudla)

Originally, Smart's protocol did not offer perfect forward secrecy. [15] addressed this by changing the agreed session key to $FK = H_1(K \Vert abP)$ where $H_1 : \mathbb{G}_2 \times \mathbb{G}_1 \to \{0,1\}^k$. Another variant of Smart's protocol that also achieves perfect forward secrecy was proposed by Choie *et al.* [19], discussed in Section 4.2.4.

4.2.2 Chen and Kudla's Protocol

In [15], Chen and Kudla proposed an authenticated identity-based key agreement protocol. If users A and B want to establish a shared secret, they proceed as follows. A selects a random and uniformly distributed ephemeral private key $a \in \mathbb{Z}_q^*$ and sends $W_A = aQ_A \in \mathbb{G}_1$ to B. When B receives the message, he also selects a random ephemeral private key $b \in \mathbb{Z}_q^*$ and sends $W_B = bQ_B \in \mathbb{G}_1$ back to A. User A then computes the shared key $K_{AB} = \hat{e}(S_A, W_B + aQ_B)$ and user B computes $K_{BA} = \hat{e}(W_A + bQ_A, S_B)$. If both users follow the protocol correctly, they will share the same secret $K = K_{AB} = K_{BA} = \hat{e}(Q_A, Q_B)^{s(a+b)}$.

A		B
$a \in \mathbb{Z}_q^*$	$\xrightarrow{W_A, T_A}$	$b \in \mathbb{Z}_q^*$
$K_{AB} = \hat{e}(S_A, W_B + aQ_B)$	$\xleftarrow{W_B, T_B}$	$K_{BA} = \hat{e}(W_A + bQ_A, S_B)$
$K = K_{AB} = K_{BA} = \hat{e}(Q_A, Q_B)^{s(a+b)}$		
$FK = H_1(K \Vert abP)$		

Figure 4: Chen and Kudla's Protocol (without escrow)

In an escrowless variant of the scheme, A and B also exchange $T_A = aP$ and $T_B = bP$ respectively, such that the final key becomes $FK = H_1(K \Vert abP)$ where $H_1 : \mathbb{G}_2 \times \mathbb{G}_1 \to \{0,1\}^k$. Thus, the protocol achieves perfect forward secrecy and TA forward secrecy. The protocol also achieves known session-specific temporary information security, as compro-

mising both short-term keys does not lead to the recovery of the established session key. However, this is only under the assumption that no such keys are recovered by the PKG.

4.2.3 Shim's Protocol (modified by Yuan and Li)

In [57], Shim proposed an efficient identity-based key agreement protocol. The protocol was later broken by Yuan and Li [61], who demonstrated a man-in-the-middle attack on the protocol. However, Yuan and Li also proposed a modified variant of Shim's protocol which was proven secure in [14].

A		**B**
$a \in \mathbb{Z}_q^*$	$\xrightarrow{T_A}$	$b \in \mathbb{Z}_q^*$
$K_{AB} = \hat{e}(aP_0 + S_A, T_B + Q_B)$	$\xleftarrow{T_B}$	$K_{BA} = \hat{e}(bP_0 + S_B, T_A + Q_A)$
$K = K_{AB} = K_{BA} = \hat{e}(P,P)^{abs} \cdot \hat{e}(P,Q_B)^{as} \cdot \hat{e}(Q_A,P)^{bs} \cdot \hat{e}(Q_A,Q_B)^{s}$		
$FK = H_1(A, B, abP, K)$		

Figure 5: Shim's Protocol (modified by Yuan and Li)

If A and B wish to establish a shared secret key, they randomly select $a, b \in \mathbb{Z}_q^*$ and generate the corresponding public ephemeral key $aP, bP \in \mathbb{G}_1$ respectively. A then sends $T_A = aP$ to B, who sends $T_B = bP$ back to A. Entity A then computes $K_{AB} = \hat{e}(aP_0 + S_A, T_B + Q_B)$, and similarily, entity B computes $K_{BA} = \hat{e}(bP_0 + S_B, T_A + Q_A)$. If both entities have followed the protocol correctly, they share the same secret $K = K_{AB} = K_{BA} = \hat{e}(P,P)^{abs} \cdot \hat{e}(P,Q_B)^{as} \cdot \hat{e}(Q_A,P)^{bs} \cdot \hat{e}(Q_A,Q_B)^{s}$. The session key is then $FK = H_1(A, B, abP, K)$ where $H_1 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{G}_1 \times \mathbb{G}_2 \to \{0,1\}^k$.

The protocol achieves security attributes such as known session key security, perfect forward secrecy, key-compromise impersonation, and unknown key share. The protocol also achieves weak known session-specific temporary information security, as compromise of short-term keys by any (non-PKG) adversary does not reveal the established session key.

4.2.4 Choie *et al*'s Protocol

Choie et al. [19] proposed two identity-based authenticated key agreement protocols, one of which adopts a signature scheme to provide authentication. This protocol, known as Protocol I, was later shown insecure against signature forgery attacks in [56]. Protocol II, on the other hand, is a modified variant of Smart's scheme which adds perfect forward secrecy as well as KGC forward secrecy.

A		**B**
$a \in \mathbb{Z}_q^*$	$\xrightarrow{T_A}$	$b \in \mathbb{Z}_q^*$
$h = H_2(aT_B)$	$\xleftarrow{T_B}$	$h = H_2(bT_A)$
$k_A = \hat{e}(haQ_B, P_0) \cdot \hat{e}(S_A, hT_B)$		$k_B = \hat{e}(hbQ_A, P_0) \cdot \hat{e}(S_B, hT_A)$
$K = k_A = k_B = \hat{e}(S_B, T_A)^h \cdot \hat{e}(S_A, T_B)^h$		
$FK = H_1(K, Q_A, Q_B)$		

Figure 6: Choie *et al*'s Protocol II

Let A and B be two entities that engage in a protocol run to establish a common key. Both entities each randomly select their short-term key pair to be used in the session. Thus, A has an ephemeral private key $a \in \mathbb{Z}_q^*$ and a corresponding ephemeral public

key $aP \in \mathbb{G}_1$. Similarily, B has an ephemeral private/public key pair (b, bP). A initiates the protocol by sending $T_A = aP$ to B, who responds by sending $T_B = bP$ back to A. Entity A then computes $h = H_2(aT_B)$ and $k_A = \hat{e}(haQ_B, P_0) \cdot \hat{e}(S_A, hT_B)$ where H_2 is a cryptographic hash function $H_2 : \mathbb{G}_1 \to \mathbb{Z}_q^*$. In the same way, entity B computes $h = H_2(bT_A)$ and $k_B = \hat{e}(hbQ_A, P_0) \cdot \hat{e}(S_B, hT_A)$. Finally, both entities compute the session key $FK = H_1(K, Q_A, Q_B)$ where $K = k_A = k_B$ and $H_1 : \mathbb{G}_2 \times \mathbb{G}_1 \times \mathbb{G}_1 \to \{0,1\}^k$.

The protocol achieves most of the well-known security attributes, but does not achieve known session-specific temporary information security. Thus, an adversary who compromises both short-term keys of a session will be able to recover the established session key.

4.3 Certificateless Authenticated Key Agreement

Certificateless public key cryptography shares many properties with identity-based cryptography. Both assume the existance of a trusted third party which holds a master key and do not use certificates as in traditional PKI. Moreover, CL-PKC, as ID-PKC, can use pairings to devise very efficient schemes with desirable properties. However, in the certificateless setting, the KGC does not know the users' private keys. Thus, certificateless cryptography offers better security over ID-PKC in many situations.

4.3.1 Al-Riyami and Paterson's Protocol

Al-Riyami and Paterson [1] introduced in their paper on certificateless public key cryptography a simple certificateless authenticated two-party key agreement protocol. The initialization of the protocol is formally specified using the algorithms of certificateless public key cryptography [1]. These include `Setup`, `Partial-Private-Key-Extract`, `Set-Secret-Value`, `Set-Private-Key`, and `Set-Public Key`. These algorithms are explained in detail in Appendix A.1.

Assume that entities A and B wish to agree on a secret key. They first each choose the random ephemeral values $a, b \in \mathbb{Z}_q^*$ as usual, and create the corresponding ephemeral public keys $aP, bP \in \mathbb{G}_1$ respectively. A then sends the ephemeral key $T_A = aP$ and A's public key $P_A = \langle X_A, Y_A \rangle$ to B, who then in the same fashion responds with $T_B = bP$ and $P_B = \langle X_B, Y_B \rangle$ to A. Here, $X_i = x_i P$ and $Y_i = x_i P_0 = x_i sP$, where x_i is an entity's long-term secret value and s is the KGC master key.

A		**B**
$a \in \mathbb{Z}_q^*$	$T_A, \langle X_A, Y_A \rangle \longrightarrow$	$b \in \mathbb{Z}_q^*$
$\hat{e}(X_B, P_0)? = \hat{e}(Y_B, P)$	$T_B, \langle X_B, Y_B \rangle \longleftarrow$	$\hat{e}(X_A, P_0)? = \hat{e}(Y_A, P)$
$K_A = \hat{e}(Q_B, Y_B)^a \cdot \hat{e}(S_A, T_B)$		$K_B = \hat{e}(Q_A, Y_A)^b \cdot \hat{e}(S_B, T_A)$
$K = K_A = K_B = \hat{e}(S_B, T_A) \cdot \hat{e}(S_A, T_B)$		
$FK = H(K \Vert abP)$		

Figure 7: Al-Riyami and Paterson's Protocol

Once the messages are exchanged, both users verify that the same KGC master key has been used in each other's public keys. A checks if $\hat{e}(X_B, P_0) = \hat{e}(Y_B, P)$ and B checks if $\hat{e}(X_A, P_0) = \hat{e}(Y_A, P)$. See that $\hat{e}(x_i P, sP) = \hat{e}(x_i sP, P) = \hat{e}(x_i P, P)^s$. A then computes $K_A = \hat{e}(Q_B, Y_B)^a \cdot \hat{e}(S_A, T_B)$ and B computes $K_B = \hat{e}(Q_A, Y_A)^b \cdot \hat{e}(S_B, T_A)$, such that $K = K_A = K_B$ becomes the shared key between A and B. Both entities then compute the

session key $FK = H_1(K\|abP)$ where $H_1 : \mathbb{G}_2 \times \mathbb{G}_1 \rightarrow \{0,1\}^k$.

As pointed out by the authors, the protocol is vulnerable to a man-in-the-middle attack if the KGC replaces both the short-term and long-term public keys exchanged in a protocol run. The reason only a KGC can mount such an attack is because the corresponding (partial) private key must be obtained in order to compute K. However, such an attack can be mounted on all certificateless schemes, and therefore it must be assumed that the KGC is trusted not to replace public keys. Note that the protocol may be weak against denial-of-service attacks as an adversary with no attachment to the KGC, may efficiently compute valid public keys simply by knowing (P, P_0) and repeatedly query the victim to compute keys. For each key received, the victim will need to compute four pairings, and thus the computational load could be considerable if many requests are made.

Note also that if both ephemeral private keys are disclosed, the protocol breaks: $K = \hat{e}(Q_B, Y_B)^a \cdot \hat{e}(Q_A, Y_A)^b$. The key derivation function also does not resist this attack as abP is computed easily knowing both a and b. Thus, the protocol fails to achieve the property of known session-specific temporary information security.

5 Certificateless Authenticated Key Agreement

This chapter proposes a new certificateless authenticated two-party key agreement protocol and shows how it can be adapted to a multi-TA (trusted authority) setting. It also demonstrates how key confirmation can be implemented to ensure explicit authentication of messages between protocol participants. Futhermore, the chapter examines the security and the efficiency of the proposed protocol and compares it to existing certificateless and identity-based schemes.

5.1 A Certificatelss Authenticated Key Agreement Protocol

The protocol involves three entities, the communicating users A and B and the key generation center (KGC) from which the protocol participants are issued their respective partial private keys. When entity A enters the domain of the KGC, the KGC issues the partial private key $D_A = sQ_A$ where $Q_A = H_1(\mathtt{ID}_A) \in \mathbb{G}_1$, H_1 is a hash function $H : \{0,1\}^* \rightarrow \mathbb{G}_1$, $\mathtt{ID}_A$ is a unique identifier of entity A, and s is the KGC master key. A then combines the partial private key with a secret value $x_A \in \mathbb{Z}_q^*$ unknown to the KGC and generates the complete private key $S_A = \langle D_A, x_A \rangle$. A's public key is then $P_A = X_A = x_A P \in \mathbb{G}_1$ where P is a public generator of $\mathbb{G}_1$. In the following protocol, A's secret key S'_A is constructed from S_A such that $S'_A = D_A + x_A Q_A = (s + x_A)Q_A$.

If entities A and B want to jointly establish a session key, they first choose the ephemeral random values $a, b \in \mathbb{Z}_q^*$ and compute $T_A = aP$ and $T_B = bP$ respectively. Thus, each entity holds a long-term and a short-term (session-specific) key pair. A then sends $\langle T_A, P_A \rangle$ to B, who responds by sending $\langle T_B, P_B \rangle$ back to A. Both entities then validate each other's public keys by testing the group membership $P_A, P_B \in \mathbb{G}_1^*$. This step is essential to avoid the small sub-group attack observed by Lim and Lee [38]. Entities A and B then compute $K_A = \hat{e}(Q_B, P_0 + P_B)^a \cdot \hat{e}(S'_A, T_B)$ and $K_B = \hat{e}(S'_B, T_A) \cdot \hat{e}(Q_A, P_0 + P_A)^b$ respectively. Note that the scheme is consistent because $K_A = K_B = \hat{e}(Q_B, P)^{a(s+x_B)} \cdot \hat{e}(Q_A, P)^{b(s+x_A)}$.

A		**B**
$a \in \mathbb{Z}_q^*$	$\xrightarrow{T_A, P_A}$	$b \in \mathbb{Z}_q^*$
$K_A = \hat{e}(Q_B, P_0 + P_B)^a \cdot \hat{e}(S'_A, T_B)$	$\xleftarrow{T_B, P_B}$	$K_B = \hat{e}(S'_B, T_A) \cdot \hat{e}(Q_A, P_0 + P_A)^b$
$K = K_A = K_B = \hat{e}(Q_B, P)^{a(s+x_B)} \cdot \hat{e}(Q_A, P)^{b(s+x_A)}$		
$FK = H_2(K \Vert abP \Vert x_A x_B P)$		

Figure 8: Proposed AK Protocol

In order to ensure that an attacker cannot gain any information from the session key, A and B use a key derivation function on K. The shared session key is then $FK = H_2(K \Vert abP \Vert x_A x_B P)$ where $H_2 : \mathbb{G}_2 \times \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \{0,1\}^k$. Thus, the protocol achieves forward secrecy as well as known session-specific temporary information security. These properties and other security attributes are discussed in Section 5.4.6.

Observe that the protocol initialization slightly differs from the algorithms used in Al-Riyami and Paterson's protocol [1], formally specificied in `BasicCL-PKE` (see Appendix

A.1). Specifically, the public key P_A only comprise one element of $\mathbb{G}_1$ and no longer binds an entity to a specific KGC, thus allowing protocol participants under different trusted authorities to establish keys. Also see that S_A separates D_A from x_A such that these values can be used independently (in [1], $S_A = x_A D_A$). Both these changes conform to the initialization algorithms of the improved certificateless public key encryption scheme of [2] (later fixed by [62] to address an adaptive chosen chiphertext attack vulnerability), and the protocol may thus be implemented in a setting where public and private keys are constructed accordingly.

5.2 Certificateless Key Agreement Using Separate TAs

It may in some situations be desirable for users of distinct domains (under different KGCs) to exchange session keys. For example, in order for encrypted VoIP to be able to operate globally, key agreement and compatibility between networks becomes a necessary requirement. By introducing a slight modification to the above scheme we can acheive a multi-TA protocol without increasing computational nor communication overhead. In order for this to be possible, two trusted authorities, say KGC_1 and KGC_2 each have to generate a pair $(s_1P \in \mathbb{G}_1, s_1 \in \mathbb{Z}_q^*)$ and $(s_2P \in \mathbb{G}_1, s_2 \in \mathbb{Z}_q^*)$ where P and $\mathbb{G}_1$ are globally agreed.

Suppose that A's private key is generated by KGC_1 with master key s_1 and that B's private key is generated by KGC_2 with master key s_2. If A and B then wish to establish a common secret key, they must run the protocol using each other's KGC master public key. Thus, A uses s_2P in computing the shared key, while B, on the other hand, uses s_1P. After both parties have chosen the ephemeral keys $a, b \in \mathbb{Z}_q^*$ respectively and exchanged the corresponding public keys, entity A computes the key $K_A = \hat{e}(Q_B, s_2P + P_B)^a \cdot \hat{e}((s_1 + x_A)Q_A, T_B)$, while entity B computes the key $K_B = \hat{e}((s_2 + x_B)Q_B, T_A) \cdot \hat{e}(Q_A, s_1P + P_A)^b$.

A		**B**
$S_A = \langle s_1 Q_A, x_A \rangle$		$S_B = \langle s_2 Q_B, x_B \rangle$
$a \in \mathbb{Z}_q^*$	$\xrightarrow{T_A, P_A}$	$b \in \mathbb{Z}_q^*$
$K_A = \hat{e}(Q_B, s_2P + P_B)^a \cdot$	$\xleftarrow{T_B, P_B}$	$K_B = \hat{e}((s_2 + x_B)Q_B, T_A) \cdot$
$\hat{e}((s_1 + x_A)Q_A, T_B)$		$\hat{e}(Q_A, s_1P + P_A)^b$
$K = K_A = K_B = \hat{e}(Q_B, P)^{a(s_2 + x_B)} \cdot \hat{e}(Q_A, P)^{b(s_1 + x_A)}$		
$FK = H_2(K \Vert abP \Vert x_A x_B P)$		

Figure 9: Proposed AK Protocol Using Separate TAs

The scheme is consistent as $K_A = K_B = \hat{e}(Q_B, P)^{a(s_2 + x_B)} \cdot \hat{e}(Q_A, P)^{b(s_1 + x_A)}$. To achieve the desired security attributes, a KDF is used to generate the session key such that $FK = H_2(K \Vert abP \Vert x_A x_B P)$. Also note that a foreign master public key cannot be obtained unless a user registers under the desired KGC, or it is transmitted by someone who is in possession of it. As the former defeats the purpose of the protocol, such keys must be exchanged by the participants in an authenticated manner.

5.3 Certificateless Key Agreement Using Key Confirmation

Although authenticated key agreement (AK) provides the assurance that nobody except the intended party can compute the session key, it may in sometimes be desirable to have some kind of confirmation as to whether the key has been successfully created or

not. An AKC protocol can be derived from an AK protocol by adding the MACs of the flow number, identities, and the ephemeral keys. This section describes an AKC variant of the proposed AK protocol and uses a method identical to that of Blake-Wilson *et al* in [6] which has been proven secure in the random oracle model. In turn, other schemes [15, 58] have also adopted this method in providing explicit authentication of messages.

In the following protocol, MACs are used for providing key confirmation and are computed under the key $FK' = H_3(K\|abP\|x_Ax_BP)$ where H_3 is a key derivation function independent[1] from H_2.

A		**B**
$a \in \mathbb{Z}_q^*$	$\xrightarrow{T_A, P_A}$	$b \in \mathbb{Z}_q^*$
	$\xleftarrow{T_B, P_B, MAC_{FK'}(2, ID_B, ID_A, T_B, T_A, P_B, P_A)}$	
	$\xrightarrow{MAC_{FK'}(3, ID_A, ID_B, T_A, T_B, P_A, P_B)}$	
$K = K_A = K_B = \hat{e}(Q_B, P)^{a(s+x_B)} \cdot \hat{e}(Q_A, P)^{b(s+x_A)}$		
$FK = H_2(K\|abP\|x_Ax_BP)$		

Figure 10: Proposed AK Protocol with Key Confirmation

A starts the protocol in the usual way by sending $T_A, P_A \in \mathbb{G}_1$ to B. After having received the message, B returns T_B, P_B and a MAC computed under the key FK' containing the flow number, identities, ephemeral keys, and the public keys. Both long-term and short-term public keys must be included as they can be replaced by an adversary. The MAC is computed under a key known only to A and B in order to maintain key indistinguishability. A then computes the MAC value as B would and checks that it is identical to the one provided in the message. If the MAC values match, A confirms to B by creating a new MAC, also computed under FK'.

Although key confirmation introduces one additional flow, the MACs can be computed efficiently and thus the AKC protocols have essentially the same computational overhead as the AK protocols. Assuming that the ephemeral keys are different in each run, one may heuristically argue that the desired key confirmation is obtained.

5.4 Security Analysis

A security analysis is intended to provide some assurance about the security properties of a protocol and to facilitate its design in a way that subtle flaws may be avoided. In evaluating the security of our key agreement protocol, we show that (in a best case scenario) an adversary cannot recover a session key unless the bilinear Diffie-Hellman problem is solved. We also show in the random oracle model that the lack of a KDF may allow an adaptive attacker to distinguish a true session key from a random bit string.

5.4.1 Defining the Adversary

As the public keys are not derived directly from a user identity as in ID-PKC, and not authenticated in the form of a certificate as in traditional PKI, it has to be assumed that an adversary can replace these. However, an adversary will not gain anything useful unless the corresponding private key for the replaced public key is obtained, which requires

[1] When a hash function is modeled by a random oracle, $H_2(\cdot) = H(2, \cdot)$ and $H_3(\cdot) = H(3, \cdot)$ are independent random oracles.

cooperation with the KGC. An adversary with this capability may mount a man-in-the-middle attack on the key exchange in an undetectable way. For instance, if entities A and B engage in a key agreement, the adversary may replace $\langle aP, x_A P\rangle$ sent by A with $\langle a'P, x'_A P\rangle$, and similarily substitute $\langle bP, x_B P\rangle$ sent by B with $\langle b'P, x'_B P\rangle$. Thus, the adversary will have established a key $K_{B'A} = \hat{e}(Q_A, sP + x_A P)^{b'} \cdot \hat{e}((s + x'_B)Q_B, aP)$ with A and a key $K_{A'B} = \hat{e}(Q_B, sP + x_B P)^{a'} \cdot \hat{e}((s + x'_A)Q_A, bP)$ with B. By obtaining the corresponding (partial) private key for both replaced public keys, the session keys can be computed efficiently.

The man-in-the-middle attack can only be mounted by the KGC or an adversary who has obtained the KGC master key. This observation can further help us define the adversaries for certificateless authenticated key agreement (CL-AKA) and aid in constructing a proper security model. Essentially, an adversary should not replace public keys while at the same time holding the KGC master key. Similarily, an adversary who has replaced a public key should not be able to obtain the partial private key for the replaced key. Consequently, the KGC must be trusted not to replace entities' public keys.

If keys are generated using the alternative key generation technique suggested in [1], then a KGC will leave evidence if it tries to replace public keys. In this technique, the public key of A is bound to the identifier, such that $Q_A = H_1(ID_A \| P_A)$. The partial private key issued to A by the KGC is still of the form $D_A = sQ_A$, but is also bound to A's public key. Thus, A can only hold one single public key and one private key. Consequently, if a cheating KGC tries to replace A's public key, there will be two working public keys for A, for which each has a corresponding partial private key. Only the KGC is able to create partial private keys, and thus the public key replacement is apparent.

In the next section, we will present a security and adversary model for CL-AKA that formally captures the findings presented here.

5.4.2 CL-AKA Security Model

Many key agreement protocols provide proof that adopt the extended formulation by Blake-Wilson [6] of the Bellare-Rogaway model [5]. Such a model may be able to test the security strength of a protocol, but may in some situations be insufficient depending on the power of the adversary. An adaptive adversary may for instance be able to obtain the shared secret between two entities if the session key is not constructed properly (for instance, see Section 5.4.3).

It is also desirable to evaluate the security of a protocol in the event that ephemeral keys can be compromised. Most models completely disregard this possibility and thus fail to achieve known session-specific temporary information security [17]. Such security can be particularily troublesome to achieve in an identity-based setting where the PKG already knows all entities' private keys. If any of the session-specific keys are compromised, the session key can easily be recovered by the PKG. Moreover, identity-based key agreement schemes that are intended to be escrowless completely relies on the session-specific information being kept secret. It would be unreasonable to think that an adversary has no means of obtaining such values if they are precomputed and possibly stored insecurely. In the certificateless setting, on the other hand, each entity has two secrets. Compromising any one of these secrets should not affect the security of the protocol. In order to properly capture compromise of session-specific temporary information and to formulate a security model for CL-AKA, we adopt the asymmetric key agreement model of [17].

In the BR model [5], each party involved in a session (run) of a protocol is treated as an oracle. An adversary can access the oracle by issuing the allowed queries. An oracle $\Pi^s_{i,j}$ denotes an instance s of a party i that wants to establish a key with a party j. The instance of j is $\Pi^t_{j,i}$ for some t. Given an input message, the oracle $\Pi^s_{i,j}$ runs the protocol Π and generates the output by $\Pi(1^k, i, j, SP_i, P_j, conv^s_{i,j}, r^s_{i,j}, x) = (m, \delta^s_{i,j}, \sigma^s_{i,j})$ where x is the input message; m is the output message; 1^k is the security paramter; SP_i is the private and public key pair of party i; P_j is the public key of j; $r^s_{i,j}$ is the random coin flips (ephemeral key) of sender i; $\delta^s_{i,j}$ is the decision of the oracle, and $\sigma^s_{i,j}$ is the generated session key. Upon completion, Π updates the conversation transcript $conv^s_{i,j}$ as $conv^s_{i,j}.x.m$. Here, $x.m$ denotes the concatenation of two strings, x and m.

A general adversary can access an oracle by issuing any of the following queries:

- $Send(\Pi^s_{i,j}, x)$. $\Pi^s_{i,j}$ executes $\Pi(1^k, i, j, SP_i, P_j, conv^s_{i,j}, r^s_{i,j}, x)$ and responds with m and $\delta^s_{i,j}$. If the oracle $\Pi^s_{i,j}$ does not exist, it will be created. The Send query allows an adversary to send a message to any oracle $\Pi^s_{i,j}$, such that i believes the message has been sent from j. The adversary may initiate protocol runs using such queries.
- $Reveal(\Pi^s_{i,j})$. $\Pi^s_{i,j}$ reveals the private output $\sigma^s_{i,j}$ of the session if the oracle accepts. The Reveal query allows an adversary to ask an oracle $\Pi^s_{i,j}$ to reveal the session key it currently holds.
- $Corrupt(i, K)$. The party i responds with the private key S_i and updates SP_i (if $K \neq \lambda$). The Corrupt query allows an adversary to ask a party to reveal its long-term private key or to replace the key pair with any key of the adversary's choice.
- $Partial(i)$. The party i responds with the partial private key D_i. This is a query introduced by us such that an adversary may request partial private keys for any entity. Note that the partial private key is fixed to the identity, and thus cannot be replaced by an adversary. An entity issued the Partial query is not corrupted. Partial queries essentially simulate an eavesdropping KGC's advantage against an outside attacker.
- $Replace(\Pi^s_{i,j}, P)$. The oracle $\Pi^s_{i,j}$ updates $P_j = P$ when $i \neq j$. The Replace query allows an adversary to force i to replace the public key for j. As entities in certificateless schemes must exchange public keys, it is natural to assume that an adversary can replace these. Allowing this query addresses the source substitution attack.
- $Coin(\Pi^s_{i,j}, r)$. The oracle $\Pi^s_{i,j}$ replies with the random coin flips $r^s_{i,j}$ of the sender i in a session s with a partner j. If r is defined, the oracle will use r as the random flips $r^s_{i,j}$. The Coin query was introduced in [17] and allows an adversary to control any input of a protocol algorithm such as the session-specific temporary information. Also note that Coin queries can be issued to oracles without corrupting a party. Moreover, Coin queries can indicate whether disclosing the ephemeral private keys reveals the session key or not.
- $Test(\Pi^s_{i,j})$. Allows an adversary to query an oracle $\Pi^s_{i,j}$ to output $\sigma^s_{i,j}$, which is either a true session key or a randomly generated key. The adversary then must guess if the key is real or not.

An oracle can exist in a number of different states. In the *accepted* state, the oracle has generated a session key after having received properly formulated messages. In the *rejected* state, the oracle has not established a session key, and thus rejects holding one.

An oracle that is in the *-state, has not decided whether to accept or reject. An *opened* oracle has responded to a reveal query and has thus revealed its session key. An oracle that has been *corrupted*, has responded to a corrupt query and revealed or replaced its private key. An oracle $\Pi^s_{i,j}$ is also corrupted if its public key is replaced in $\Pi^t_{j,i}$ (using the Replace query). An oracle $\Pi^s_{i,j}$ is *controlled* if it has responded to a Coin query.

Each party in a protocol has its own session transcript[2]. If two oracles $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$ have, via the adversary, received properly formatted messages and established a shared session secret, then these oracles have had a *matching conversation* (see [5] for a formal definition). The relation between the oracles' transcripts define whether a matching conversation has taken place.

An adversary's main goal in the standard definition for security for a key agreement protocol is to be able to distinguish a session key established by two arbitrary entities from a string of random bits. In the first phase, the adversary E can issue any number of queries to a set of oracles. When it has decided it has collected enough information, E ends the first phase. In the second phase, E issues a Test query to a *fresh oracle* $\Pi^s_{i,j}$, defined as follows.

Definition 5.4.1 (fresh oracle)**.** *An oracle $\Pi^s_{i,j}$ is fresh if (1) $\Pi^s_{i,j}$ has accepted (it knows the partner j); (2) $\Pi^s_{i,j}$ is unopened (has not been issued the Reveal query); (3) party i is not* both *controlled and corrupted; (4) party $j \neq i$ is not corrupted; (5) there is an unopened oracle $\Pi^t_{j,i}$ which has had a matching conversation to $\Pi^s_{i,j}$.*

Note that this definition allows the party i to be corrupted, and thus can be used to address the key-compromise impersonation property (discussed in Section 5.4.6). A party should not both be corrupted and controlled as it would enable an adversary to recover the session key. It is possible to further extend the definition to address forward secrecy by allowing both entities to be corrupted, as long as both oracles remain unopened and uncontrolled. The definition can also be used to address known session-specific temporary information security in which both entities are controlled only (have been issued the Coin query).

After E has issued the Test query, oracle $\Pi^s_{i,j}$, as a challenger, randomly chooses $b \in \{0,1\}$ and responds with the session key $\sigma^s_{i,j}$ if $b = 0$. Otherwise, it returns a random sample generated according to the distribution of the session secret $\sigma^s_{i,j}$. The adversary must guess the value of b by issuing a prediction bit b', and thus the advantage is defined to be

$$Advantage^E(k) = \max\{0, Pr[b' = b] - \frac{1}{2}\}$$

Essentially, an entity involved in a session has two secrets, a short-term (session-specific temporary key) and a long-term (universal private key). Compromising any one of these secrets should not enable the adversary to obtain the secret session key. Thus, we allow an adversary to obtain any pair of secrets from any session as long as it does not have knowledge of more than one secret from each entity involved. Note, however, that allowing an adversary to be actively involved in the choice of session-specific temporary information makes perfect forward secrecy (PFS) impossible. For instance, an adversary

[2]A session transcript is used to uniquely identify a session by the involved parties, and can be the concatenation of messages exchanged, otherwise known as a session ID.

can obtain a party's short-term secret for a given session, and in some other subsequent session, corrupt the party and obtain its private key. Thus, the adversary can compute all the session keys for which it previously has learned the session-specific information. Similarily, if an adversary has previously obtained the private key for some entity, it can compute the session keys of subsequent sessions for which it learns the session-specific information. For this reason, the model uses a weaker form of PFS [36], which only guarantees perfect forward secrecy in the face of adversaries that is not actively involved in the choice of session-specific temporary information.

We will now present two distinct types of adversaries for CL-AKA. These are also similar to those used in the CL-PKE scheme of [1]. A Type-I adversary may replace public keys at will, but does not know the KGC master key. In order to model security against an eavesdropping KGC, we also want to consider a Type-II adversary which knows the KGC master key, but does not replace public keys.

CL-AKA Type-I Adversary An adversary $\mathcal{A}_{\mathrm{I}}$ does not have access to the KGC master key. However, $\mathcal{A}_{\mathrm{I}}$ may replace public keys, request (partial) private and session-specific temporary keys, and issue reveal queries for all entities of its choice. A Type-I adversary has the following restrictions:

1. $\mathcal{A}_{\mathrm{I}}$ cannot request the private key for any identity if the corresponding public key has already been replaced.
2. $\mathcal{A}_{\mathrm{I}}$ cannot both replace the public key for an entity involved in a Test session *and* request the partial private key for that entity in some other phase.
3. In phase 2, $\mathcal{A}_{\mathrm{I}}$ cannot reveal the key of a session currently being tested.

The first restriction comes naturally as it would be unreasonable to assume that the challenger is able to respond to a Corrupt query if the public key has been replaced. However, an adversary can still query for the partial private key and mount a standard man-in-the-middle attack. Thus, we introduce the second restriction. Obviously, no adversary should be able to issue a Reveal query on the test session as it would allow it to distinguish the session key from a randomly generated one.

CL-AKA Type-II Adversary An adversary $\mathcal{A}_{\mathrm{II}}$ has access to the KGC master key, but cannot replace public keys of entities. An adversary $\mathcal{A}_{\mathrm{II}}$ can compute partial private keys itself (using the master key), query for private and session-specific temporary keys, and issue reveal queries. A Type-II adversary has the following restrictions:

1. $\mathcal{A}_{\mathrm{II}}$ cannot replace any public keys.
2. In phase 2, $\mathcal{A}_{\mathrm{II}}$ cannot reveal the key of a session currently being tested.

An adversary can be *adaptive* if it's allowed to continue performing queries after having issued the Test query, but before guessing the value of b. However, the oracle $\Pi^{s}_{i,j}$ being tested should still remain fresh (see definition). Adaptive adversaries are considered more powerful than non-adaptive adversaries as they may use the values obtained in the test session to derive the session key (or some other sensitive information) using a different session. Section 5.4.3 shows that such an attack easily can be mounted on any protocol (including ours), if the session key is not constructed properly.

Sometimes it is also useful to define a friendly adversary that does not tamper with the messages between oracles, but only passively monitors the protocol.

Definition 5.4.2 (benign adversary)**.** *An adversary is called a* benign adversary *if it faithfully conveys messages between two oracles* $\Pi^s_{i,j}$ *and* $\Pi^t_{j,i}$.

In the face of a benign adversary, a secure authenticated key agreement protocol [6] is defined as follows:

Definition 5.4.3. *A protocol* Π *is a secure AK if:*

1. *In the presence of the benign adversary on* $\Pi^s_{i,j}$ *and* $\Pi^t_{j,i}$*, both oracles always accept holding the same session key* σ*, and this key is distributed uniformly at random on* $\{0,1\}^k$*; and for every adversary* E*:*
2. *If two oracles* $\Pi^s_{i,j}$ *and* $\Pi^t_{j,i}$ *have matching conversations and both* i *and* j *are uncorrupted, then both accept and hold the same session key* σ*;*
3. $\mathrm{Advantage}^E(k)$ *is negligible.*

Similarily, a secure authenticated key agreement protocol with key confirmation is defined as follows:

Definition 5.4.4. *A protocol* Π *is a secure AKC if:*

1. *In the presence of the benign adversary on* $\Pi^s_{i,j}$ *and* $\Pi^t_{j,i}$*, both oracles always accept holding the same session key* σ*, and this key is distributed uniformly at random on* $\{0,1\}^k$*; and for every adversary* E*:*
2. *If two oracles* $\Pi^s_{i,j}$ *and* $\Pi^t_{j,i}$ *have matching conversations and both* i *and* j *are uncorrupted, then both accept and hold the same session key* σ*;*
3. *The probability of no-matching*[3] *event is negligible;*
4. $\mathrm{Advantage}^E(k)$ *is negligible.*

5.4.3 Session Key Reveal Attack

In the proposed protocol, the key derivation function modelled as a random oracle hash function not only ensures forward secrecy, but also avoids a number of attacks which otherwise may exploit the properties of underlying mathematical structures (such as the bilinear pairing). We will now show that given a powerful adaptive adversary in the security model of our scheme, the session key between any two parties can be recovered in the absence of a KDF.

The adversary E begins by issuing a number of queries to a set of oracles. Essentially, these queries should not make E's probability in distinguishing a true session key from a randomly generated one non-negligible. When E decides it has collected enough information, it issues a Test query on a fresh oracle $\Pi^s_{i,j}$. The oracle $\Pi^s_{i,j}$ answers with $\sigma^s_{i,j}$ which is either the session key computed from T_A and T_B or a random one. The outcome of $\sigma^s_{i,j}$ depends on a value $b \in \{0,1\}$ selected at random by $\Pi^s_{i,j}$, and thus the adversary must guess the value of b.

Unlike a general adversary, the adaptive adversary is allowed to continue making queries after having issued the Test query for some oracle $\Pi^s_{i,j}$. However, the oracle

[3] We use no-matching to describe an oracle who has reached the accepted state (generated a session key), but there's no party who've engaged in a matching conversation.

currently being tested must still remain fresh. Thus, before making its guess, E establishes a different session using the ephemeral public keys uT_A and uT_B where $u \in \mathbb{Z}_q^*$. The adversary then issues a Reveal query to the oracle which responds with the session key K'. The adversary can then recover the session key for $\Pi^s_{i,j}$ by computing $K = K'^{u^{-1}}$. E will then be able to guess b correctly (checks if $\sigma^s_{i,j} = K$) and wins the game. A similar attack can also be mounted on Al-Riyami and Paterson's [1] protocol .

5.4.4 Reduction to Forging Attack

If a KDF such as a hash function is used in computing the final session key, then the above attack is not possible since the Reveal query will return a hash value[4]. The value $H(\sigma)$ is unique for every *signature* σ (in our case denoted by $K\|abP\|x_Ax_BP$ for entities A and B) input to the hash function H, and thus the attacker is left with two options in distinguishing the output key from a random string. One is to find a signature which hashes to the same value as the key of the Test session. The other is to compute the signature using the known public values and parameters. Respectively, these are known as a *key-replication attack* and a *forging attack*.

We will now show that the security reduces to a forging attack by demonstrating that a key-replication attack is impossible, given that the hash function is a random oracle. In the key-replication attack, an adversary must obtain a signature σ from an arbitrary session that matches the signature σ_T of a Test session. Assume that a session between two entities A and B, who exchange the keys aP, bP respectively, is chosen as a Test session (where $K = \hat{e}(Q_A, P)^{b(s+x_A)} \cdot \hat{e}(Q_B, P)^{a(s+x_B)}$). Assume further that the same session-specific temporary information is used in a session between two entities C and D. Due to the bilinearity of K, it is possible to exchange Q_A, Q_B with Q_C, Q_D if the corresponding long-term private keys are changed accordingly. Specifically, K of the two sessions will remain the same if $\alpha(s + x_A) = \gamma(s + x_C)$ where $\alpha \neq \gamma$ for $H(ID_A) = \alpha P$ and $H(ID_C) = \gamma P$, and if $\beta(s + x_B) = \delta(s + x_D)$ where $\beta \neq \delta$ for $H(ID_B) = \beta P$ and $H(ID_D) = \delta P$. However, in obtaining any signature that matches σ_T, it is also a requirement that $x_Ax_BP = x_Cx_DP$ holds. Thus, the participants of the two sessions must be the same (and consequently also the sessions themselves must be identical), and therefore an attack of this type cannot be mounted.

We would also like to point out that if x_Ax_BP for a key agreement between entities A and B had not been included in the KDF, an adversary would still have no means of checking that K of σ_T matches K of some other signature σ, unless the DLP problem is solved (such that for instance α in $H(ID_A) = \alpha P$ is known). Naturally, if an adversary solves the DLP, then the above scheme is easily broken.

5.4.5 Session Key Forgery

In this section we will heuristically argue that in order for any adversary $\mathcal{A}$ to compute a signature σ that gives the same hashed key as the Test session, the bilinear Diffie-Hellman (BDH) problem or the computational Diffie-Hellman (CDH) problem must be solved. Which problem that needs to be solved depends on the information available to the adversary. If the adversary may obtain any one of two secrets from each participating entity in a session, then the BDH problem must be solved *at best*. An adversary who is armed with nothing except the public information exchanged between the protocol participants, on the other hand, must solve the CDH problem.

[4]We assume that the Reveal query only returns the final session key FK.

Protocol	KnSK	FwS	KCI	UKS	KContl	KnSSTI
Smart[58, 15]	✓	✓	✓	✓	✓	x
Chen-Kudla[15] #2'	✓	✓	✓	✓	✓	weak
Shim[57, 61]	✓	✓	✓	✓	✓	weak
Choie-Yeong-Lee[19] #2	✓	✓	✓	✓	✓	x
Al-Riyami-Paterson[1]	✓	✓	✓	✓	✓	x
Proposed protocol	✓	✓	✓	✓	✓	✓

Table 1: Security attributes comparison

Reduction to the BDH/CDH problem

The shared secret between entities A and B is given by $FK = H(K\|abP\|x_A x_B P)$. Assume that $\mathcal{A}$ has managed to obtain the short-term private key a (from A) and the long-term private key S_B (from B). It is easy to see that the adversary is able to compute $a \cdot T_B = abP$ and $x_B P_A = x_A x_B P$. However, K cannot be computed as the attacker only knows the private keys for one pairing. Thus, the adversary requires access to a BDH oracle in order to compute both pairings of K. Specifically, $\mathcal{A}$ must query the oracle with the instance $(Q_A, bP, (s + x_A)P)$ to obtain the solution $\hat{e}(P, P)^{\alpha b(s+x_A)}$ where $Q_A = \alpha P$. The session key FK between A and B can then be computed efficiently.

Note that "as best" is used to point out that if any other combination of keys is obtained (as long as only one is retrieved from each party), then the presumably harder CDH problem must be solved. For instance, if two ephemeral keys are compromised in a session, then the adversary can compute K and abP, but must solve the CDH problem (from $x_A P$ and $x_B P$) in order to retrieve $x_A x_B P$. Similarily, if both long-term private keys are compromised, then the CDH problem (from aP and bP) must be solved in order to obtain abP. Naturally, if no such keys are compromised, then the adversary also must solve the CDH problem.

5.4.6 Security Attributes

In this section we will heuristically argue that the protocol satisfies the following security properties. Table 1 summarizes the security attributes of a selection of identity-based and certificateless authenticated key agreement protocols.

1. **Known session key security (KnSK)**: As ephemeral values are used in generating session keys, a compromised session key does not compromise past or future sessions. All protocol runs, even when its participants remains the same, produce a different session key. An adversary's inability to perform the key-replication attack as demonstrated in Section 5.4.3 also shows that the protocol provides known-key security.
2. **Forward secrecy (FwS)**: We let this property constitute two separate parts; both to capture the forward secrecy against an outside adversary and against an adversary who posesses the KGC master key (or a cheating KGC).
    1. *Weak perfect forward secrecy*: If the long-term private keys of entities A and B are compromised, previously established session keys will still remain unknown to an adversary due to the key derivation function $H(K\|abP\|x_A x_B P)$. As the adversary does not know any of the ephermeral values used in a session, she must compute abP from aP and bP which is the CDH problem. Here, we assume that the adversary has not been actively involved in the choice of temporary information of

past sessions, and thus obtain the property of *weak* perfect forward secrecy [36]. If a KDF had not been used, an attacker could easily recover the shared key by computing $K = \hat{e}(S'_B, T_A) \cdot \hat{e}(S'_A, T_B)$.

2. *KGC forward secrecy*: Compromise of the KGC master key s does not enable one to reveal previously established session keys. Actually, a KGC does not gain any advantage over an outside attacker in obtaining the session key between two arbitrary entities. This is one of the key advantages of certificateless schemes over identity-based cryptography. In the session key $K = \hat{e}(Q_B, P)^{a(s+x_B)} \cdot \hat{e}(Q_A, P)^{b(s+x_A)}$, an outside attacker does not know s nor the pair $\langle x_A, x_B \rangle$. Therefore, solving $(s + x_A)$ can be reduced to solving a single variable x_A. Thus, an outside attacker must solve $\hat{e}(Q_B, P)^{ax_B} \cdot (Q_A, P)^{bx_A}$ in order to obtain the session key. An attacker with access to the KGC master key s, on the other hand, may solve parts of the session key K. To demonstrate, the session key may alternatively be written as $K = \hat{e}(Q_B, P)^{as} \cdot \hat{e}(Q_B, P)^{ax_A} \cdot \hat{e}(Q_A, P)^{bs} \cdot \hat{e}(Q_A, P)^{bx_B}$. We see that this adversary can easily solve $\hat{e}(Q_B, P)^{as}$ and $\hat{e}(Q_A, P)^{bs}$. Thus, the adversary must solve $\hat{e}(Q_B, P)^{ax_A} \cdot \hat{e}(Q_A, P)^{bx_B}$, and we see that problem is essentially the same as that of an outside attacker.

3. **Key-compromise impersonation (KCI)**: The proposed AK protocol is resistant to key-compromise impersonation because the key is computed using asymmetric information. Assume that the adversary E knows A's private key $S_A = \langle D_A, x_A \rangle$. If E then intercepts T_B from B during a protocol run and attempts to pass on $T_{B'}$ to A, then E will have to compute $K_{B'A} = \hat{e}(Q_B, x_B P + sP)^a \cdot \hat{e}(D_A + x_A Q_A, T_{B'})$. E cannot compute $K_{B'A}$ because she does not know the ephemeral value a. If E was to compute $K_{AB'}$, she would have to know S_B. Note also that the fresh oracle definition addresses the KCI property as it allows a party participating in a session to be issued the Corrupt query.

4. **Unknown key share (UKS)**: An entity A cannot be coerced into sharing a key with C when in fact A thinks she is sharing a key with B. If A wants to share a key with B, A uses $B's$ public key P_B and identifier Q_B in computing the session key. Thus, C must obtain the corresponding private key in order to compute the key. Note that incorporating parties' identities in the computation of a session key generally avoids the unknown key share (UKS) attack (for instance, see [37]).

5. **Key control (KContl)**: Neither party can control the outcome of the session key. Note, however, that if A sends her ephemeral key first, B may be able to predict some bits of the final key by trying different ephemeral keys before sending the key back to A. Precisely, in computing the shared session key $f(a, b)$ where a is known, B may compute 2^s variants of b and thus select approximately s bits of the joint key. This deficiency exists in all interactive key distribution protocols as pointed out by [43]. An implementation of the protocol can address this to some extent by limiting the time available to the responder in sending his or her value back to the initiator.

6. **Known session-specific temporary information security (KnSSTI)**: Compromising the ephemeral private keys of a session does not enable an attacker to compute the session key. Specifically, obtaining the keys a and b in any session between entities A and B, allows the adversary to compute $K = \hat{e}(Q_B, P_B + P_0)^a \cdot \hat{e}(Q_A, P_A + P_0)^b$

and $a \cdot T_B = b \cdot T_A = abP$. However, in order to compute $x_A x_B P$, the adversary must also know at least one private long-term key. Thus, the protocol achieves KnSSTI even in the presence of a cheating KGC who has obtained the short-term keys of a session. Note that the compromise of session-specific keys is not an unlikely scenario as these values may be precomputed and possibly stored insecurely. Many schemes fail to obtain this security attribute (for instance [1, 58, 16, 37]), and thus break in the face of an adversary who can obtain such information. Compromise of session-specific temporary information is discussed further in [17].

5.4.7 Other Security Considerations

Even if the security properties of the previous section are obtained, additional considerations must be made to ensure secure implementations. For instance, in order to withstand denial of service attacks, it's important that the first flow in the protocol does not put the responder at a serious disadvantage if many key agreement requests are made. Since public keys are public by definition, an adversary may easily be able to issue bogus requests, and thus force the victim to compute one session key per request. Although the effect is not as dramatic as in [1] where four pairings are computed, it still can become a computational burden to the responder.

One method to address the problem is to force the connection initiators to compute solutions to cryptographic puzzles [34]. The responder may for instance request that the initiator computes the value of the hash $H(x, y)$ in which y and the resulting hash value are known. The initator will then need to perform an exhaustive search on the remaining bits in order to find the value of x. Upon receiving the value x, the responder verifies its corectness and thus proceeds to carry out the key agreement.

5.4.8 Converting to Identity-Based Cryptography

This section will demonstrate that the proposed protocol cannot achieve all the security attributes of Section 5.4.6 if brought to an identity-based setting.

In identity-based cryptography, the public key is directly derived from the identity, such that an entity A's public key becomes $H_1(\mathrm{ID}_A) = Q_A$. We let A's partial private key be its full private key such that $S_A = D_A = sQ_A$. If A wants to establish a session key with B, they exchange ephemeral public keys as usual, but do not exchange public keys as these already are known in the identity-based setting. A then computes $K_{AB} = \hat{e}(Q_B, sP)^a \cdot \hat{e}(sQ_A, bP)$. Similarily, B computes $K_{BA} = \hat{e}(Q_A, sP)^b \cdot \hat{e}(sQ_B, aP)$. Using a KDF, the final session key then becomes $FK = H(K \| abP)$ where H is a suitable hash function. Notice that this protocol is identical to Smart's protocol [58, 15].

This protocol does not acheive known session-specific temporary information security as shown in Table 1. Thus, if an adversary obtains the short-term private keys, the session key can be efficiently computed. Moreover, in our security model, an adversary armed with the KGC master key can no longer query for session-specific private information as it will reveal the established session key.

5.5 Efficiency Analysis

To ensure fast response times and low power consumption, a key agreement protocol should have low communication and computational overhead. Communication overhead refers to the number of bits transmitted by each entity in a protocol run, while computational overhead refers to the cost of all arithmetic computations each entity must perform

Protocol	message	session key
Smart[58, 15]	T_A	$H(\hat{e}(S_A, T_B) \cdot \hat{e}(S_B, T_A) \| abP)$
Chen-Kudla[15] #2'	W_A, T_A	$H(\hat{e}(Q_A, Q_B)^{s(a+b)} \| abP)$
Shim[57, 61]	T_A	$H(A \| B \| abP \| \hat{e}(T_A + Q_A, T_B + Q_B)^s)$
Choie-Yeong-Lee[19] #2	T_A	$H(\hat{e}(S_A, T_B)^h \cdot \hat{e}(S_B, T_A)^h, Q_A, Q_B)$
Al-Riyami-Paterson[1]	$T_A, \langle X_A, Y_A \rangle$	$H(\hat{e}(S_A, T_B) \cdot \hat{e}(S_B, T_A) \| abP)$
Proposed protocol	T_A, X_A	$H(\hat{e}(S'_A, T_B) \cdot \hat{e}(S'_B, T_A) \| abP \| x_A x_B P)$

Table 2: Message and session key comparison

in order to carry out the key agreement. In the following sections, we evaluate the efficiency by looking at the communication and computational complexity of the proposed protocol, and compare it to existing certificateless and identity-based schemes.

5.5.1 Communication and Storage Complexity

The participants of the proposed protocol exchange short-term (session-specific) and long-term public keys. Each key is modelled by an integer element of an elliptic curve group, and are much smaller in size than keys used in traditional public key cryptosystems (i.e. RSA). For instance, in providing the security equivalent of a 1024-bit RSA key, only a 160-bit elliptic curve key is needed.

The long-term public keys only comprise one group element ($P_A = X_A = x_A P$ for entity A), whereas each public key in Al-Riyami and Paterson's [1] (AP) certificateless key agreement protocol consist of two group elements ($P_A = \langle X_A, Y_A \rangle = \langle x_A P, x_A sP \rangle$ for entity A). Not only does this reduce the overall bandwidth, but it also reduces the space required by each user in storing public keys by one half[5]. In these respects, the proposed protocol can be considered more efficient than AP's protocol. Moreover, if many keys are agreed or the protocol is used in a client/server setting (in which the server public key is known), only fresh short-term keys need to be exchanged in each protocol run. Thus, the protocol can be considered just as efficient as identity-based schemes where public keys are always known.

Table 2 provides an overview of the exchanged messages and the established session key in certificateless and identity-based protocols. Ideally, we want to compare the certificateless schemes to escrowless identity-based schemes in order to prove their benefit in this area. Note that the only real difference between identity-based and certificateless schemes here is that public keys are exchanged in the certificateless setting.

5.5.2 Computational Complexity

In evaluating the computational complexity, we look at the number of passes and the number of computations each party has to perform in order to carry out the key agreement. In the proposed protocol, each party is required to perform three scalar point multiplications (m), evaluate two bilinear pairings (p), and make one pairing exponentiation (e). Generally, point multiplications and pairing exponentiations are much faster to compute than pairings. Thus, the efficiency of pairing-based protocols is essentially measured by the number of pairings each party has to compute.

Many of the operations used in a protocol can be performed outside a protocol run. Such operations may include the hash of a peer identity or generating a short-term key

[5]This factor is just an estimate as implementations may also store other elements such as the identity/identifier of the public key owner.

Protocol	type	no precomputation	precomputation
Smart[58, 15]	ID	$2p + 2m + 1e$	$1p + 1m$
Chen-Kudla[15] #2'	ID	$1p + 4m$	$1p + 1m$
Shim[57, 61]	ID	$1p + 3m$	$1p + 1m$
Choie-Yeong-Lee[19] #2	ID	$2p + 4m$	$1p + 2m + 1e$
Al-Riyami-Paterson[1]	CL	$4p + 2m + 1e$	$4p + 1m$
Proposed protocol	CL	$2p + 3m + 1e$	$2p + 2m$

Table 3: Computation comparison

pair to be used in some future session. Hence, when talking about efficiency, we often differ between *precomputation* and no precomputation. Protocols that allow precomputation keep the operations required during on-line interaction to a minimum, and can thus significantly improve the response time. However, one drawback of using precomputation is that precomputed values can be compromised by an adversary if they are stored insecurely.

Table 3 shows the computation required by a selection of certificateless and identity-based key agreement protocols. In comparing the computation used in each protocol, only heavy operations are considered. In the proposed protocol, it is possible to precompute the short-term key pair as well as the pairing exponentiation. However, as the public keys are not known before a first-time protocol run between two entities, operations requiring the knowledge of these cannot be precomputed. On the other hand, once public keys have been exchanged, the computation required by each entity can be reduced to only one pairing and one scalar point mulitiplication. Thus, the performance of certificateless protocols can be competitive to that of identity-based protocols. Note from Table 3 that our protocol only needs to compute two pairings, whereas AP's protocol requires each entity to compute four pairings. This is because AP's protocol includes steps to verify the integrity of public keys, and thus restricts the scheme to having private keys generated by the same KGC. We also want to point out that our protocol requires parties to perform one additional point mulitiplication over AP's protocol. This is because it addresses known session-specific temporary information security. AP's protocol can also achieve this property by adopting the method used in our protocol.

It has been believed by many that bilinear pairings introduce a significant computational load to protocols. For instance, [63, 18] propose pairing-less identity-based key agreement protocols under the assumption that pairings could be too expensive to implement in low-power devices. A certificateless public key encryption scheme that does not rely on pairings was also proposed in [4]. However, the implementation of pairings is a very active area of research and much work has been done to improve the efficiency of pairings. As a result, Scott *et al.* [53] showed that pairings can be computed just as efficiently as classic cryptographic primitives on low-power devices such as smart cards. Moreover, Chen *et al.* [14] consider the efficiency of identity-based key agreement protocols using different types of pairings.

6 Future Work

Certificateless public key cryptography was only proposed in 2003, and thus many problems remain to be solved. One is to formalize a security model for certificateless authenticated two-party key agreement and establish a security proof for the proposed protocol. Note that the model presented in this thesis is not a complete model, but meant to be used as a basis for developing a fully functional security model for certificateless key agreement.

Participants of certificateless schemes, unlike identity-based schemes, must exchange public keys. Unfortunately, there are still some open problems about public keys in CL-PKC. For instance, as public keys are not directly linked to the identity of their owners, it is hard to tell a real public key from a fake public key. Thus, public keys can be replaced by an adversary in a certificateless scheme without the parties arising any suspicion.

In terms of efficiency, it would be desirable to use as few bilinear pairings as possible in a protocol. Some identity-based key agreement schemes only require each entity to compute one pairing, while in the proposed certificateless protocol, each entity must compute two pairings. Thus, it would be interesting to see if certificateless key agreement protocols using only one pairing can be constructed (for instance, by adopting existing identity-based schemes where only one pairing is used).

Another point of note, in regards to efficiency, is the use of Map-To-Point functions. In identity-based and certificateless cryptography, these functions are used to convert a hash value of an entity identifier (such as an e-mail address) to a point on an elliptic curve (over some finite field). Apparently, Map-To-Point operations are relatively expensive, and performance could be improved if the function mapped to an element h of the cyclic group $\mathbb{Z}_q^*$ instead. This idea was first presented by Sakai and Kasahara [50] in the SK key construction. McCullagh and Berreto [39] proposed an identity-based key agreement scheme that adopts the idea, but the security of the scheme is somewhat troublesome [16]. Still, it would be interesting to see if secure certificateless key agreement protocols could adopt this idea.

7 Conclusion

This thesis has proposed a certificateless authenticated two-party key agreement protocol that does not use certificates nor suffer from the key escrow property of identity-based cryptography. Each entity involved in the protocol is only required to compute two bilinear pairings, and thus it can be considered more efficient than the protocol proposed by Al-Riyami and Paterson [1]. Moreover, the public keys of the scheme no longer comprise the KGC master key and thus the protocol can be used to establish keys between users of distinct domains (under different KGCs). The protocol is also bandwidth efficient in the respect that public keys only include one group element.

The proposed protocol has been shown to obtain a set of security attributes that are generally believed to be necessary for authenticated key agreement. This includes considering the compromise of session-specific temporary information (ephemeral keys) that can occur in practical implementations if such information is precomputed or stored insecurely. Certificateless key agreement can obtain such security through simple measures, demonstrated by the proposed protocol. The protocol also obtains other security attributes such as known session key security, perfect forward secrecy, key-compromise impersonation, unknown key share, and key control.

As public keys need to be exchanged in certificateless public key cryptography, it should be assumed that an adversary can replace these. We have defined a security model for certificateless authenticated key agreement in which an adversary cannot both replace the public key of a user and obtain its partial private key. Allowing this action will enable the adversary to compute the established session key. Note that this is not a protocol flaw, but rather a deficiency of CL-PKC.

7.1 Answering the Research Questions

In Section 1.4, four research questions were defined. These questions will now be answered, based on the conclusions drawn and the information provided in the thesis.

1. *How are key agreement protocols designed in certificateless public key cryptography, and can existing identity-based schemes be adopted?*
   Certificateless key agreement protocols can look very much similar in structure to identity-based key agreement protocols. However, there are a few differences. For instance, in certificateless cryptography, public keys need to be exchanged as they are no longer derived directly from a user's identity. Each user also has a secret value incorporated into its private key such that the (un)trusted authority cannot obtain private keys using the master key. In designing certificateless key agreement protocols, it is to some extent possible to adopt existing identity-based schemes. For instance, Al-Riyami and Paterson's [1] certificateless authenticated key agreement protocol adopts Smart's [58] identity-based scheme (Table 2 indicates that the session key construction is essentially the same). The proposed protocol can also be considered a variant of Smart's.
2. *What are the possible attacks on a certificateless authenticated two-party key agreement*

protocol?
A key agreement protocol may be susceptible to a number of different attacks. Many of these have been identified in Section 3.2.3 which can be addressed by ensuring that a protocol obtains a set of security attributes, such as those of Section 5.4.6. However, such attributes do not cover all possible attacks. For instance, [57] was believed to attain all desirable security attributes, but was later found flawed by [61] who mounted a man-in-the-middle attack on the protocol. The security of key agreement protocols depend also on the security model used and the power of the adversary. All protocols can be proven secure in some model given some assumptions. In the certificateless setting, it is required that an adversary cannot both replace public keys and obtain the corresponding private keys.

3. *How is proper authentication achieved in certificateless key agreement protocols?*
Authentication in certificateless key agreement can be achieved in the same way as in identity-based key agreement. As the identity of an entity is unique and incorporated into the private/public key pair, any entity can be assured that the owner of a public key is who he or she claims to be. The desired authentication can thus be obtained by requiring the owner of the public key to use its corresponding private key in generating the shared secret. This is true as long as a public key is not replaced, which can occur in the certificateless setting. However, replacing a public key will not provide the attacker with anything useful unless the corresponding (partial) private key is obtained.

4. *How does the efficiency and security of certificateless key agreement measure up against identity-based key agreement?*
Certificateless key agreement can be considered more secure than identity-based key agreement as the KGC does not know every user's private key. Although identity-based schemes can be made escrowless, meaning that the trusted authority cannot obtain the estblished session key between two entitites, they still offer limited security against an adversary armed with the master key. On the other hand, identity-based schemes can be considered more efficient than their certificateless counterparts as they do not have to exchange or store public keys. However, once public keys have been exchanged in certificateless protocols, the computational overhead in the two models can essentially be the same.

Bibliography

[1] S.S. Al-Riyami and K. Paterson. Certificateless Public Key Cryptography. In C. S. Laih, editor, *Advances in Cryptology - Asiacrypt 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 452-473. Springer-Verlag, 2003.

[2] S.S. Al-Riyami and K. Paterson. CBE from CL-PKE: A Generic Construction and Efficient Schemes. In S. Vaudenay, editor, *PKC 2005*, volume 3386 of *Lecture Notes in Computer Science*, pages 398-415. Springer-Verlag, Berlin 2005.

[3] S.S. Al-Riyami. Cryptographic Schemes based on Elliptic Curve Pairings. Ph.D. Thesis, Royal Holloway, University of London, 2004.

[4] J. Baek, R. Safavi-Naini, and W. Susilo. Certificateless Public Key Encryption Without Pairing. In *Proc. of the 8th Information Security Conference (ISC 2005)*, volume 3650 of *Lecture Notes in Computer Science*, pages 134-148. Springer-Verlag, 2005.

[5] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Proceedings of Advances in Cryptology - Crypto '93*, volume 773 of *Lecture Notes in Computer Science*, pages 232-249. Springer-Verlag, 1993.

[6] S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In *Proceedings of the sixth IMA International Conference on Cryptography and Coding*, volume 1355 of *Lecture Notes in Computer Science*, pages 30-45. Springer-Verlag, 1997.

[7] S. Blake-Wilson and A. Menezes. Authenticated Diffie-Hellman Key Agreement Protocols. In *Proc. of the 5th Annual Workshop on Selected Areas in Cryptography (SAC '98)*, volume 1556 of *Lecture Notes in Computer Science*, pages 339-361. Springer-Verlag, 1998.

[8] D. Boneh. The Decision Diffie-Hellman Problem. In *Proceedings of the Third International Symposium on Algorithmic Number Theory*, volume 1423 of *Lecture Notes in Computer Science*, pages 48-63. Springer-Verlag, 1998.

[9] D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213-229. Springer-Verlag, 2001.

[10] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. In *Proceedings of Asiacrypt 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514-32. Springer-Verlag, 2001.

[11] D. Boneh and R. Venkatesan. Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes. In N. Koblitz, editor, *Advances in Cryptology - CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 126-142. Springer-Verlag, 1996.

[12] C. Boyd. Towards extensional goals in authentication protocols. In *Proceedings of the 1997 DIMACS Workshop on Design and Formal Verification of Security Protocols*. 1997.

[13] M. Burmester. On the risk of opening distributed keys. In Y. Desmedt, editor, *Advances in Cryptology - CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, pages 308-317. Springer-Verlag, 1994.

[14] L. Chen, Z. Cheng, and N.P. Smart. Identity-based Key Agreement Protocols From Pairings. Cryptology ePrint Archive, Report 2006/199.

[15] L. Chen and C. Kudla. Identity Based Authenticated Key Agreement Protocols from Pairings. In *Proc. 16th IEEE Security Foundations Workshop*, pages 219-233. IEEE Computer Society Press, 2003.

[16] Z. Cheng and L. Chen. On Security Proof of McCullagh-Barreto's Key Agreement Protocol and its Variants. Cryptology ePrint Archive. Report 2005/201.

[17] Z. Cheng, M. Nistazakis, R. Comley, and L. Vasiu. On The Indistinguishability-Based Security Model of Key Agreement Protocols - Simple Cases. In *Proc. of ACNS 04*, June 2004.

[18] K.Y. Choi, J.Y. Hwang, D.H. Lee, and I.S. Seo. ID-based Authenticated Key Agreement for Low-Power Mobile Devices. In *Proceedings of the 10th Australasian Conference on Information Security and Privacy - ACISP 2005*, volume 3574 of *Lecture Notes in Computer Science*, pages 494 - 505. Springer-Verlag, 2005.

[19] Y. Choie, E. Jeong, and E. Lee. Efficient identity-based authenticated key agreement protocol from pairings. *Applied Mathematics and Computation*, 162:179-188, 2005.

[20] A.W. Dent and C. Kudla. On Proofs of Security for Certificateless Cryptosystems. Cryptology ePrint Archive, Report 2005/426.

[21] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE TIT*, vol. 22, (1976), pages 644-654.

[22] X. Du, Y. Wang, J. Ge, and Y. Wang. An Improved ID-based Authenticated Group Key Agreement Scheme. Cryptology ePrint Archive, Report 2003/260.

[23] R. Dutta, R. Barua, and P. Sarkar. Pairing-Based Cryptographic Protocols: A Survery. Cryptology ePring Archive, Report 2004/064.

[24] G. Frey and H. Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of Computation*, 62:865-874, 1994.

[25] 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.

[26] S.D. Galbraith, K.G. Paterson, and N.P. Smart. Pairings for Cryptographers. Cryptology ePrint Archive, Report 2006/165.

[27] M. Girault. Self-certified public keys. In *Advances in Cryptology - EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 490-497. Springer-Verlag, 1992.

[28] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270-299, 1984.

[29] P. Gutmann. Everything you Never Wanted to Know about PKI but were Forced to Find Out. University of Auckland. See `http://www.govis.org.nz/insecurity/p-gutmann.pdf`. Last visited 30th June 2006.

[30] P. Gutmann. PKI: It's Not Dead, Just Resting. *IEEE Computer*, 35(8), pages 41-49, 2002.

[31] G. Horn, K.M. Martin, and C.J. Mitchell. Authentication Protocols for Mobile Network Environment Value-added Services. *IEEE Transactions on Vehicular Technology*, 51:383-392, 2002.

[32] A. Joux. A One Round Protocol for Tripartite Diffie-Hellman. In W. Bosma, editor, *Proceddings of the 4th Algorithmic Number Theory Symposium*, volume 1838 of *Lecture Notes in Computer Science*, pages 385-394. Springer, 2000.

[33] A. Joux. The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems. In C. Fieker and D. R. Kohel, editors, *Proc. Algorithmic Number Theory, 5th International Symposium (ANTS-V)*, volume 2369 of *Lecture Notes in Computer Science*, pages 20-32. Springer-Verlag, 2002.

[34] A. Jules and J. Brainard. Client puzzles: A cryptographic defense against connection depletion attacks. In *Proceedings of the 1999 ISOC Network and Distributed System Security Symposium*, pages 151-165, 1999.

[35] L. Kohnfelder. Toward a Practical Public-Key Cryptosystem. Bachelor's thesis, EECS Dept., Massachusetts Institute of Technology, May, 1978.

[36] H. Krawczyk. HMQV: A High-Performance Secure Diffie-Hellman Protocol. In *Advances in Cryptology - CRYPTO'05*, volume 3621 of *Lecture Notes in Computer Science*, pages 546-566. Springer-Verlag, 2005.

[37] K. Lauther and A. Mityagin. Security Analysis of KEA Authenticated Key Exchange Protocol. *PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 378-394. Springer-Verlag, 2006.

[38] 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.

[39] N. McCullagh and P.S.L.M. Barreto. A new two-party identity-based authenticated key agreement. In A.J. Menezes, editor, *Cryptographers' Track at RSA Conference - CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 262-274. Springer-Verlag, 2005.

[40] A. Menezes. Cryptography Using Bilinear Maps. University of Waterloo. 2003.

[41] A. Menezes, T. Okamoto, and S. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39:1639-1646, 1993.

[42] A. Menezes, M. Qu, and S. Vanstone. Some new key agreement protocols providing mutual implicit authentication. In *Proceedings of the Second Workshop on Selected Areas in Cryptography (SAC '95)*, pages 22-32. 1995.

[43] C.J. Mitchell, M. Ward, and P. Wilson. Key control in key agreement protocols. *Electronics Letters*, 34:980-981, 1998.

[44] NIST. SKIPJACK and KEA Algorithm Specification. See http://csrc.nist.gov/encryption/skipjack/skipjack.pdf. Last visited 30th June 2006.

[45] S. Pohlig and M. Hellman. An improved algorithm for computing discrete logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, 24:106-110, 1978.

[46] J. M. Pollard. Monte Carlo methods for index computation (modp). *Mathematics of Computation*, 32(143):918-924, July 1978.

[47] J. Raymond and A. Stiglic. Security Issues in the Diffie-Hellman Key Agreement Protocol. 2000.

[48] B. van Rompay. Analysis and Design of Cryptographic Hash Functions, MAC Algorithms and Block Ciphers. Ph.D. thesis. Katholieke Universiteit Leuven.

[49] E. Ryu, E. Yoon, and K. Yoo, An Efficient ID-Based Authenticated Key Agreement Protocol. *Networking 2004*, volume 3042 of *Lecture Notes in Computer Science*, pages 1458-1463. Springer-Verlag, 2004.

[50] R. Sakai and M. Kasahara. ID based cryptosystems with pairing on elliptic curve. In *2003 Symposium on Cryptography and Information Security - SCIS'2003*, Hamamatsu, Japan, 2003.

[51] D. Salomon. Data privacy and security. Springer-Verlag New York, 2003.

[52] M Scott. Authenticated ID-based key exchange and remote log-in with insecure token and PIN number. Cryptology ePrint Archive, Report 2002/164.

[53] M. Scott, N. Costigan, and W. Abdulwahab. Implementing Cryptographic Pairings on Smartcards. Cryptology ePrint Archive, Report 2006/144.

[54] A. Shamir. Identity-based cryptosystems and signature schemes, In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology - CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47-53. Springer-Verlag, 1985.

[55] Y. Shi, G. Chen, and J. Li. ID-Based One Round Authenticated Group Key Agreement Protocol with Bilinear Pairings. *ITCC (1) 2005*, pages 757-761. 2005.

[56] K. Shim. Cryptanalysis of Two ID-based Authenticated Key Agreement Protocols from Pairings. Cryptology ePrint Archive, Report 2005/357.

[57] K. Shim. Efficient ID-based authenticated key agreement protocol based on the Weil pairing. *Electronics Letters*, 39(8):653-654, 2003.

[58] N.P. Smart. An identity based authenticated key agreement protocol based on the Weil pairing. *Electronics Letters*, 38:630-632, 2002.

[59] E. 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.

[60] G. Xie. An ID-based key agreement scheme from pairing. Cryptology ePrint Archive, Report 2005/093.

[61] Q. Yuan and S. Li. A New Efficient ID-Based Authenticated Key Agreement Protocol. Cryptology ePrint Archive, Report 2005/309.

[62] Z. Zhang and D. Feng. On the Security of a Certificateless Public-Key Encryption. Cryptology ePrint Archive, Report 2005/426.

[63] R.W. Zhu, G. Yang, and D.S. Wong. An Efficient Identity-based Key Exchange Protocol with KGS Forward Secrecy for Low-power Devices. In *Proceedings of the 1st Workshop on Internet and Network Economics (WINE 2005)*, volume 3828 of *Lecture Notes in Computer Science*, pages 500-509. Springer-Verlag, 2005.

A Certificateless PKE Schemes

A.1 The Basic CL-PKE Scheme

Seven algorithms constitute what is formally specified as the BasicCL-PKE scheme [1].

`Setup` runs as follows:

1. Run $\mathcal{IG}$ on input k to genereate output $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ where $\mathbb{G}_1$ and $\mathbb{G}_2$ are groups of some prime order q and $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ 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 $H_1 : \{0,1\}^* \to \mathbb{G}_1^*$ and $H_2 : \mathbb{G}_2 \to \{0,1\}^n$, where n will be the bit-length of plaintexts.

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

`Partial-Private-Key-Extract` takes as input an identifier $\texttt{ID}_A \in 0,1^n$ and outputs the partial private key D_A for entity A. The key is constructed by first computing $Q_A = H_1(\texttt{ID}_A) \in \mathbb{G}_1^*$ and then computing $D_A = sQ_A \in \mathbb{G}_1^*$. A can verify the correctness of the algorithm by checking $\hat{e}(D_A, P) = \hat{e}(Q_A, P_0)$.

`Set-Secret-Value` takes `params` and an entity A's identifier $\texttt{ID}_A$ as input and outputs A's secret value $x_A \in \mathbb{Z}_q^*$.

`Set-Private-Key` takes `params`, an entity A's partial private key D_A, and A's secret value $x_A \in \mathbb{Z}_q^*$ as input, and returns the (full) private key S_A by computing $S_A = x_A D_A = x_A s Q_A \in \mathbb{G}_1^*$.

`Set-Public-Key` takes `params` and entity A's secret value $x_A \in \mathbb{Z}_q^*$ as input and constructs A's public key $P_A = \langle X_A, Y_A \rangle$, where $X_A = x_A P$ and $Y_A = x_A P_0 = x_A sP$.

`Encrypt`: To encrypt $M \in \mathcal{M}$ for entity A with identifier $\texttt{ID}_A \in \{0,1\}^*$ and a 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 $\hat{e}(X_A, P_0) = \hat{e}(Y_A, P)$. If not, output $\perp$ and abort the encryption.
2. Compute $Q_A = H_1(\texttt{ID}_A) \in \mathbb{G}_1^*$.
3. Choose a random value $r \in \mathbb{Z}_q^*$.
4. Compute and output the ciphertext:

$$C = \langle rP, M \oplus H_2(e(Q_A, Y_A)^r) \rangle$$

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

$$V \oplus H_2(e(S_A, U))$$

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 the decryption is the inverse of encryption

$$\begin{aligned} V \oplus H_2(e(S_A, U)) &= V \oplus H_2(e(x_A sQ_A, rP)) \\ &= V \oplus H_2(e(Q_A, x_A sP)^r) \\ &= V \oplus H_2(e(Q_A, Y_A)^r) \\ &= M \end{aligned}$$

A.2 The FullCL-PKE Scheme

FullCL-PKE is in many ways similar to BasicCL-PKE, but adds chosen ciphertext security by adapting the Fujisaki-Okamoto padding technique.

`Setup`: Identical to `BasicCL-PKE`, but adds two additional cryptographic hash functions $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}_q^*$ and $H_4 : \{0,1\}^n \to \{0,1\}^n$.

The system parameters are $\texttt{params} = \langle \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, H_1, H_2, H_3, H_4\rangle$. The `master-key` and the message space is identical to `BasicCL-PKE`. The ciphertext space is $C = \mathbb{G}_1 \times 0,1^{2n}$.

`Partial-Private-Key-Exract`: Identical to `BasicCK-PKE`.

`Set-Secret-Value`: Identical to `BasicCK-PKE`.

`Set-Private-Key`: Identical to `BasicCK-PKE`.

`Set-Public-Key`: Identical to `BasicCK-PKE`.

`Encrypt`: To encrypt $M \in \mathcal{M}$ for entity A with identifier $\texttt{ID}_A \in \{0,1\}^*$ 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 $\hat{e}(X_A, P_0) = \hat{e}(Y_A, P)$. If not, output $\perp$ and abort the encryption.
2. Compute $Q_A = H_1(\texttt{ID}_A) \in \mathbb{G}_1^*$.
3. Choose a random $\sigma \in \{0,1\}^n$.
4. Set $r = H_3(\sigma, M)$.
5. Compute and output the ciphertext:

$$C = \langle rP, \sigma \oplus H_2(e(Q_A, Y_A)^r), M \oplus H_4(\sigma)\rangle$$

`Decrypt`: Suppose $C = \langle U, V, W\rangle \in \mathcal{C}$. Do decrypt the ciphertext using the private key $S_A = x_A D_A$, perform the following steps:

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 cipertext.
4. Output M' as the decryption of C.

A.3 The improved FullCL-PKE Scheme (FullCL-PKE*)

In [2], the certificateless public key encryption (CL-PKE) scheme of [1] was improved in two ways. Firstly, it is more efficient than the scheme in [1] as the public key now only constitute one group element. Secondly, the security of the scheme rests on the

hardness of the well studied Bilinear Diffie-Hellman Problem (BDHP), rather than the non-standard generalized BDHP.

`Setup` takes security parameter k as input and returns the system parameters $\texttt{params} = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_0, H_1, H_2, H_3, H_4, H_5 \rangle$ and the `master-key` is $s \in \mathbb{Z}_q^*$. The message space is $M = 0,1^n$ and the ciphertext space is $C = \mathbb{G}_1 \times 0,1^{2n}$.

`Partial-Private-Key-Extract` takes as input an identifier $\texttt{ID}_A \in 0,1^n$ and outputs the partial private key D_A for entity A. The key is constructed by first computing $Q_A = H_1(\texttt{ID}_A) \in \mathbb{G}_1^*$ and then computing $D_A = sQ_A \in \mathbb{G}_1^*$. A can verify the correctness of the algorithm by checking $\hat{e}(D_A, P) = \hat{e}(Q_A, P_0)$.

`Set-Secret-Value` takes `params` and an entity A's identifier $\texttt{ID}_A$ as input and outputs A's secret value $x_A \in \mathbb{Z}_q^*$.

`Set-Private-Key` takes `params`, entity A's partial private key D_A, and A's secret value $x_A \in \mathbb{Z}_q^*$ as input, and outputs A's secret key pair $S_A = \langle D_A, x_A \rangle$.

`Set-Public-Key` takes `params` and entity A's secret value $x_A \in \mathbb{Z}_q^*$ as input and constructs A's public key $P_A = x_A P$.

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

1. Verify that $P_A \in \mathbb{G}_1^*$, or output $\perp$.
2. Compute $Q_A = H_1(\texttt{ID}_A) \in \mathbb{G}_1^*$.
3. Choose a random $\sigma \in \{0,1\}^n$.
4. Set $r = H_3(\sigma, M)$.
5. Compute and output the ciphertext:

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

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

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. Otherwise, output M' as the decryption of C.