Cryptographic Applications of Bilinear Maps

Noel Michael McCullagh

B Sc , M Sc

A thesis submitted for the degree of

Ph D

to the

Dublin City University

Faculty of Engineering and Computing

School of Computing

Supervisor Dr Michael Scott

October 2005

Declaration

I, Noel Michael McCullagh, hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ph.D. is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

N. McCullagh

Name: Noel Michael McCullagh
ID: 97903183
Date: October, 2005

The original work in this thesis is as follows:

1. Ch. 3. Sec. 3.10 was joint work with Chevallier-Mames, Coron, Naccache and Scott.

2. Ch. 5. Sec. 5.5 was joint work with Barreto, Libert and Quisquater.

3. Ch. 6. Sec. 6.2.2 is my own work, Section 6.5 was joint work with Libert and Quisquater.

4. Ch. 7. Sec. 7.5.1 is my own work and Section 7.6 was joint work with Barreto.

5. Ch. 8. Sec. 8.6 was joint work with Barreto, Libert and Quisquater.

6. Appendix. A full Java library of pairing based cryptography software was created.

Acknowledgements

There are many people I need to thank for making the last three years so enjoyable

Firstly, I wish to sincerely thank my supervisor Mike, without whose support I would never have made it this far Mike has been a great support to me, not only with his vast knowledge of the area, but also with his constant encouragement and enthusiasm when things got tough Thank you

I also wish to sincerely thank my very good friend Neil Costigan I met Neil for the first time when we both started research studies in the university three years ago Since then Neil has become so much more than a work colleague He is one of the most enthusiastic, energetic, motivating and fun people I know He has made a huge difference to the last three years He has been the first to cheer me up when I am feeling down, and the first to congratulate me on the successes Thank you

I would like to thank the other postgraduate students - Barry Cameron, Cara, Claire K, Claire W, Dalen, David, Dave, Hego, Karl, Katrina, Mark, Mary, Michelle, Niall, Noreen, Riona, Ronan, Sara, Thibault & Tommy

My thanks also to my good friends Scott & Yann and their families They have made my years in Dublin so enjoyable

I would also like to thank my co-authors Paulo S L M Barreto Benoit Chevallier-Mames, Jean-Sebastien Coron, Benoit Libert, David Naccache and Jean-Jacque Quisquater

Last but definitely not least, I would especially like to thank my Mum and Dad, my brothers Mark and John and my sister Laura They have been a huge support to me, and have *always* been there for me Thank you so much

Abstract

Bilinear maps have become an important new item in the cryptographer's toolkit They first came to prominence when they were used by Menezes, Okamoto and Vanstone to help solve the elliptic curve discrete logarithm problem on elliptic curves of small embedding degree

In 1984, Shamir developed the first identity based signature scheme, and posed the construction of an identity based encryption scheme as an open problem [118] Subsequently identity based identification and identity based key agreement schemes were proposed However, identity based encryption remained an open problem In 2000, Sakai, Ohgishi and Kasahara used bilinear maps to implement an efficient identity based non-interactive key agreement and identity based digital signature [111] In 2001, some 17 years after it was suggested, Boneh and Franklin proposed the first efficient identity based encryption scheme, constructed using bilinear maps [31]

In this thesis we review some of the numerous cryptographic protocols that have been constructed using bilinear maps

We first give a review of public key cryptography We then review the mathematics behind the two known bilinear maps, the Weil and Tate pairings, including several improvements suggested in [67, 14] We develop a Java library to implement pairing based cryptography In Ch 4 we look at some of the cryptographically hard problems that arise from bilinear maps In Ch 5 we review identity based signature schemes and present the fastest known scheme In Ch 6 we review some encryption schemes, make some observations that help improve the performance of many identity based cryptosystems, and propose the fastest scheme for public key encryption with keyword search In Ch 7 we review identity based key agreements and propose the fastest scheme secure in a modified Bellare-Rogaway model [19] In Ch 8 we review identity based signcryption schemes and present the fastest known scheme

Contents

List of Algorithms

Chapter 1

Introductory Mathematical Background

1 1 Modular Arithmetic

Nearly all modern cryptographic systems require a basic understanding of modular arithmetic The idea behind modular arithmetic is very simple and most primary school children are familiar with it from the concept of a clock face They learn to convert between 12 and 24 hour clock representation This is an example of congruence modulo 12, where 13 00 in the 24 hour representation can be converted to 01 00 in the 12 hour representation

Formally we work in the positive integers, including zero[1] We fix a positive integer modulus N and work with the set of integers $\{0, 1, \quad , N-1\}$ This is the set of integers *modulo* N Any numbers a and b, that are related as $a = b + xN$, for some integer x, are said to be *congruent* modulo N Congruence is usually denoted $\equiv$ That is, using our clock face example, $13 \equiv 1 \mod 12$ If it is obvious that we are working modulo N we may just say that $a = b \pmod N$ or $a = b$

When working modulo N we can also consider negative numbers, but it is the convention to write them positively Again, we use the 12 hour clock face for our example Say that

[1] In a 24 hour clock, 12 midnight will be shown as zero, 00 00

we wish to take 2 hours away from 1 o'clock That is, what is 2 hours before 1 o'clock? If we think of the clock face then we realise that this would be expressed as 11 o'clock What has happened is that we think of 12, the modulus, as 0 Formally this means $N - a \equiv -a \mod N$ Informally, using our clock-face example, $11 \equiv -1 \mod 12$ If $a < 0$ or $a \geq N$ we add or subtract some multiple of N until we have a number in the range $\{0, \quad, N-1\}$ This is known as *reduction modulo N* This set of integers can be written formally as $\mathbb{Z}/N\mathbb{Z}$, or $\mathbb{Z}_N$

1.2 Infinite Groups

A group $(\mathcal{G}, *)$ consists of a non empty set $\mathcal{G}$ with a binary operator[2] $*$, which satisfies the following properties [125] [91, Ch 2] By way of example, we consider the set of integers, $\mathbb{Z}$ and the binary operation, integer addition, $+$

- **The operation is closed**

$$\forall a, b \in \mathcal{G} \quad a * b \in \mathcal{G} \tag{1 1}$$

$$5 + 4 = 9 \tag{1 2}$$

- **The operation is associative**

$$\forall a, b, c \in \mathcal{G} \quad (a * b) * c = a * (b * c), \tag{1 3}$$

$$(5 + 4) + 3 = 5 + (4 + 3) \tag{1 4}$$

- **The set $\mathcal{G}$ contains an identity element** An identity element e is one that has the property

$$\forall a \in \mathcal{G} \quad (e * a) = a, \tag{1 5}$$

$$0 + 5 = 5 \tag{1 6}$$

[2]binary operation an operation taking two operands

- **The unique existence of an inverse element** Each element in the group $\mathcal{G}$ has a unique inverse The inverse of an element is an element in $\mathcal{G}$, such that the following property holds, where b is the inverse of a and e is the identity element defined previously

$$\forall a \in \mathcal{G}\ \exists b \in \mathcal{G}\quad (a * b) = e, \tag{1 7}$$

$$5 + (-5) = 0 \tag{1 8}$$

A group has all of the above four properties Some groups also have the following property

- **The operation is commutative**

$$\forall a, b \in \mathcal{G}\quad (a * b) = (b * a), \tag{1 9}$$

$$5 + 4 = 4 + 5 \tag{1 10}$$

A group that is also commutative is called an *abelian* group Most of the groups that we use in cryptography are abelian, as it is this last property that makes them cryptographically useful[3] We will assume that all groups that we discuss in the remainder of this thesis are abelian

A group is called *multiplicative* if we tend to write its group operation as , whereas a group where we tend to write its group operation as $+$ is called *additive* This will also effect the way that we write the identity element and the inverse element

For a *multiplicative* group we have

- **Identity element** The identity element is written as 1

$$\forall a \in \mathcal{G}\quad (a\ \ 1) = a \tag{1 11}$$

[3]In most cases in cryptography we are only interested in groups where $g^{xy} = g^{yx}$

- **Inverse element** The inverse element is written as a^{-1}

$$\forall a \in \mathcal{G} \quad (a \cdot a^{-1}) = 1 \tag{1.12}$$

- **Repeated application of the group operator** A shorthand notation for repeated multiplication is exponentiation

$$\forall a \in \mathcal{G} \qquad \underbrace{(a \cdots a)}_{n \text{ times}} = a^n \tag{1.13}$$

For an *additive* group we have

- **Identity element** The identity element is written as 0

$$\forall a \in \mathcal{G} \quad (a + 0) = a \tag{1.14}$$

- **Inverse element** The inverse element is written as $-a$

$$\forall a \in \mathcal{G} \quad (a + (-a)) = 0 \tag{1.15}$$

- **Repeated application of the group operator** A shorthand notation for repeated addition is scalar multiplication

$$\forall a \in \mathcal{G} \qquad \underbrace{(a + \cdots + a)}_{n \text{ times}} = n \cdot a \tag{1.16}$$

1 3 Infinite Fields

A *ring* is a set $\mathcal{A}$ with two operations, usually denoted $+$ and $\cdot$, for addition and multiplication [91, Ch 2] The ring is usually denoted $(\mathcal{A}, \cdot, +)$ The addition operation has the same properties as it had when it was previously defined for groups If it happens that multiplication is commutative then we say that the ring is commutative [125] By definition a ring operation will be closed It should be obvious that $(\mathbb{Z}, \cdot, +)$ - the set of integers, $(\mathbb{Q}, \cdot, +)$ - the set of rational numbers, and $(\mathbb{R}, \cdot, +)$ - the set of real numbers, are all infinite commutative rings

If the ring has a multiplicative identity then we say it is a *ring with identity*

A *field* is a ring such that

- $(\mathcal{G}, +)$ is an abelian group with identity denoted by 0
- $(\mathcal{G}\backslash\{0\}, \cdot)$ is an abelian group, with identity denoted by 1
- $(\mathcal{G}, \cdot, +)$ satisfies the distributive law

$$a \cdot (b+c) = (a \cdot b) + (a \cdot c) = (b+c) \cdot a \tag{1 17}$$

Therefore, a field is a commutative ring for which every non-zero element has a multiplicative inverse

1 4 Finite Groups and Fields

A group is *finite* if it has a finite number of elements in its set [83, Ch 1] The *order* of a finite group $\mathcal{G}$ is the number of elements in its set, and is denoted $|\mathcal{G}|$ or $\#\mathcal{G}$ An abelian group $(\mathcal{G}, *)$ is called *cyclic* if there is some element α, from which every other element in the group can be obtained though repeated application of the group operation Such an element is called a *generator* of $\mathcal{G}$ Mathematically we denote that g is a generator of the group $\mathcal{G}$ as

$$\langle g \rangle = \mathcal{G} \tag{1 18}$$

For an additive group this means

$$y = x \cdot g \tag{1 19}$$

and for a multiplicative group this means

$$y = g^x \tag{1 20}$$

where y can be any element of $\mathcal{G}$ y obviously depends on x x is called the *discrete logarithm* of y with respect to (the base) g

The *order* of an element g, of a finite cyclic group is the smallest non-zero integer t such that $g^t = e$, the identity element

A group $(\mathcal{G}, *)$ may contain a number of *subgroups* A group $(\mathcal{K}, *)$ is a subgroup of $(\mathcal{G}, *)$ if it itself is a group with respect to the group operation $*$ (to recap that means that it is *closed, has an identity, every element has an inverse and the group operation is associative*) and $\mathcal{K}$ is a subset of $\mathcal{G}$ The order of a group $\mathcal{K}$ will always divide the order of $\mathcal{G}$ $(\mathcal{K}, *)$ is called a *proper* subgroup if $\mathcal{K} \neq \mathcal{G}$

An element $x \in \mathbb{Z}_N$ has a multiplicative inverse modulo N if and only if the *greatest common divisor* $\gcd(x, N) = 1$ We can define the set of all invertible elements (those that have multiplicative inverses) of the set $\mathbb{Z}_N$ as $\mathbb{Z}_N^*$ Formally $\mathbb{Z}_N^*$ is defined as

$$\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N \quad \gcd(x, N) = 1\} \tag{1 21}$$

We would like to know how many elements are in $\mathbb{Z}_N^*$ This is given by Euler's Phi Function[4] $\phi(N)$, which for any integer N returns the number of integers that are smaller than and co-prime to N

To determine the Euler ϕ function for an integer N we must be able to factor N The number of integers less than and co-prime to a prime p is $(p-1)$ Since, if p is a prime then all of the numbers less than it will have no factors in common with it

Therefore, for any prime p, we have

$$\phi(p) = (p-1), \tag{1 22}$$

$$\mathbb{Z}_p^* = \{x \in \mathbb{Z}_p \quad \gcd(x,p) = 1\} = \{1, \quad p-1\} \tag{1 23}$$

Another group of integers that are of importance to cryptography are the prime powers What is the Euler totient function for any prime power $q = p^m$? The only numbers that are going to have factors in common with q are the multiples of p That is $p, \quad , (p^{m-1})p$ For any prime power there are going to be (p^{m-1}) of these factors [125, Ch 1]

Therefore, for any prime power $q = p^m$, we have

$$\phi(q) = (q) - (p^{m-1}) = (p^m) - (p^{m-1}) = p^{m-1}(p-1) = p^m\left(1 - \frac{1}{p}\right) \tag{1 24}$$

We also know that for any two co-prime numbers n and m

$$\phi(m \quad n) = \phi(m) \quad \phi(n) \tag{1 25}$$

[4] Also called Euler's totient function

Building on the above results, we can determine Euler's totient function for any arbitrary integer for which we have a known factorisation We simply work out Euler's totient function for each of the constituent prime powers and then calculate the product of these terms

$$\phi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right) \tag{1 26}$$

Finding the Members of $\mathbb{Z}_N^*$

If it is possible to quickly factor a and b, then the $\gcd(a, b)$ is given as the product of the factors common to a and b However, this is generally not efficient with integers that are used in industrial strength cryptographic systems[5] To find the gcd of two integers we use the Euclidean algorithm As before, for α to be a member of $\mathbb{Z}_N^*$, $\alpha < N$ and $\gcd(\alpha, N) = 1$

1 4 1 Euclidean Algorithm

The Euclidean algorithm depends on the *division algorithm for integers* [91, Ch 2] The division algorithm makes use of the fact that if a and b are positive integers, there exist unique, non-negative integers q and r such that

$$a = qb + r \quad 0 \le r < b \tag{1 27}$$

This is simple to see given a numerical example, consider $a = 75, b = 34$

$$75 = 2 \quad 34 + 7 \tag{1 28}$$

In the above, q is known as the quotient, and r as a remainder

The Euclidean algorithm, which is used to obtain the gcd of two numbers, works by

[5]Industrial strength cryptography is a vague term, but for RSA moduli $\approx 2^{1024}$ seems like a minimum

repeated application of the division algorithm until the remainder r is 0 To get the gcd of two numbers set the first equal to a and the second equal to b in equation 1 27 Now repeatedly apply the algorithm, at each stage replacing $a_i = b_{i-1}$ and $b_i = r_{i-1}$ This works since every divisor of both a and b will be a divisor of both b and r

Continuing on, we now calculate the gcd of 75 and 34

$$a = qb + r \tag{1 29}$$

$$75 = 2 \times 34 + 7, \tag{1 30}$$

$$34 = 4 \times 7 + 6, \tag{1 31}$$

$$7 = 1 \times 6 + 1, \tag{1 32}$$

$$6 = 6 \times 1 + 0 \tag{1 33}$$

Since $r = 0$, we have that the $\gcd(75, 34) = 1$ (which is the last value of b above) We also know that since the integers are co-prime, 34 has a multiplicative inverse modulo 75 Therefore, 34 is an element of $\mathbb{Z}_{75}^*$

Algorithm 1 1 Euclidean Algorithm

INPUT Positive integers a and b, with $a \leq b$
OUTPUT $\gcd(a, b)$

while $(b \neq 0)$ **do**
  $r \leftarrow a \mod b$
  $a \leftarrow b$
  $b \leftarrow r$
end while
return a

1 4 2 Extension Fields

The *order* of a finite field is the number of elements in the field, and is denoted $\#\mathbb{F}$ for the field $\mathbb{F}$ There exists a finite field $\mathbb{F}$ of order q if and only if q is a prime power, i e $q = p^m$ [125] p is called the characteristic of the field, and is denoted **char** $\mathbb{F}$ If $m = 1$ then we say that $\mathbb{F}$ is a *prime field* If $m \geq 2$ then we say that $\mathbb{F}$ is an *extension field* For any prime power q there is only one field of order q up to *isomorphism* Any two fields of the same order are said to be *isomorphic*, meaning that they are structurally the same It is possible to map between two isomorphic fields (which we denote $\mathcal{F}_1$ and $\mathcal{F}_2$) using a field isomorphism Φ [125, Ch 1]

$$\Phi \quad \mathcal{F}_1 \rightarrow \mathcal{F}_2 \tag{1 34}$$

The mapping Φ has the following structure-preserving properties

$$\Phi(\alpha + \beta) = \Phi(\alpha) + \Phi(\beta), \tag{1 35}$$

$$\Phi(\alpha \quad \beta) = \Phi(\alpha) \quad \Phi(\beta) \tag{1 36}$$

Higher degree extension fields contain all of the elements in $\mathbb{F}_p$ In fact, $\mathbb{F}_{p^e}$ will contain all of the elements of $\mathbb{F}_{p^d}$ for all d dividing e These lower degree extension fields are called *subfields* of the (higher degree) extension field

An isomorphism that maps from a field $\mathcal{F}_1$ to itself, is called an *automorphism*

$$\Phi \quad \mathcal{F}_1 \rightarrow \mathcal{F}_1 \tag{1 37}$$

One particularly interesting automorphism is called the p^{th} *power Frobenius map* It is defined for any finite field as

$$\Phi \begin{cases} \mathbb{F}_{p^k} & \rightarrow & \mathbb{F}_{p^k}, \\ \alpha & \rightarrow & \alpha^p \end{cases} \tag{1 38}$$

where p is the characteristic of the field The set of elements fixed by the Frobenius map acting on extension field $\mathbb{F}_{p^k}$ is the set of elements in the prime field $\mathbb{F}_p$

$$\mathbb{F}_p = \{\Phi(\alpha) \quad \alpha \in \mathbb{F}_{p^k}\} \tag{1 39}$$

1 5 Calculating the Multiplicative Inverse

Finding multiplicative inverses is very important in cryptography It is the basis for determining key pairs in the famous RSA encryption algorithm devised by Rivest Shamir and Adleman [107] Now that we have established which integers have multiplicative inverses we wish to actually determine the multiplicative inverse To do this we use the extended Euclidean algorithm which is given in Sec 1 5 1

1 5 1 Extended Euclidean Algorithm

The extended Euclidean algorithm is a variation on the Euclidean algorithm with some additional bookkeeping information The greatest common divisor of a and b can be expressed as an integer linear combination of a and b That is, there are integers s and t such that

$$\gcd(a, b) = s \quad a + t \quad b \tag{1 40}$$

Now, assume that a is invertible mod b ($\gcd(a, b) = 1$), and that b is larger than a Rewriting the above equation, we have the following

$$1 = s\ a + t\ b, \tag{1 41}$$

$$1 - t\ b = s\ a, \tag{1 42}$$

$$1 = s\ a \mod b \tag{1 43}$$

In other words, s is the multiplicative inverse of $a \mod b$ For finding multiplicative inverses we do not require t Here we give a variation of the extended Euclidean algorithm where t is ignored in the interests of computational efficiency We can calculate the value of s as we work through the Extended Euclidean algorithm The values of s_0 and s_1 are initially set to 0 and 1, subsequent values of s_i are given by $s_{i-2} - s_{i-1}q_{i-2} \mod a_0$

$$a = qb + r \tag{1 44}$$

$$75 = 2 \times 34 + 7, s_0 = 0, \tag{1 45}$$

$$34 = 4 \times 7 + 6, s_1 = 1, \tag{1 46}$$

$$7 = 1 \times 6 + 1, s_2 = 0 - (1 \times 2) \mod 75 = 73, \tag{1 47}$$

$$6 = 6 \times 1 + 0, s_3 = 1 - (73 \times 4) \mod 75 = 9, \tag{1 48}$$

$$s_4 = 73 - (9 \times 1) \mod 75 = 64 \tag{1 49}$$

And so we have calculated that 64 is the multiplicative inverse of 34 modulo 75 This can be checked as

$$34 \times 64 = 2176 \equiv 1 \mod 75 \tag{1 50}$$

The Extended Euclidean Algorithm is given in Algorithm 1 2

Algorithm 1 2 The Extended Euclidean Algorithm for finding the multiplicative inverse of $a \mod b$

INPUT Two integers a and b such that $a \geq b$, $b > 0$ and $\gcd(a, b) = 1$
OUTPUT $a^{-1} \mod b$

```
x1 ← 0
x2 ← 1
y1 ← 1
y2 ← 0
while (b > 0) do
  q ← ⌊a/b⌋
  r ← a − qb
  x ← x2 − q x1
  a ← b
  b ← r
  x2 ← x1
  x1 ← x
end while
return x2
```

1 6 Random Number Generation

Most cryptographic algorithms rely on the ability to produce random numbers The RSA encryption algorithm has a requirement to generate two large random primes El Gamal and discrete logarithm based encryption systems have a requirement that a private key be a random integer in a suitably large interval $\{0, \quad , n\}$ The size of this interval is known as the key space The key space should be large enough that even the most determined adversary cannot search for the actual key used

Suppose we have a truly random 128 bit number (for example, to be used as an AES[6] encryption key) Then, an adversary would have to make on average 2^{127} guesses before her stumbled upon the correct value Even if only one bit of a supposedly random sequence is known then the key space is halved This means that the remaining key space can be searched in half the time

Suppose for example that the attacker knows half of the bits in the key[7], then the key

[6] AES Advanced Encryption Standard, a modern symmetric block cipher and NIST approved replacement for DES, the Data Encryption Standard

[7] Using modern fault and power analysis attacks, this may not be unreasonable

space is only 2^{64}, taking on average 2^{63} guesses before the correct combination is stumbled upon Such a scheme, that was considered secure using a full length random key would no longer be considered secure Therefore a good source of random numbers is critical to the security of all cryptographic systems One of the best attacks on implementations of cryptographic systems is to cripple the random number generator in the system The beauty of this attack is that as long as the output of the random number generator still "looks" random (but actually has some exploitable properties) then the unsuspecting user may continue to use the random number generator for years into the future

There are a number of ways to produce random numbers We give some examples in Sec 1 6 1

1 6 1 Natural Sources of Randomness

There are many natural sources of randomness [60] One that we would all be familiar with is background noise This fluctuates constantly Someone shuffling paper at the desk next-door Someone typing on a keyboard across the office Colleagues discussing work in an open plan office Someone taking a drink from a water cooler Buses, cars and lorries driving past the window These sounds naturally vary throughout the day We can use this naturally occurring randomness to generate random numbers for cryptographic systems

Compact disc audio is encoded at 16 bit resolution, this gives the ability to trace a sound wave though 64K different levels of displacement If we take just the least significant bit of this representation it will be randomly switching from zero to one and back This, in reality, bares little connection to the outside sound and would be extremely hard to manipulate Java code which uses background noise to generate random numbers in included in appendix A

Another natural source of randomness is background radiation There is a small amount of background radiation all the time The time between the emission of particles during radioactive decay is random This can be exploited to create a random number generator Intel's Hardware Random Number Generator uses electrically generated signals that

Figure 1 1 Generating random numbers from a sound wave

are produced randomly in resistors – for example Johnson noise (commonly referred to as thermal noise), shot noise, and flicker noise, which are as a result of random electron and material behaviour The difference in measurement between two resistors placed close to each other is taken, to reduce any effects caused by electromagnetic radiation, temperature, etc[8] [78]

Another natural source of randomness would be a person typing on a keyboard [39] This might, at first sound strange, but we do not look at the words that the person types Instead, we set a timer running and we time when the person presses the individual keys Provided the time increments are small enough then it will be impossible for the person to predict what the least significant bit of the timer will be when they press on the key If you have even tried to stop a 1/1000 sec stopwatch exactly at 1 000 sec you will know how difficult this is Computers can time increments **much** smaller than this

Of course, the above is only an example of the methods that can be used It is also possible to combine the output of several different sources of randomness, for example, by using a one-way (hash) function[9]

1 6 2 Pseudo-Random Number Generators

"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin " - John Von Neumann (1951) [78]

The random number generation methods mentioned above generate *true* random numbers Usually the above methods are not used to generate large quantities of random

[8] It is assumed that radiation will affect both resistors similarly

[9] Hash functions will be discussed in more detail in Ch 5

numbers Instead we use one of the above methods to generate a random seed value ≈ 128 bits in length We can then use this seed value as the basis for generating substantially more pseudo-random bits via a pseudo-random number generator (PRNG) The output of a PRNG is not truly random, but it should appear random Because we are not now working with random sources we introduce a definition that allows us to judge the quality of the randomness produced by our PRNG

Definition [91, Ch 5] A pseudo random bit generator is said to pass the *next-bit test* if there is no polynomial-time algorithm which, on input of the first l bits of an output sequence s, can predict the $(l+1)^{st}$ bit of s with probability significantly greater than 1/2 [91]

The above definition seems ideal, but how do we know that no such algorithm exists? Strangely enough, we don't However, the approach taken is to link the difficulty in predicting the next bit of output with what is believed to be a cryptographically hard problem[10] Therefore, a PRNG for which some advantage in predicting its output can be transformed into some advantage in solving an intractable problem is called a *cryptographically secure pseudo-random number generator* (CS-PRNG)

Blum-Blum-Shub PRNG

The Blum-Blum-Shub (BBS) PRNG [24] is one of the most famous CS-PRNG's It links the intractability of integer factorisation with the ability to determine the next output bit of the pseudo random sequence See Algorithm 1 3

1.7 Prime Number Generation

Prime number generation is needed for almost all public key encryption systems In the RSA encryption scheme the modulus is composed of two large primes

[10] This is generally how cryptographic protocols are proven secure, for more details, see Ch 4

Algorithm 1 3 Blum-Blum-Shub CSPRNG

INPUT Two large primes p and q, each congruent to $3 \mod 4$, l the number of random bits required, and a random seed s in the range $\{1 \quad N-1\}$ such that $\gcd(s, N) = 1$, where $N = pq$

OUTPUT A pseudo random number in the range $\{0 \quad 2^l - 1\}$

$r \leftarrow 0$
$x \leftarrow s^2 \mod N$
for $(i \leftarrow 0, i < l, i \leftarrow i + 1)$ **do**
  $z \leftarrow x \mod 2$
  $r \leftarrow 2r + z$
  $x \leftarrow x^2 \mod N$
end for
return r

$$N = p \quad q \tag{1 51}$$

In the generalised El Gamal public key encryption scheme we need to find a large prime modulus

There are algorithms that will produce a number that is provably prime There are also *probabilistic* algorithms that will tell us if a candidate number is probably prime However, these algorithms have a small probability of producing a 'false positive" That is, they may indicate that a composite number is prime With repeated independent tests we can reduce e, the error level, to one that is deemed acceptable

$$e = \varepsilon^n \tag{1 52}$$

Where ε is the probability of an error in one invocation of the primality test, and n is the number of invocations of the test

The strategy in industrial cryptography is to generate a random large number, and then check if it is prime This leads then to the obvious question, if we just generate a random large number, what are the chances that it is prime? Will it take days of trial and error before we happen upon a prime number? How many primes are there anyway? The approximate number of primes, less than any number x is given by

$$\text{number of primes less than } x \approx \frac{x}{\ln x} \tag{1 53}$$

Luckily there are infinitely many prime numbers [**125**, Ch 8][**91**, Ch 4] These primes are also randomly distributed, If x is a candidate number chosen at random, the probability that it is prime is given by

$$Pr[x \text{ is prime}] \simeq \frac{1}{\ln x} \tag{1 54}$$

where $Pr[\,]$ is used to denoted the probability of the event

To give some kind of perspective, this means that if we have a 512 bit candidate number the chance that it is prime approximately $1/177$ So, provided we have an efficient means of testing primality, obtaining a random large prime is not a particularly difficult task

1 7 1 Miller-Rabin Primality Test

First we look at Fermat's test This in itself is a useful primality test Though not used in practice, it is ideal for some definitions

Theorem 1 7 1 ***Fermat's Little Theorem*** *Suppose that p is prime, and $\alpha \in \mathbb{Z}_p^*$, then*

$$\alpha^p = \alpha \mod p \tag{1 55}$$

It follows from Fermat's little theorem that for any candidate number n,

$$\alpha^{(n-1)} = 1 \mod n \tag{1 56}$$

will hold if n is prime, whereas it is *unlikely* to hold if n is not prime

If equation 1 56 does *not* hold then we know that the number is definitely composite However, if we have a number for which the above equation holds then there is still a chance that the number is a composite We call such a number a Fermat pseudo prime to the base α However, if n is a composite then it can be shown that

$$Pr[\alpha^{(n-1)} \neq 1 \mod n] > 1/2 \tag{1 57}$$

This test can be repeated k times, each time with a different α A number that passes k repetitions of the tests is composite with probability at most $1/2^k$ If a number is detected as composite, α is called a *Fermat witness* to the compositeness of n

However there are a certain class of composite numbers for which the Fermat test will report that n is prime for any α co-prime to n They are the so-called *Carmichael numbers* [38, 93] They are much rarer than the primes, however they are still too common to allow the use of the Fermat primality test for industrial cryptography Instead we use the Miller-Rabin primality test

The Miller-Rabin primality test [106] is given in Algorithm **1 4** This test has a $1/4$ chance of wrongly certifying that a composite number is a prime Again, however, this error rate can be reduced to any figure by repeated application of the test The error rate is given as $1/4^k$ where k is the number of applications of the test Algorithm **1 4** repeats the test k times where k is given as an input

Algorithm 1 4 Miller-Rabin Primality Test
INPUT Odd integer n, and error bound k
OUTPUT If n is prime, with maximum error $1/4^k$

```
Write n − 1 as 2^s m, with m odd
for (j = 0, j < k, j = j + 1) do
  a ∈_R {2,    , n − 2}
  b ← a^m  mod n
  if b ≠ 1 and b ≠ (n − 1) then
    i ← 1
    while i < s and b ≠ (n − 1) do
      b ← b^2  mod n
      if ((b = 1)) then
        return false
      end if
      i = i + 1
    end while
    if (b ≠ (n − 1)) then
      return false
    end if
  end if
end for
return true
```

1 8 Discrete Logarithm Problem

The discrete logarithm is the inverse of discrete exponentiation in a finite cyclic group This was introduced in Sec 1 4 Given a cyclic group $\mathcal{G}$ of order n, the group operation $*$ and a generator g, we saw earlier that any element of $\mathcal{G}$ can be calculated as

$$y = g^x \tag{1 58}$$

where x, the discrete logarithm of y to the base g, is unique in the range $\{0 \quad n-1\}$ We denote that x is the discrete logarithm of y as follows $x = \log_g y$

Definition [91, Ch 3] The discrete logarithm problem (DLP) is the following Given a prime p, a generator g of $\mathbb{Z}_p^*$, and an element y in $\mathbb{Z}_p^*$, find the integer x, $0 \leq x \leq p-2$, such that $g^x = y \pmod p$

Definition [91 Ch 3] The generalised discrete logarithm problem (GDLP) is the following given a finite cyclic group $\mathcal{G}$ of order n, a generator g of $\mathcal{G}$, and an element $y \in \mathcal{G}$, find the integer $x, 0 \leq x \leq n-1$, such that $g^x = y$

The security of many cryptographic systems depends on the assumption that the discrete logarithm problem is intractable The most famous of these include the Diffie-Hellman key exchange, the Digital Signature Algorithm and the El Gamal encryption scheme

1 9 Encryption Schemes

Encryption schemes are used to keep confidential information that is to be transferred over an insecure channel There are two main families of encryption algorithms, *symmetric* or secret key encryption[11] and *asymmetric* or public key encryption[12]

In a symmetric encryption scheme the same key is used to encrypt and decrypt information There are two functions, E_k, which is used to represent the *encryption* function E with the secret key k and D_k, which represents the decryption function D with the secret key k E and D may or may not be the same function, but for a symmetric encryption algorithm the following relationship holds

$$m = D_k(E_k(m)) \tag{1 59}$$

where m is the data that is to be encrypted, and the same key k is used both for encryption and decryption

Obviously if this information is going to be transferred from one user to another (as opposed to encrypting information held, for example, on a hard disk), then both of these

[11] Examples include AES, DES, IDEA and TEA
[12] Examples include RSA and El Gamal

users must share the same secret key Symmetric encryption schemes suffer from two main problems

- **Key Distribution Problem** How to distribute encryption keys between users Depending on the importance of the secrets being transferred it may be feasible for the communicating parties to meet and agree on encryption keys However, this is a huge overhead It may be possible for all users to agree long term keys with one trusted party, who then acts as a go between to help clients agree session keys between themselves This is the basis of the popular Kerberos network authentication protocol [132] This method does not scale well

- **Key Management Problem** A new key is needed for each client with which you wish to communicate If the same key is used to communicate with two different recipients, they will be both be able to read messages that were meant for the other To securely communicate with n users, n different encryption keys will be needed

Asymmetric cryptography helps to resolve these problems In asymmetric cryptography encryption and decryption are carried out with two separate, but mathematically related keys - often called a *public key pair*, and consisting of a public and private key The public key is made public and the private key remains secret It is computationally infeasible to determine the private key knowing only the public key In this setting, encryption is carried out using the public key, and decryption is carried out using the private key We have the relationship

$$m = D_{k_{pri}}(E_{k_{pub}}(m)) \tag{1 60}$$

where E and D are encryption and decryption functions and k_{pub} and k_{pri} are related public and private keys respectively

A related idea is that of a **digital signature**[13] Using a digital signature you can sign

[13] Digital signatures will be explained in more detail in Ch 5

messages using the private key This signature can then be checked using the corresponding public key If σ is output when the private key k_{pri} is used to sign the message m, then

$$V_{k_{pub}}(\sigma, m) \tag{1 61}$$

will only output **true** on input of the same message m, signature σ and corresponding public key k_{pub}

1 10 El Gamal Encryption

The El Gamal encryption scheme [61] relies for its security on the assumption that the discrete logarithm problem is intractable The generalised El Gamal encryption scheme works over any finite cyclic group $\mathcal{G}$ where the following three conditions apply

- **Efficient** The group operation in $\mathcal{G}$ should be efficient
- **Secure** The discrete logarithm problem should be computationally infeasible
- **Practical** Elements in $\mathcal{G}$ can be reasonably compactly represented

The following are some of the groups over which El Gamal can be implemented

- The multiplicative group $\mathbb{Z}_p^*$, where p is prime
- The additive group of points on an elliptic curve over a finite field
- The multiplicative group $\mathbb{F}_q^*$, where q is a prime power, $q = p^m$ for some prime p

The El Gamal encryption scheme requires that each user perform the following setup algorithm to obtain a key pair

To encrypt a message to a user in the system the sender must first obtain an authentic copy of the recipient's public key To authenticate a public key the users agree on an entity that they all trust Such an entity is called a *trusted authority* This trusted authority then uses its private key to sign and thereby authenticate the public keys of all other users in the

Algorithm 1 5 ElGamal Public Key Pair Generation Algorithm

INPUT A finite cyclic group $\mathcal{G}$ of order n, and g, a generator of $\mathcal{G}$
OUTPUT An ElGamal public key pair (k_{pub}, k_{pri})

Generate a random integer $x \in_R \{1, \ldots, n-1\}$
$k_{pri} \leftarrow x$
$k_{pub} \leftarrow g^x$
return (k_{pri}, k_{pub})

system The client's public key, information about the client and the trusted authority's signature, together with optional additional information is called a public key *certificate* In this way the trusted authority binds the public key to the owner and the key distribution problem that we had with symmetric cryptosystems earlier is overcome[14]

Once a certified public key for the recipient has been obtained the sender now performs encryption as shown in Algorithm 1 6

Algorithm 1 6 ElGamal Public Key Encryption Algorithm

INPUT $\mathcal{G}$, n, g and k_{pub} as output from algorithm 1 5, and m, the message to be encrypted
OUTPUT An ElGamal ciphertext

Represent m as an element of the group $\mathcal{G}$
Generate a random integer $\alpha \in_R \{1, \ldots, n-1\}$
$R = g^\alpha$
$C = m \cdot k_{pub}^\alpha$
return (R,C)

To recover the plaintext message, the recipient, being the only person who knows the private key corresponding to the public key that was used by the sender, can carry out Algorithm 1 7

It is possible for each user in the system to use the same group $\mathcal{G}$ and generator g Now, since these values are common, and do not have to be distributed as part of the public key, the user's public key simply becomes $y = g^x$ The public key is distributed in an authenticated manner, whereas the private key is a secret known only the user who owns

[14]For all users except the trusted authority

Algorithm 1 7 ElGamal Public Key Decryption Algorithm

INPUT $\mathcal{G}$, and k_{pri} as output from algorithm 1 5, and (R, C), the output of algorithm 1 6

OUTPUT A plaintext message m

$\gamma = R^{k_{pri}}$
$m = \gamma^{-1} \cdot C$
return (m)

the key pair[15]

[15] A user is said to own a key pair if they know the corresponding private key

Chapter 2

Elliptic Curve Arithmetic

2.1 Long Form Weierstraß Equation

This chapter contains many well known standard number theoretic results. General references for this chapter include [21, 22, 137] and [62] for the number theoretic material, [62, 72, 90] and [108] and also [125, Ch.2] for the implementational details.

Definition An *Elliptic Curve* E over a field $\mathbb{F}_{p^k}$ (denoted either $E/\mathbb{F}_{p^k}$ or $E(\mathbb{F}_{p^k})$) is defined by the equation

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

where a_1, a_2, a_3, a_4 and $a_6 \in \mathbb{F}_{p^k}$. This is known as the *long form*, or *generalised* Weierstraß equation.

We must also check that the discriminant $\Delta \neq 0$, where Δ, the discriminant, is defined as follows:

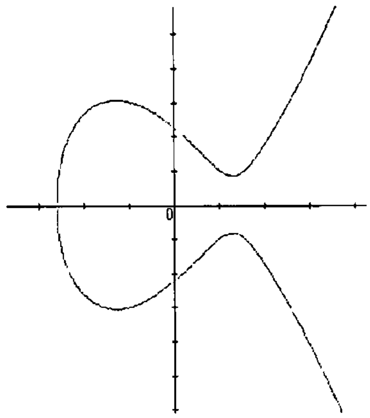


Figure 2 1 An elliptic curve

$$d_2 = a_1^2 + 4a_2, \tag{2 2}$$

$$d_4 = 2a_4 + a_1 a_3, \tag{2 3}$$

$$d_6 = a_3^2 + 4a_6, \tag{2 4}$$

$$d_8 = a_1^2 a_6 + 4a_2 a_6 - a_1 a_3 a_4 + a_2 a_3^2 - a_4^2, \tag{2 5}$$

$$\Delta = -d_2^2 d_8 - 8d_4^3 - 27d_6^2 + 9d_2 d_4 d_6 \tag{2 6}$$

If we want to consider the points in some extension field L of $\mathbb{F}_{p^k}$, $L \supseteq \mathbb{F}_{p^k}$, then the set of *L-rational* points on E is given as

$$E(L) = \{(x, y) \in L \times L \quad y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6\} \cup \mathcal{O} \tag{2 7}$$

- The condition $\Delta \neq 0$ is required to ensure that the curve is *smooth* That means that there is no point on the curve that has two or more distinct tangent lines
- The point $\mathcal{O}$ is called the point at infinity, and exists in all extension fields

For clarification, Fig 2 1 shows an elliptic curve defined over $\mathbb{R}$, the reals

Two elliptic curves E_1 and E_2 defined over $\mathbb{F}_{p^k}$ are said to be isomorphic over $\mathbb{F}_{p^k}$ if there exists $u, r, s, t \in \mathbb{F}_{p^k}, u \neq 0$, such that the change of variables

$$(x, y) \rightarrow (u^2 r + r, u^3 y + u^2 sx + t) \tag{2.8}$$

changes E_1 into E_2. This transformation is called an *admissible change of variables*. The point at infinity $\mathcal{O}$ remains unchanged.

2.1.1 Short Form Weierstraß Equations

This change of variables can be used to simplify the above Weierstraß equation. These changes of variables differ depending on whether the underlying field $\mathbb{F}_{p^k}$ has characteristic 2, 3 or $p > 3$ (sometimes called the *large prime* case).

If **char** $\mathbb{F}_{p^k} = 2$, then there are two possible cases to consider. If $a \neq 0$ then the admissible change of variables is

$$(x, y) \rightarrow \left(a_1^2 x + \frac{a_3}{a_1}, a_1^3 y + \frac{a_1^2 a_4 + a_3^2}{a_1^3}\right) \tag{2.9}$$

which transforms

$$\mathbb{E} \quad y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6 \tag{2.10}$$

into

$$y^2 + xy = x^3 + ax^2 + b \tag{2.11}$$

Such a curve is called *non-supersingular* and has discriminant $\Delta = b$.

If $a = 0$ then the admissible change of variables is

$$(x, y) \rightarrow (x + a_2, y) \tag{2.12}$$

which transforms E into

$$y^2 + cy = x^3 + ax + b \tag{2 13}$$

Such a curve is called *supersingular* and has discriminant $\Delta = c^4$

If **char** $\mathbb{F}_{p^k} = 3$, again there are two possible cases to consider If $a_1^2 \neq -a_2$ then the admissible change of variables is

$$(x, y) \to \left(x + \frac{a_4 - a_1 a_3}{a_1^2 + a_2}, y + a_1 x + a_1 \frac{a_4 - a_1 a_3}{a_1^2 + a_2} + a_3\right) \tag{2 14}$$

which again transforms E into

$$y^2 = x^3 + ax^2 + b \tag{2 15}$$

where $a, b \in \mathbb{F}_{p^k}$ Such a curve is said to be non-supersingular and has discriminant $\Delta = -a^3 b$

If $a_1^2 = -a_2$ then the admissible change of variables is

$$(x, y) \to (x, y + a_1 x + a_3) \tag{2 16}$$

which transforms E into

$$y^2 = x^3 + ax + b \tag{2 17}$$

Such a curve is called supersingular and has discriminant $\Delta = -a^3$

If **char** $\mathbb{F}_{p^k} \neq 2, 3$ then the following admissible change of variables

$$(x, y) \to \left(\frac{x - 3a_1^2 - 12a_2}{36}, \frac{y - 3a_1 x}{216} - \frac{a_1^3 + 4a_1 a_2 - 12a_3}{24}\right) \tag{2 18}$$

transforms E into

$$y^2 = x^3 + ax + b \tag{2 19}$$

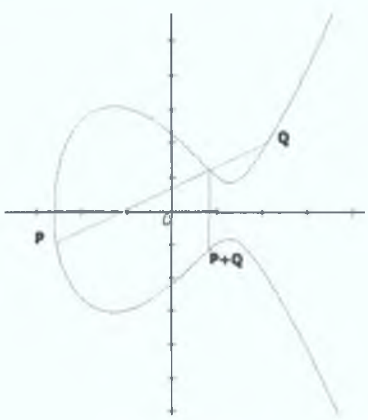
The discriminant Δ of this equation is given as

$$\Delta = -16(4a^3 + 27b^2). \tag{2.20}$$

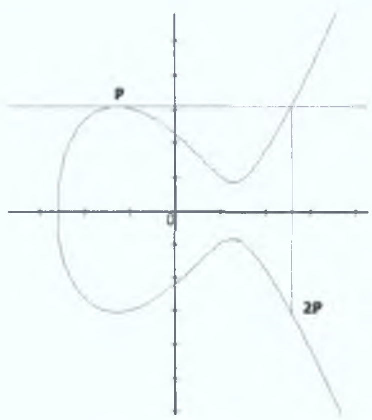
These shortened forms of the generalised (long form) Weierstraß equation are called *simplified* or *short form* Weierstraß equations. For the remainder of this dissertation we will, except where otherwise stated, use the short form Weierstraß notation. We will also assume that the curves are defined over a field $\mathbb{F}_{p^k}$ such that **char** $\mathbb{F}_{p^k} \neq 2, 3$.

2.2 Group Law Over Elliptic Curves

We now show how a finite group (see Sec. 1.4) can be instantiated over an elliptic curve. If E is a curve defined over the field $\mathbb{F}_{p^k}$ there is a binary group operation called *elliptic curve point addition* which operates on two points on the curve to give a third point on the curve. This is given by the *chord-and-tangent rule* [72, 125, 90]. Together with this addition rule the set of points on the curve (including $\mathcal{O}$) form an abelian group, with $\mathcal{O}$ serving as the identity element. The following two images show the two step chord-and-tangent process.



point addition



point doubling

Table 2.1: Point Addition and Point Doubling.

The chord and tangent rule also defines addition of a point to itself. This operation is known as *point doubling*. This is similar to point addition, but instead of calculating the

Algorithm 2 1 General Point Addition Algorithm for Elliptic Curves
INPUT An Elliptic Curve E defined over field $\mathbb{F}_{p^k}$, two distinct points $P, Q \in E$ $P \neq -Q$ and $P, Q \neq \mathcal{O}$
OUTPUT $R = (P + Q)$

Calculate l, the line that intersects P and Q
Calculate where l intersects E again *E being a cubic equation, this will always happen*
Call this point $-R$
Calculate where v, the vertical line that intersects $-R$ intersects E again
Call this point R
return R

line that intersects two points we calculate the tangent line to E that intersects E at the point P This line will intersect E at one more point, which we call $-2P$ Reflect the point $-2P$ in the x-axis to obtain the point $2P = (P + P)$

Algorithm 2 2 General Point Doubling Algorithm for Elliptic Curves
INPUT An Elliptic Curve E defined over field $\mathbb{F}_{p^k}$, and a point $P \in E$ $P \neq \mathcal{O}$
OUTPUT $2P = (P + P)$

Calculate t, the line that is a tangent to E, at P Assume t is not vertical
Calculate where t intersects E again *E being a cubic equation, this will always happen*
Call this point $-2P$
Calculate where v, the vertical line that intersects $-2P$ intersects E again
Call this point $2P$
return $2P$

2 2 1 Point Addition for $E/\mathbb{F}_{p^k}$ where char $\mathbb{F}_{p^k} \neq 2, 3$

The addition of two points P, Q where $P, Q \neq \mathcal{O}$ and $P \neq -Q$

Let $P = (x_1, y_1)$, $Q = (x_2, y_2)$ and $R = (x_3, y_3) = (P + Q)$

$$\lambda = \left(\frac{y_2 - y_1}{x_2 - x_1}\right), \qquad (2\ 21)$$

$$x_3 = \lambda^2 - x_1 - x_2, \qquad (2\ 22)$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \qquad (2\ 23)$$

2 2 2 Point Doubling for $E/\mathbb{F}_{p^k}$ where char $\mathbb{F}_{p^k} \neq 2, 3$

The doubling of a point P where $P \neq \mathcal{O}$

Let $P = (x_1, y_1)$, $2P = (x_2, y_2)$

$$\lambda = \left(\frac{3x_1^2 + a}{2y_1}\right), \qquad (2\ 24)$$

$$x_2 = \lambda^2 - 2x_1, \qquad (2\ 25)$$

$$y_2 = \lambda(x_1 - x_2) - y_1 \qquad (2\ 26)$$

Java source code for point addition and point doubling over **char** $\mathbb{F}_{p^k} \neq 2, 3$ is included in Appendix B, and in the accompanying CD-ROM

The addition operation, along with the set of points on an elliptic curve give us a group over which to implement cryptographic systems So far we have not dealt with $\mathcal{O}$, the point at infinity The point at infinity serves as the identity element of the group We specify special rules for point addition which include $\mathcal{O}$

2 2 3 $\mathcal{O}$, The Point at Infinity

To define the point at infinity, we must first define what is meant by the negative of a point

The negative of a point

The negative of a point is simply the reflection of the point in the x-axis An elliptic curve is symmetric about the x-axis For this reason the negative of a point $P = (x, y)$ will be the point $(x, -y)$ and is denoted $-P$ Point subtraction is carried out as the addition of the negative of a point

We now look at what happens if we are to perform the addition rule between a point and its negative Since the negative of a point is the reflection of that point in the x-axis, the line l between a point and its negative will be a vertical line A vertical line that passes though an elliptic curve (which is not a tangent line) does *not* intersect the curve three times[1], as any non-vertical line would It intersects the curve only twice, once at the point P and the again at the negative of that point $(-P)$ We say that this line also cuts the curve again at the point at infinity $\mathcal{O}$

If we go back to our group theory we see that this final definition allows us to complete the definition of a group This group is instantiated over the elliptic curve E

Special cases for point addition

- **Addition of a point to its negative**

$$\forall P \text{ on } E \quad (P) + (-P) = \mathcal{O} \tag{2 27}$$

- **Addition of a point to the point at infinity**

$$\forall P \text{ on } E \quad (P) + (\mathcal{O}) = P \tag{2 28}$$

The group $(G, +)$ instantiated over an elliptic curve $E/\mathbb{F}_{p^k}$ has the properties of a group

- **Commutative** $\forall P_1, P_2$ on E $P_1 + P_2 = P_2 + P_1$ This can be easily seen, since the line that intersects P_1 and P_2 is the same line that intersects P_2 and P_1
- **Existence of an Identity Element** As mentioned above, $\mathcal{O}$ is defined as the identity element, and has the properties expected of an identity element
- **Existence of Inverse Elements** As above, the negative of a point P is denoted $-P$
- **Associativity** $\forall P_1, P_2, P_3$ on E $(P_1 + P_2) + P_3 = P_1 + (P_2 + P_3)$ The proof of associativity is quite complex, see [137] for more details

[1] As it would in point addition between two distinct points, where one point is not the negative of the other

2.3 Group Order

Let $E/\mathbb{F}_p$ be a curve E defined over a field $\mathbb{F}_p$ [72] Then the number of points with coordinates in $\mathbb{F}_p$ is denoted as $\#E/\mathbb{F}_p$ or $\#E(\mathbb{F}_p)$ This is called the *order* of the curve $E(\mathbb{F}_p)$ Since E is defined over $\mathbb{F}_p$, and is symmetric about the x-axis then an upper bound for the number of points is given by $2p+1$ (2 for each value of x and remembering $\mathcal{O}$) *Hasse* gives us a tighter upper and lower bounds on the number of points $\#E(\mathbb{F}_p)$

Theorem 2 3 1 *(Hasse) If E is an elliptic curve defined over $\mathbb{F}_p$, then*

$$p+1-2\sqrt{p} \leq \#E(\mathbb{F}_p) \leq p+1+2\sqrt{p} \tag{2 29}$$

Since $\sqrt{p}$ is *relatively* small compared to p we know that $\#E(\mathbb{F}_p) \approx p$

2 3 1 The Trace of Frobenius, t

t, the *trace of Frobenius* is defined [125] as

$$t = p+1-\#E(\mathbb{F}_p) \tag{2 30}$$

This gives us, when combined with equation 2 29 above

$$|t| \leq 2\sqrt{p} \tag{2 31}$$

The trace of Frobenius (which we will simply call the *trace* from now on) can be used to tell us whether a particular curve has cryptographic weaknesses or not

- The curve $E(\mathbb{F}_p)$ is said to be *anomalous* if its trace is 1 This means, together with equation 2 30, that the order of the curve is equal to p

- The curve $E(\mathbb{F}_p)$ is said to be *supersingular* if the characteristic p divides the trace t Since $|t| \leq 2\sqrt{p}$, this means that $t=0$ and the order of such curves is $p+1$ Such curves are considered weak in cryptography, and for discrete logarithm based cryptosystems are usually avoided However, these curves are popular in pairing based cryptosystems as it is only practical to operate on elements of $\mathbb{F}_{p^k}$ when k is small[2] If p does not divide the trace then the curve is non-supersingular Much work has been done on the use of non-supersingular curves in pairing based cryptosystems [15, 16, 95]

2 3 2 The Curve Embedding Degree, k

Consider an arbitrary elliptic curve defined over the field $\mathbb{F}_p$ This curve contains points P of prime order r, meaning that $rP = \mathcal{O}$, and r is the smallest positive integer for which $rP = \mathcal{O}$ The order of a point divides the curve order ($r \mid \#E(\mathbb{F}_p)$)

This same curve can be defined over an extension field $\mathbb{F}_{p^k}$ For a certain value of k the group of points on the curve become interesting This is the lowest degree extension field which includes the r^{th} roots of unity This value of k is called the *embedding degree* This is also referred to as the *security multiplier* The embedding degree k is defined by the equations

$$r \mid p^k - 1 \qquad (2\ 32)$$

and

$$r \nmid p^s - 1 \ \forall\ 0 < s < k \qquad (2\ 33)$$

The r^{th} roots of unity also form a cyclic group of order r

2 4 Discrete Logarithm Problem over Elliptic Curves

Elliptic curves can be generated such that $E(\mathbb{F}_p)$ contains a unique group of points of large prime order r This group of points is denoted $E(\mathbb{F}_p)[r]$ Formally we have

[2] Values in this field are used in the calculation of the Weil and Tate pairings, which are the only known implementations of bilinear maps We will examine this in more detail in Ch 3

$$E(\mathbb{F}_p)[r] = \{P \in \mathbb{F}_p \ \ rP = \mathcal{O} \text{ and } \forall\ 0 < i < r, iP \neq \mathcal{O}\} \cup \mathcal{O} \tag{2 34}$$

This group of points can be used to instantiate a class of public key cryptosystems which are based on the difficulty of the discrete logarithm problem (DLP) These are loosely referred to as El Gamal type cryptosystems[3] The difficulty of the discrete logarithm problem depends heavily on the group of elements $\mathcal{G}$ over which the problem is set Obviously using the set $\mathbb{Z}_N$, some group element α, and the addition operation the discrete logarithm problem is trivial, given $y = x \ \ \alpha$, x is given as $x = y/\alpha$ The DLP over elliptic curves (EC-DLP), which uses as its set $\mathcal{G}$ the points of large prime order r on a elliptic curve defined over a finite field $E(\mathbb{F}_p)$ is assumed to be intractable Therefore, provided r is a large enough prime, this provides a suitable group over which to construct cryptographic systems

As mentioned in the previous section, some elliptic curves are weaker than others, for example supersingular curves and non-supersingular curves with small embedding degree (k) Ironically, these curves are of particular interest in pairing based cryptography One of the first uses of pairings was to attack this group of curves, the attack was proposed by Menezes, Okamoto and Vanstone, the *MOV* attack We will look at this in more detail in section 4 2 5

2.5 Efficient Point Scalar Multiplication

When we looked at the discrete logarithm based problems (Sec 1 8), we required that we had a finite group $\mathcal{G}$ over which the discrete logarithm problem was intractable, a generator g of $\mathcal{G}$, and a random integer in the range $0 \leq x \leq r-1$, where $r = \#\mathcal{G}$ This satisfied the condition for **security** Also, for **practicality** we had the condition that the group operation $+$ must be efficiently computable The group operation over the points on an elliptic curve is addition Obviously for the DLP to be computationally infeasible r must

[3] See Sec 1 10, for a more detailed description of the El Gamal encryption system

be large x being uniformly distributed in the range $0 \leq x \leq r - 1$, will be *on average* $\approx r/2$ The naive approach to point scalar multiplication would be to repeatedly perform addition the required number of times This would require $(r - 1)$ additions which would not be practical

The process of computing $y = x\ g$, where y and g are points on an elliptic curve is known as *elliptic curve point scalar multiplication* It is also sometimes called *point exponentiation* as it is seen as the elliptic curve analogue of exponentiation over finite fields

We look at two real world methods used to speed up elliptic curve point scalar multiplication These are the relatively simple to understand *double-and-add* method and the *NAF window* method Java code examples are included in the accompanying CD-ROM

2 5 1 Double-and-Add Method for Point Scalar Multiplication

The double-and-add method works for any group where the operation is written additively We will just give the generic case here Consider the multiplication of a group element by 5 This can be performed in several equally valid ways For example, we could compute

$$5\ x = x + x + x + x + x \tag{2 35}$$

An equally valid way would be

$$5\ x = (x + x) + (x + x) + x \tag{2 36}$$

At this stage there probably looks like there isn't any difference in the two representations However, equation 2 36 actually requires one less addition operation than equation 2 35 This can be seen more clearly if it is written as

$$y = x + x, \tag{2 37}$$

$$5\ x = (y) + (y) + x \tag{2 38}$$

Now, it is obvious only 3 addition operation were needed as opposed to the 4 that were required in equation 2.35. If we expand out equation 2.37 again we see that it can be written as

$$5 \cdot x = 2(2x) + x \tag{2.39}$$

This is known as the double-and-add method for fast scalar multiplication. Equation 2.39 is particularly nice as it has a recursive formula.

Take a slightly larger scalar, say 20. Written in its binary representation, 10100, we see that if the least significant bit (rightmost bit) is a one then we double and add, if it is a zero, we just double. Using this small example we have

$$\begin{array}{lll}
z = 0, & e = 20 & \\
bit_0(e) = 0 & y = x & z = z = 0 \\
bit_1(e) = 0 & y = 2x & z = z = 0 \\
bit_2(e) = 1 & y = 2^2x & z = z + y = 0 + 4x = 4x \\
bit_3(e) = 0 & y = 2^3x & z = z = 4x \\
bit_4(e) = 1 & y = 2^4x & z = z + y = 4x + 16x = 20x
\end{array}$$

Each successive doubling takes one addition operation. Each "add" takes one addition operation. Therefore we have cut the number of group operations required from 19 to 6. Obviously an addition operation will be required if $LSB_e = 1$, where e is the multiplier therefore we have cut the number of group operations required from $(e-1)$ to $\approx 1.5x$ where $x = \lceil(\lg(e+1))\rceil$ is the length of the binary representation of e and e is a random number[4].

The Double and Add algorithm is given in Algorithm 2.3.

[4]With approximately half of the digits being one and the other half being zero.

Algorithm 2 3 Double and Add Algorithm for Elliptic Curve Point Scalar Multiplication

INPUT An Elliptic Curve E defined over field $\mathbb{F}_{p^k}$, a point $P \in E(\mathbb{F}_{p^k})[r]$ $P \neq \mathcal{O}$ and exponent x $0 \leq x < r$

OUTPUT $R = xP$

```
let {x_t ... x_0} represent the binary expansion of x
Q ← O
for (i = t, i ≤ 0, i ← i − 1) do
  Q ← Q + Q
  if (x_i = 1) then
    Q ← Q + P
  end if
end for
return Q
```

2 5 2 NAF Window Method for Point Scalar Multiplication

NAF Non-Adjacent Form

As we can see from the above calculation the number of operations that we carry out is dependent on the number of non-zero digits in the binary representation of the exponent Every time we encounter a 0 digit we must do one addition operation for the "double" Every time we encounter a 1 digit in the binary representation of the exponent we must do two addition operations, one for the 'add" and one for the 'double" In order to make this operation more efficient, we must reduce the number of 1 digits - however, we cannot change the exponent If the exponent in discrete logarithm based cryptosystems is chosen in a way such that it is not random this would seriously damage the security of the system See the discussion on random numbers in Sec 1 6 However, using elliptic curves with **char** $\neq 2, 3$ we have that if $P = (x, y)$, then $-P = (x, -y)$ This conversion is extremely efficient and can be used with a *signed binary representation* of the exponent

Consider for example the number 31 in decimal Written in binary we have

$$31_{10} = 11111_2 \tag{2 40}$$

This can also be written in signed binary representation, where the digits $0, \pm 1$ are allowed Conventionally, -1 is written as $\bar{1}$ 31_{10} can be written as

$$31_{10} = (32 - 1)_{10} = (100000 - 1)_2 = (1, 0, 0, 0, 0, \bar{1}) \tag{2 41}$$

Therefore, if we are using 31 as an exponent[5] the number of addition operations using the double-and-add method and the conventional binary representation would be 10 Using this new signed binary representation the number of addition operations would be 8

Formally, the NAF of a positive integer is defined as

Definition [72, Ch 3] A non-adjacent form (NAF) of a positive integer k is an expression $k = \sum_{i=0}^{l-1} k_i 2^i$ where $k_i \in \{0, \pm 1\}$, $k_{l-1} \neq 0$ and no two consecutive digits k_i are nonzero The length of the NAF is l

If k is a positive integer, then a few properties of NAF(k)[72 Ch 3] are

- For each k, NAF(k) is unique
- Importantly, NAF(k) has the fewest nonzero digits of any signed binary representation of k
- If the binary representation of k has length l, the length of NAF(k) will not exceed $(l + 1)$
- The average density of 1 digits in NAF(k) is $\approx 1/3$ for a random value k

The algorithm for working out the NAF representation of a number is given in Algorithm 2 4

This NAF representation can now be used with a modified version of the Double and Add algorithm given in Algorithm 2 3 Whereas in the conventional double and add algorithm only had "double" and 'add" operations to work with, we now have a subtraction operation that will be triggered by the -1 that we now have in the signed binary representation We can use the NAF representation obtained from Algorithm 2 4 in the following adapted

[5] This exponent, small, but with high hamming weight, is for clarity of exposition only

Algorithm 2 4 An Algorithm for Generating the NAF Representation of a Positive Integer k

INPUT A positive integer k
OUTPUT $\mathrm{NAF}(k) = \{k_{i-1}, \ldots, k_0\}$

$i \leftarrow 0$
while $(k \geq 1)$ **do**
  if $((k \mod 2) = 1)$ **then**
    $k_i \leftarrow 2 - (k \mod 4)$
    $k \leftarrow k - k_i$
  else
    $k_i \leftarrow 0$
  end if
  $k \leftarrow k/2$
  $i \leftarrow i + 1$
end while
return $\{k_{i-1}, \ldots, k_0\}$

double and add algorithm The NAF point scalar multiplication algorithm is given in Algorithm 2 5

Algorithm 2 5 An Algorithm for Elliptic Curves Point Scalar Multiplication based on NAF Representation

INPUT An Elliptic Curve E defined over field $\mathbb{F}_{p^k}$, a point $P \in E(\mathbb{F}_{p^k})[r]$ $(P \neq \mathcal{O})$, and exponent x $(0 \leq x < r)$
OUTPUT $Q = xP$

let $\{x_i \ldots x_0\}$ represent the NAF signed binary expansion of x
(for details see Algorithm 2 4)
$Q \leftarrow \mathcal{O}$
for $j = i$ **downto** 0 **do**
  $Q \leftarrow Q + Q$
  if $((x_j = 1))$ **then**
    $Q \leftarrow Q + P$
  end if
  if $((x_j = -1))$ **then**
    $Q \leftarrow Q - P$
  end if
end for
return Q

Following on from the NAF representation presented in the previous section we can produce a *width-w NAF* Whereas, previously we only had $0, \pm 1$ we now allow ourselves the integers in the range $-2^{w-1} \leq u < 2^{w-1}$ Using this new representation we can require that for any w consecutive digits, there is only one non-zero value

Definition [72, Ch 3] let $w > 2$ be a positive integer A *width-w NAF* of a positive integer k is an expression $k = \sum_{i=0}^{l-1} k_i (2^w)^i$ where each non-zero coefficient k_i is odd, $|k_i| < 2^{w-1}, k_{l-1} \neq 0$, and at most one of any w consecutive digits is non-zero

The w-NAF of a number is computed using Algorithm 2 6, which is closely related to Algorithm 2 4

Algorithm 2 6 An Algorithm for Generating the w-NAF Representation of a Positive Integer k

INPUT A positive integer k
OUTPUT w-NAF$(k) = \{k_{i-1}, \ldots, k_0\}$

```
i ← 0
while (k ≥ 1) do
  if ((k mod 2) = 1) then
    k_i ← k mod 2^w
    k ← k − k_i
  else
    k_i ← 0
  end if
  k ← k/2
  i ← i + 1
end while
return {k_{i−1}, ..., k_0}
```

2 6 Multiple Point Scalar Multiplication

We now look at efficient multiple point scalar multiplication This is used for example if we wish to calculate some point $R = xP + yQ$ The idea is to perform two or more point scalar multiplications simultaneously A precomputed table is calculated such as the one shown in Table 2 2

Algorithm 2 7 An Algorithm for Elliptic Curves Point Scalar Multiplication based on w-NAF Representation

INPUT An Elliptic Curve E defined over field $\mathbb{F}_{p^k}$, a point $P \in E(\mathbb{F}_{p^k})[r]$ $P \neq \mathcal{O}$ and exponent x $0 \leq x < r$

OUTPUT $Q = xP$

let $\{x_{l-1} \quad x_0\}$ represent the w-NAF expansion of x
(for details see algorithm 2 6)
$Q \leftarrow \mathcal{O}$
$P_i \leftarrow iP$ for $i \in \{1, 3, 5, \quad (2^{w-1} - 1)\}$
for i **from** $l - 1$ **downto** 0 **do**
  $Q \leftarrow Q + Q$
  if $(x_j \neq 0)$ **then**
    $Q \leftarrow Q + P_{x_j}$
  else if $(x_j < 0)$ **then**
    $Q \leftarrow Q - P_{-x_j}$
  end if
end for
return Q

Precomputation
$0P + 1Q$
$0P + 2Q$
$0P + (2^w - 1)Q$
$1P + 1Q$
$(2^w - 1)P + (2^w - 1)Q$

Table 2 2 Combined multi-point scalar multiplication

Using this set of values, together with the w-NAF representation, it is possible to adjust Algorithm 2 7 to compute any point of the form $R = xP + yQ$ The more points that are precomputed the more efficient the algorithm becomes, though the storage requirements become correspondingly larger

2 7 Point Compression

Because we know the equation of the curve, giving both co-ordinates is giving more than the minimum required information Given the x co-ordinate, the corresponding y co-ordinate must be one of two possible values The idea of representing a point as one co-ordinate plus additional identifying information about the second co-ordinate is called *point compression* [133]

Elliptic curves are mapped by an equation of the form $y^2 = x^3 + ax + b$ Any x co-ordinate of a point that is on the curve will be associated with two possible y co-ordinate values These values will be $\pm y$, since $y = \pm\sqrt{x^3 + ax + b}$ Therefore we must specify which of these two possible values is being referred to This requires one additional bit of information This is the bit $\bar{y} = LSB(y)$, and works, for curves defined over $\mathbb{F}_p$, since if y is even, then $-y$ will be odd[6] These points are the negatives of each other

The original supersingular curve specified by Boneh and Franklin for use with their identity based encryption scheme [31], is $y^2 = x^3 + 1 \mod p$ where $p \equiv 2 \mod 3$ This curve has the interesting property that for each y co-ordinate there is exactly one x co-ordinate Obviously, for each x co-ordinate there are two possible values for y However, this leads to an even more efficient compression based on the y point In this situation an additional bit does not have to be stored, because x can uniquely be recovered from the equation

$$x = \sqrt[3]{y^2 - 1} \tag{2 42}$$

2 8 Projective Space

As we have seen all elliptic curve public key cryptosystems rely on the basic group operation – point addition We have looked at faster ways of computing point scalar multiplication, but

[6] $-y = (p - y) \mod p$

these techniques are built upon point addition and point doubling Obviously if we can make these operations faster then we can improve the performance of the overall cryptosystem

We have also seen that we can have several different representations for the same point For example a point defined over the field $\mathbb{F}_p$ can be represented as $P = (x, y)$, where x and y are both integers in $\mathbb{F}_p$ Alternatively if this point is to be transmitted, and we want to make a trade-off between computational and bandwidth considerations – decreasing bandwidth requirements at a cost of increasing computational requirements – then we can represent this point as $P = (x, \bar{y})$, where $\bar{y}$ represents the LSB of the y co-ordinate There is now no redundancy in this representation

There are two issues raised above – the complexity of the basic point addition operation and the ability to represent points in different formats There are representations for points which allow us to perform the group operation using a smaller than standard amount of computation – especially by eliminating the modular inversion operation There are several such co-ordinate systems They are two dimensional **Affine**, and the three dimensional **Standard Projective**, **Jacobian Projective** and **López-Dahab Projective** [72, 1, 126] co-ordinate systems

Now we have the same set of points represented in four different ways The first of these representations is defined over two dimensions, whereas the others are defined over three dimensions Obviously, however, if all variables can be in the range $\{0, \quad, (q-1)\}$ then the latter three co-ordinate systems allow us to represent q^3 elements whereas the affine co-ordinate system allow us only to represent q^2 elements

We can construct equivalence classes One can define an equivalence relationship over the set $\mathbb{F}_{p^k}^3 \backslash \{(0,0,0)\}$ as

$$(X_1, Y_1, Z_1) \equiv (X_2, Y_2, Z_2) \text{ if } X_1 = \lambda^c X_2, Y_1 = \lambda^d Y_2, Z_1 = \lambda Z_2 \text{ for some } \lambda \in \mathbb{F}_{p^k}^* \quad (2\ 43)$$

Using Jacobian projective co-ordinates we have, $c = 2, d = 3$ and the following

$$(X : Y : Z) = \{(\lambda X, \lambda Y, \lambda Z) : \lambda \in \mathbb{F}_{p^k}^*\} \tag{2.44}$$

$(X : Y : Z)$ is called a projective point, and $\{(\lambda^c X, \lambda^d Y, \lambda Z)\}$ is called a representative of this projective point. If $Z \neq 0$ then $(X/Z^c, Y/Z^d, 1)$ is a representative of the point $(X : Y : Z)$. Therefore, this gives us a one-to-one relationship between the set of projective points and the set of affine points.

$$\begin{aligned} \mathbb{P}(\mathbb{F}_{p^k})^* &= \{(X : Y : Z) : X, Y, Z \in K, Z \neq 0\}, && (2.45) \\ \mathbb{A}(\mathbb{F}_{p^k}) &= \{(x, y) : x, y \in \mathbb{F}_{p^k}\} && (2.46) \end{aligned}$$

Using standard projective co-ordinates we have the transformation $(X : Y : Z)$, $Z \neq 0$ corresponds to the affine point $(x, y) \leftarrow (X/Z^2, Y/Z^3)$. Now, given the curve equation

$$y^2 = x^3 + ax + b \tag{2.47}$$

we can substitute in these new values and get the corresponding curve equation using projective co-ordinates.

$$\begin{aligned} (Y/Z^3)^2 &= (X/Z^2)^3 + a(X/Z^2) + b(Z/Z), && (2.48) \\ Y^2/Z^6 &= X^3/Z^6 + a(X/Z^2) + b(Z/Z), && (2.49) \\ Y^2 &= X^3 + aXZ^4 + bZ^6 && (2.50) \end{aligned}$$

Using projective co-ordinates, $\mathcal{O}$ is represented as the projective point $(0, 1, 0)$.

Now that we have q possible representations for each point, we have the ability to define point addition operations that do not require an expensive modulo inversion. If we need to, we can convert first from affine to projective coordinates, then do the computationally expensive operations, and then convert back to affine co-ordinates This will require an inversion, but will still be much quicker than working solely in affine co-ordinates If we need to convert from affine co-ordinates to standard projective co-ordinates we simply set $Z = 1$, and so the transformation is simply $(x, y) \rightarrow (X, Y, 1)$ To convert back we simply do the transformation $(x, y) \leftarrow (X/Z^2, Y/Z^3)$ Using Montgomery's trick, this requires one modular inversion

2 9 Point Reduction

A technique related to point compression is called *point reduction* Some elliptic curve cryptosystems don't require that we specify whether we mean the positive or negative of a point Both points are treated equally Therefore it is possible to operate just using the x co-ordinate of a point This was first pointed out by Miller in [94] In some situations, we can discard the y co-ordinate, because there are formulas for calculating the x coordinate of some multiple of a point that depend only on the x co-ordinate of the original point

2 10 Group Structure

As described in Sec 2 3 2, the embedding degree extension field is the lowest degree extension field which includes the r^{th} roots of unity The r^{th} roots of unity form a cyclic group of order r These elements are used in pairing based cryptography To keep the representation of this group reasonably small and to allow fast computation in this group we deliberately pick curves that have a small embedding degree If we restrict ourselves to supersingular elliptic curves then we always have $k \leq 6$ [92] If we use non-supersingular curves we can find curves that have much higher embedding degrees **For the remainder of this thesis we will assume that k is small and even** A popular choice of curve for identity based cryptography are curves where the embedding degree k is 2 The *order* of this curve, denoted $\#E(\mathbb{F}_p)$ is $(p + 1 - t)$ where t is the *trace of Frobenius* The order of this curve over $\mathbb{F}_{p^2}$ is $(p + 1 - t)(p + 1 + t)$, (which in the general case can be calculated using Weil's theorem) The group of points defined over $\mathbb{F}_{p^2}$ do not form a cyclic group For a $k = 2$ curve r exactly divides both $p + 1$ and $(p + 1 - t)$, and hence $r|t$ And $r^2|\#E$ [117, 14, 15]

Let the complete set of points defined over $\mathbb{F}_{p^2}$ be called G, of order $\#E(\mathbb{F}_{p^2})$ The set of all points that are transformed to $\mathcal{O}$ by multiplication by r is denoted $G[r]$ These are the r-torsion points Since r is prime, these are all the points of order r plus $\mathcal{O}$ There are r^2 such points, and these r^2 points can be organised as $r + 1$ distinct cyclic subgroups of order r – they all share $\mathcal{O}$ Note that one of these subgroups is $S[r]$ and consists of all those r-torsion points from the original curve $E(\mathbb{F}_p)$ - points of the form $P[(a, 0), (c, 0)]$, which are defined on both the base and extension fields

Let $CoF = \#E(\mathbb{F}_{p^2})/r^2$ Then a random point on the curve can be mapped to a point in one of these sub-groups of order r by multiplying it by this co-factor CoF The set of distinct points generated by multiplying every element of G by r is called rG The number of elements in rG is CoF This is called a co-set [117]

Consider the partitioning of the $\#E$ points into distinct co-sets This can be done by adding a random point R to every element of rG There are exactly r^2 such distinct co-sets, each with CoF elements The original co-set rG is the unique co-set that contains $\mathcal{O}$ Every co-set contains exactly one r-torsion point Elements of these co-sets are not all of the same order They do not form a group

The quotient group G/rG is the group formed by all of these co-sets [117]

Chapter 3

Bilinear Maps

3.1 Divisor Theory

Let E be an elliptic curve defined over the field K For each point $P \in E(K)$ define a *formal symbol* $[P]$ A divisor [137, Ch 11][87] is a finite linear combination of such formal symbols with integer coefficients

$$D = \sum_j a_j [P_j], a_j \in \mathbb{Z} \qquad (3\ 1)$$

A divisor is therefore an element of the free abelian group generated by the symbols $[P]$ The group of divisors is denoted $\mathrm{Div}(E)$ The *degree* of a divisor is given by

$$\deg(D) = \sum_j a_j \in \mathbb{Z} \qquad (3\ 2)$$

and as shown above evaluates to an integer

The *sum* of a divisor is simply the sum of all of the points that are represented

$$\mathrm{sum}(D) = \sum_j a_j P_j \in E(K) \qquad (3\ 3)$$

The sum function uses the standard addition formula on the points that are represented by the formal symbols The *support* of a divisor is the set of all points represented by formal symbols for which $a_j \neq 0$ It is customary to only include formal symbols if they have non-zero coefficients

$$\mathrm{supp}(D) = \{[P_j] \in D \mid a_j \neq 0\} \tag{3 4}$$

3 1 1 Function on a Curve

We now define what is meant by a function on a curve Suppose that E is an elliptic curve, then f is a function on E if it is a rational function[1]

$$f(x, y) \in \overline{K}(x, y) \tag{3 5}$$

that is defined for at least one point in $E(\overline{K})$, where $\overline{K}$ is the algebraic closure of K This means that the function must intersect E at some point A function takes values in $\overline{K} \cup \{\infty\}$ The evaluation of a function f at a point P is denoted $f(P) = f(x_P, y_P)$

A function is said to have a *zero* at P if it takes on the value 0 at P [87, 137] A function is said to have a *pole* at P if it takes the value ∞ f only has finitely many zeros and poles For every point P for which the function f is defined there is a function u_P called a *uniformiser at* P where f can be expressed in terms of u_P as follows

$$f = u_P^r \quad g, \text{ where } r \in \mathbb{Z} \text{ and } g(P) \neq 0, \infty \tag{3 6}$$

A uniformiser u_P can be obtained as the equation of a line that passes though the point P which is not a tangent to E at P Now that we have this definition we can define what is meant by the *order* of a function at a point P

$$\mathrm{ord}_P(f) = r \tag{3 7}$$

If f is a function on E then $\mathrm{ord}_P(f)$ counts the multiplicity of f at P $\mathrm{ord}_P(f)$ is positive

[1] A rational function is formed when one polynomial divides another polynomial

when $f(P) = 0$ and negative when $f(P) = \infty$ If $\mathrm{ord}_P > 0, P$ is a zero, if $\mathrm{ord}_P < 0, P$ is a pole, if $\mathrm{ord}_P = 0, P$ is neither a zero or a pole A pole or zero of multiplicity one is called "simple", of multiplicity 2 is called 'double'

3 1 2 Principal Divisor

A *principal divisor* on E is a divisor of some function f which is defined over E [90], as shown in Equation 3 8 This is denoted as $D = \mathrm{div}(f)$

$$\mathrm{div}(f) = \sum_{P \in E} ord_P(f)[P] \tag{3 8}$$

A principal divisor D will have $\deg(D) = 0$ and $\mathrm{sum}(D) = \mathcal{O}$ We have now established a relationship between a divisor D and a function f on E

Suppose that P_1, P_2 and P_3 are three points on E that lie on the line defined by the function

$$f(x, y) = ax + by + c = 0 \tag{3 9}$$

Then, since $\deg(f) = 0$, and f has three zeros at P_1, P_2 and P_3 (since they are on the line) then it has a triple pole at $\mathcal{O}$ This can be written as

$$\mathrm{div}(f) = [P_1] + [P_2] + [P_3] - 3[\mathcal{O}] \tag{3 10}$$

We also know that P_3 is the point $-(P_1 + P_2)$, since the reflection of P_3, using the elliptic curve addition formula given in section 2 2 is the point $(P_1 + P_2)$ We know that the equation of the vertical line running though P_3 and $-P_3$ is given by equation $(x - x_3) = 0$, where $P_3 = (x_3, y_3)$ That is

$$\mathrm{div}(x - x_3) = [P_3] + [-P_3] - 2[\mathcal{O}] \tag{3 11}$$

Therefore

$$\operatorname{div}\left(\frac{ax+by+c}{x-x_3}\right) = \operatorname{div}(ax+by+c) - \operatorname{div}(x-x_3) = [P_1] + [P_2] - [-P_3] - [\mathcal{O}] \quad (3.12)$$

Since $P_1 + P_2 = -P_3$ on E, this may be written as

$$[P_1] + [P_2] = [P_1 + P_2] + [\mathcal{O}] + \operatorname{div}\left(\frac{ax+by+c}{x-x_3}\right) \quad (3.13)$$

In this way principal divisors may be expressed in terms of a formal sum and the divisor of a function. We can use this idea to incrementally build from a divisor D, a function f such that $\operatorname{div}(f) = D$, at each point replacing part of the formal sum by a more complex function. First, we check that the formal sum has sum equal $\mathcal{O}$ and deg equal to 0.

Consider for example the curve E defined over $\mathbb{F}_{11}$ given by

$$y^2 = x^3 + 4x \quad (3.14)$$

Let

$$D = [(0,0)] + [(2,4)] + [(4,5)] + [(6,3)] - 4[\mathcal{O}] \quad (3.15)$$

Then, with a bit of work, $\operatorname{sum}(D) = \mathcal{O}$[2], and $\deg(D) = 0$, therefore it follows that D is the divisor of a function. We wish to find this function. We use the approach taken above, where we incrementally resolve parts of the formal sum into divisors of functions and then combine these smaller divisors into a more complex divisor.

The line though $(0,0)$ and $(2,4)$ is $y - 2x = 0$. It is a tangent to E at $(2,4)$, so

$$\operatorname{div}(y-2x) = [(0,0)] + 2[(2,4)] - 3[\mathcal{O}] \quad (3.16)$$

The vertical line though $(2,4)$ is $x - 2 = 0$, therefore we have

[2] $[(2,4)] + [(4,5)] + [(6,3)]$ are all on the same line ($(2,4) = ((4,5) + (6,3))$) and $(0,0) = -2(2,4)$, so $(0,0) + (4,5) + (6,3) + (2,4) = -2(2,4) + 2(2,4) = \mathcal{O}$

$$\operatorname{div}(x-2) = [(2,4)] + [(2,7)] - 2[\mathcal{O}]^3 \tag{3.17}$$

And

$$\operatorname{div}\left(\frac{y-2x}{x-2}\right) = \operatorname{div}(y-2x) - \operatorname{div}(x-2), \tag{3.18}$$

$$\operatorname{div}\left(\frac{y-2x}{x-2}\right) = [(0,0)] + [(2,4)] - [(2,7)] - [\mathcal{O}] \tag{3.19}$$

Remember

$$D = [(0,0)] + [(2,4)] + [(4,5)] + [(6,3)] - 4[\mathcal{O}] \tag{3.20}$$

Therefore

$$D = [(2,7)] + \operatorname{div}\left(\frac{y-2x}{x-2}\right) + [(4,5)] + [(6,3)] - 3[\mathcal{O}] \tag{3.21}$$

We can also calculate the following function

$$[(4,5)] + [(6,3)] = [(2,4)] + [\mathcal{O}] + \operatorname{div}\left(\frac{y+x+2}{x-2}\right) \tag{3.22}$$

Using these two equations we can determine the equation of the function f for which $D = \operatorname{div}(f)$

$$D = [(2,7)] + [(2,4)] - 2[\mathcal{O}] + \operatorname{div}\left(\frac{y-2x}{x-2}\right) + \operatorname{div}\left(\frac{y+x+2}{x-2}\right) \tag{3.23}$$

$$D = \operatorname{div}(x-2) + \operatorname{div}\left(\frac{y-2x}{x-2}\right) + \operatorname{div}\left(\frac{y+x+2}{x-2}\right) \tag{3.24}$$

$$D = \operatorname{div}\left(\frac{(y-2x)(x+y+2)}{x-2}\right) \tag{3.25}$$

[3] $7 \equiv (-4) \mod 11$

3 2 Weil Pairing

The Weil pairing is a bilinear map which takes two points of order r in the embedding degree extension field, and maps to an element of $\mathbb{F}_{p^k}$ [137]

$$e \quad E(\mathbb{F}_{p^k})[r] \times E(\mathbb{F}_{p^k})[r] \to \mu_r \tag{3 26}$$

Here μ_r is the set of r^{th} roots of unity in $\mathbb{F}_{p^k}$

Let $T \in E[r]$ Then there exists a function f_T such that

$$\mathrm{div}(f_T) = r[T] - r[\mathcal{O}] \tag{3 27}$$

since $\mathrm{sum}(\mathrm{div}(f_T)) = \mathcal{O}$ and $\deg(\mathrm{div}(f_T)) = 0$

Let $T' \in E[r^2]$ be such that $rT' = T$ Then there also exists a function g_T such that

$$\mathrm{div}(g_T) = \sum_{R \in E[r]} ([T' + R] - [R]) \tag{3 28}$$

The sum of the points in the divisor is $\mathcal{O}$ This follows from the fact that there are r^2 points R in $E[r]$ The points R in $\sum[T' + R]$ and $\sum[R]$ cancel and therefore the sum is $\sum[T'] = \mathcal{O}$ The value of g_T does not depend on R

Let $f_T \circ r$ denote a function that starts with a point, multiplies it by r and then applies f_T The points $P = T' + R$ with $R \in E[r]$ are those points P with $rP = T$ It follows that

$$\mathrm{div}(f_T \circ r) = r\left(\sum_R [T' + R]\right) - r\left(\sum_R [R]\right) = \mathrm{div}(g_T^r) \tag{3 29}$$

Let $S \in E[r]$ and let $P \in E(K)$ Then

$$g_T(P+S)^r = f_T(r(P+S)) = f_T(rP) = g_T(P)^r \tag{3 30}$$

Therefore $g_T(P+S)/g_T(P) \in \mu_r$ and is independent of P

The *Weil Pairing* is defined as

$$e_r(S,T) = \frac{g_T(P+S)}{g_T(P)} \tag{3 31}$$

3 2 1 Bilinearity of the Weil Pairing

We now examine the bilinearity of the pairing [137, 22]

We first look at linearity in the first variable To recap, from the previous section, we have

$$e_r(S,T) = \frac{g_T(P+S)}{g_T(P)}, \tag{3 32}$$

expanding we have

$$e_r(S_1,T)e_r(S_2,T) = \frac{g_T(P+S_1)}{g_T(P)}\frac{g_T(P+S_2)}{g_T(P)} \tag{3 33}$$

But the result of the pairing is independent of the choice of P, so we can replace P in the second pairing, with the (rather convenient) value $P+S_1$ This gives

$$e_r(S_1,T)e_r(S_2,T) = \frac{g_T(P+S_1)}{g_T(P)}\frac{g_T(P+S_1+S_2)}{g_T(P+S_1)}, \tag{3 34}$$

which simplifies to

$$e_r(S_1,T)e_r(S_2,T) = \frac{g_T(P+S_1+S_2)}{g_T(P)}, \tag{3 35}$$

$$\text{i e } e_r(S_1,T)e_r(S_2,T) = e_r(S_1+S_2,T) \tag{3 36}$$

□

We next examine linearity in the second variable

Suppose we have three points T_1, T_2 and $T_3 \in E(r)$, such that $T_1 + T_2 = T_3$ Let $g_1 \quad g_3$ be the functions used to define $e_r(S,T_i)$ Let h be the function, such that

$$\mathrm{div}(h) = [T_3] - [T_1] + [\mathcal{O}] \tag{3 37}$$

We also know that if $T \in E[n]$, then

$$\mathrm{div}(f) = n[T] - n[\mathcal{O}] \tag{3 38}$$

for some function f and so for $i = 1, \quad ,3$ we have

$$\mathrm{div}(f_i) = n[T_i] - n[\mathcal{O}] \tag{3 39}$$

and so we can express h in terms of the f_i's

$$\mathrm{div}\left(\frac{f_3}{f_1 f_2}\right) = n\mathrm{div}(h) = \mathrm{div}(h^n) \tag{3 40}$$

This allows us to write

$$f_3 \equiv f_1 f_2 h^n \tag{3 41}$$

From equation 3 30 we have

$$f(nP) = g(P)^n \tag{3 42}$$

Combining the previous two results we get

$$f_3 \equiv f_1 f_2 h^n \text{ implies } g_3 \equiv g_1 g_2 (h \quad n), \tag{3 43}$$

which implies

$$e_r(S, T_1 + T_2) = \frac{g_3(P+S)}{g_3(P)} = \frac{g_1(P+S)}{g_1(P)} \frac{g_2(P+S)}{g_2(P)} \frac{h(n(P+S))}{h(nP)} \tag{3 44}$$

But, since $n(P+S) = \mathcal{O}$ the last term is equal to $1_{\mathbb{F}_{p^k}}$ this gives

$$e_r(S, T_1 + T_2) = \frac{g_3(P+S)}{g_3(P)} = \frac{g_1(P+S)}{g_1(P)}\frac{g_2(P+S)}{g_2(P)} = e_r(S, T_1)e_r(S, T_2) \tag{3 45}$$

as desired

3.3 Tate Pairing

There is another pairing called the Tate pairing, which is generally more efficient to compute It is a bilinear map of the form [22, 137]

$$e \quad E(\mathbb{F}_{p^k})[r] \times E(\mathbb{F}_{p^k})/rE(\mathbb{F}_{p^k}) \to \mathbb{F}_{p^k}/(\mathbb{F}_{p^k})^r \tag{3 46}$$

where $rE(\mathbb{F}_{p^k})$ is defined to be $rE(\mathbb{F}_{p^k}) = \{rP \quad P \in E(\mathbb{F}_{p^k})\}$

Let $P \in E[r]$ Since $rP = \mathcal{O}$, it follows that there is a function D_P such that $\mathrm{div}(D_P) = r(\mathrm{div}(P)$ - r $\mathrm{div}(\mathcal{O}))$ Let D_Q be any degree 0 divisor such that the support of D_Q is disjoint from the support of D_P Now, two divisors are said to be equivalent, denoted $D \sim D'$, if the difference between them is a principal divisor[4] Therefore if we have two functions f and f' such that $\mathrm{div}(f) = D$ and the $\mathrm{div}(f') = D'$ f, can be replaced by a function f' such that

$$\mathrm{div}(f'_P) = [\mathcal{O}] - [P] \tag{3 47}$$

Therefore exists a function f_P such that

$$\mathrm{div}(f_P) = rD_P \tag{3 48}$$

Let $D_Q = \sum_i a_i[Q_i]$ be a divisor of degree 0 such that $\mathrm{sum}(D_Q) = Q$ and such that D_P and D_Q have no points in common We can define the Tate pairing as

[4]To recap A principal divisor, which is a divisor of a function, is one such that $\deg(D) = 0$ and $\mathrm{sum}(D) = \mathcal{O}$

$$\langle P, Q\rangle = f_P(D_Q) \tag{3 49}$$

where, for any function f_P, whose divisor has no points in common with D_Q we define

$$f_P(D_Q) = \prod_i f_P(Q_i)^{a_i} \tag{3 50}$$

Assume that f_P is defined over $\mathbb{F}_{p^k}$, and let R be any point in $E(\mathbb{F}_{p^k})$ Let $D_Q = [Q+R]-[R] \in \mathbb{F}_{p^k}$, then the Tate pairing can be defined as

$$f_P(D_Q) = f_P(Q+R)/f_P(R) \tag{3 51}$$

3 3 1 Bilinearity of the Tate Pairing

We now look at linearity of the first variable From equation 3 51 we have the Tate pairing defined as $\langle P, Q\rangle_n = f_P(D_Q)$ As with proving the bilinearity of the Weil pairing we let $P_1, P_2 \in E(\mathbb{F}_p)[r], D_{P_1}$ and D_{P_2} be the respective divisors and f_{P_1} and f_{P_2} be the corresponding functions

Adding two divisors of points gives the divisor on the addition of the two points, therefore we have

$$D_{P_1} + D_{P_2} = D_{P_1+P_2} \equiv [P_1+P_2] - [\mathcal{O}] \tag{3 52}$$

For $i = 1, 2$, there exists functions f_{P_i} such that

$$\mathrm{div}(f_{P_i}) = rD_{P_i}, \tag{3 53}$$

and

$$\mathrm{div}(f_{P_1} f_{P_2}) = rD_{P_1+P_2} \tag{3 54}$$

Therefore

$$\langle P_1 + P_2, Q\rangle_r = f_{P_1} f_{P_2}(D_Q) = \langle P_1, Q\rangle_r \langle P_2, Q\rangle_r \tag{3 55}$$

Hence, the function is linear in the first variable

Looking at the second variable we have

Let $Q_3 = Q_1 + Q_2$ Let $D_{Q_i} = [Q_i] - [O]$ for $i = 1 \quad 3$

We know that

$$D_{Q_1} + D_{Q_2} = [Q_1 + Q_2] - [O] = D_{Q_3} = [Q_3] - [O] \tag{3 56}$$

Therefore we have

$$\langle P, Q_3\rangle_r = \langle P, Q_1 + Q_2\rangle_r = f(D_{Q_1} + D_{Q_2}) = f(D_{Q_1}) + f(D_{Q_2}) = \langle P, Q_1\rangle_r \langle P\ Q_2\rangle_r \tag{3 57}$$

Therefore we have linearity in the second variable

3 3 2 Reduced Tate Pairing

As we have established in the previous sections, the Weil pairing gives a definitive answer, whereas the Tate pairing equates to a set of equivalence classes The Weil pairing can be used directly for implementing a bilinear function for use with the cryptographic protocols to be described in later chapters However, as it is described above, the Tate pairing is not ideal for use in cryptography We would prefer if the pairing resulted in a definitive answer

To make the Tate pairing useful for cryptography we need a many-to-one mapping that will take all the members of an equivalence class and reduce them to the same result This can be achieved by a simple exponentiation [22]

$$t_r(P, Q) = \langle P, Q\rangle_r^{(p^k - 1)/r} \tag{3 58}$$

This is known as the reduced Tate pairing[5] and gives a definite result in the r^{th} roots of unity group, which we denote as μ_r. From now on, when we mention the Tate pairing it can be assumed that we are talking about the reduced Tate pairing.

3.4 Modified Pairings

The Weil and Tate pairings take two distinct (non linearly dependent) arguments. However, many protocols specify a bilinear map where both arguments come from the same group over $\mathbb{F}_p$. Therefore, when using a supersingular curve we need a non-rational endomorphism of the form [22]:

$$\phi : E(\mathbb{F}_p) \to E(\mathbb{F}_{p^k}) \tag{3.59}$$

This mapping is known as a *distortion map* [134]. For a supersingular curve a distortion map always exists, whereas, for non-supersingular curves, no such distortion map exists [134]. We do not go into the details of these distortion maps here.

The modified Tate pairing is generally denoted $\hat{t}$:

$$\hat{t} : E(\mathbb{F}_{p^k})[r] \times E(\mathbb{F}_{p^k})[r] \to \mu_r, \tag{3.60}$$

$$\hat{t}(P, Q) = t(P, \phi(Q)). \tag{3.61}$$

where $\phi(\cdot)$ is used to denote the distortion map.

The distorted Weil pairing is generally denoted $\hat{e}$:

$$\hat{e} : E(\mathbb{F}_{p^k})[r] \times E(\mathbb{F}_{p^k})[r] \to \mu_r, \tag{3.62}$$

$$\hat{e}(P, Q) = e(P, \phi(Q)). \tag{3.63}$$

[5] In common usage, the term "Tate pairing" is generally assumed to refer to the reduced Tate pairing.

Using all of the above techniques, on a supersingular curve, we can take both points from the same group, use the computationally much more efficient Tate pairing and get a concise result (as opposed to an element of an equivalence class) This is now ideal for cryptography

3 5 Miller's Algorithm for Pairing Computation

The methods that we have given so far are probably of more use to a mathematician than a computer programmer There are much more concise, and therefore scalable methods of computing bilinear maps Miller's algorithm which is based on the 'double and add" algorithm for Point Scalar Multiplication (PSM) is at the centre of the construction of the function g which is at the heart of the Weil and Tate pairings Miller's algorithm takes both points and evaluates a partial function at each stage of an iterative process

Let D_S and D_T be two divisors of degree $\neq 0$ with no points in common, such that

1 $\mathrm{sum}(D_S) = S$

2 $\mathrm{sum}(D_T) = T$

and, using the same notation as before, let f_S and f_T be two functions such that

1 $\mathrm{div}(f_S) = rD_S$

2 $\mathrm{div}(f_T) = rD_T$

Then, the Weil pairing is given as

$$e_r(S,T) = \frac{f_T(D_S)}{f_S(D_T)} \tag{3 64}$$

and the Tate pairing can be defined as

$$\langle S,T \rangle = \frac{f_S(T+R)}{f_S(R)} \tag{3 65}$$

Interestingly, the Weil pairing can be expressed in terms of the Tate pairing

$$e_r(S,T) = \frac{\langle T,S\rangle}{\langle S,T\rangle} = \frac{\frac{f_T(S+R)}{f_T(R)}}{\frac{f_S(T+R)}{f_S(R)}} \tag{3 66}$$

Therefore, both pairings rely on an ability to construct the appropriate function f_P with divisor

$$\mathrm{div}(f_P) = r[P+R] - r[R] \tag{3 67}$$

with points $P \in E[r]$ and $R \in E$, efficiently

Miller's idea uses successive doubling to get to r However, one technicality is that $j[P+R]-j[R]$, for values $j < r$ are not divisors of functions[6], however we get a very similar divisor

$$D_{jP} = j[P+R] - j[R] - [jR] + [\mathcal{O}] \tag{3 68}$$

So

$$\mathrm{div}(f_{jP}) = D_{jP} \tag{3 69}$$

Now, assume for a moment that we know $f_j(Q_1)$ and $f_k(Q_2)$ and let $x_{(j+k)} + d = 0$ be the vertical line though $(j+k)P$ Then

$$\mathrm{div}\left(\frac{ax+by+c}{x+d}\right) = [jP] + [kP] - [(j+k)P] - [\mathcal{O}] \tag{3 70}$$

Therefore

$$\mathrm{div}(f_{(j+k)P}) = D_{jP} + D_{kP} + div\left(\frac{ax+by+c}{x+d}\right) = \mathrm{div}\left(f_j f_k \frac{ax+by+c}{x+d}\right) \tag{3 71}$$

[6] $rP = \mathcal{O}$, whereas $jP \neq \mathcal{O}$ $j < r$

To make the example concrete, consider

$$\operatorname{div}(f_{(j+k)P}) = \operatorname{div}\left(f_{jP} f_{kP} \frac{ax + by + c}{x + d}\right) |_{Q_1=(x,y)} \tag{3 72}$$

The above equation is often just written as

$$\operatorname{div}(f_{(j+k)P}) = \operatorname{div}\left(f_{jP} f_{kP} \frac{l}{v}\right) \tag{3 73}$$

where l is the sloping line between the two points (jP and kP) and v is the vertical line passing though kP

To conclude

$$\operatorname{div}(f_P) = r[P + R] - r[R] - [rP] + [\mathcal{O}] = r[P + R] - r[R] \tag{3 74}$$

Therefore, we have successfully constructed the function f_P at the heart of both the Weil and Tate Pairings

We finish this section by giving a concise algorithm for the construction of the Weil and Tate pairings in Algorithm 3 1 There is Java code in the accompanying CD-ROM which implements Miller's algorithm

Algorithm 3 1 is Miller's algorithm for the construction of the reduced Tate pairing

3 6 BKLS Algorithm for Pairing Computation

The BKLS algorithm [14] is a version of Miller's algorithm for efficiently computing the Tate pairing, it makes several improvements for cases that are of cryptographic interest

1 **Denominator Elimination**

If we consider the extremely common 'modified Tate pairing"

$$\hat{t}(P, Q) = t(P, \phi(Q)) \text{ where } P, Q \in E(\mathbb{F}_p)[r] \tag{3 75}$$

Algorithm 3 1 Miller's algorithm for computation of the reduced Tate pairing
INPUT $P \in E(\mathbb{F}_{p^k})[r]$, $Q \in E(\mathbb{F}_{p^k})$
OUTPUT $t_r(P, Q)$

```
Choose suitable S ∈ E(F_{p^k})
Q' ← Q + S
T ← P
m ← ⌊log2(r)⌋ − 1
f ← 1
while (m ≥ 0) do
  T ← 2T
  f ← f^2 (l(Q') v(S)) / (v(Q') l(S))
  if (r_m = 1) then
    T ← T + P
    f ← f (l(Q') v(S)) / (v(Q') l(S))
  end if
  m ← m − 1
end while
return f ← f^{(p^k − 1)/r}
```

we see that denominator elimination can be applied

Denominator elimination can be applied to Miller's algorithm in certain settings By picking parameters as outlined in [14, Sec 5], the denominator (f_2 in 3 1), when exponentiated to $(p-1)^{k/2}$ [7] can be made to become the value $1_{\mathbb{F}_{p^k}}$, and obviously $x/1 = x$, therefore f_2 can simply be ignored This halves the amount of computation in Miller's algorithm

2 Choice of Subgroup Order

Solinas [128] had previously noted that there are many primes that have Hamming weight as low as three[8] Using signed binary representation, these primes can be written as $2^\alpha \pm 2^\beta \pm 1$[9] It is possible to construct elliptic curves such that r, the order of the group $\mathcal{G}$, is a Solinas prime The reduces the amount of computation from $\approx 1\,5 \lg r$ to $\approx \lg r$

[7] As in the reduced Tate pairing
[8] There are only three non zero bits in their binary representation
[9] Alfred Menezes, at ECC summer school 2004, said that the NSA referred to these as "The primes from God"

3 Speeding Up the Final Exponentiation

A sizeable part of the computational effort in evaluating the reduced Tate pairing is the final exponentiation For the $p > 3$ and even k case the BKLS algorithm replaces

$$t = m^{(p^k-1)/r}, \tag{3 76}$$

with

$$x = \bar{m}/m, \tag{3 77}$$

$$t = x^{(p^{k/2}+1)/r}, \tag{3 78}$$

where $\bar{m}$ is the complex conjugate of m

Calculating the conjugate is very efficient, and the exponent is now much smaller – this will lead to a much more efficient implementation

4 Fixed Base Pairing Computation

We can optimise the pairing based on repeatedly using the same base point P When using a fixed base point, the same values will recur in repeated pairing computations These values can be computed just once and stored

When applying precomputation to pairings, the coordinates of these points, along with the slopes of the lines that connect the points are stored, as it is these values that are used in the computation of the function f_P A series of tuples $\{\lambda, x, y\}$ are stored, one for each point that arises in the calculation of rP Then simply recalculate f_P using these stored values and new values for x_Q and y_Q, the co-ordinates of the second point

5 Using MNT curves

For a time it was thought that pairing based cryptography may have to be restricted to supersingular curves Menezes Okamoto and Vanstone had pointed out that supersingular curves have embedding degree of at most 6 [92] Curves of low embedding degree are ideal for pairing based cryptography As it turns out, it is quite easy to construct (non-supersingular) curves with $k \in \{3, 4, 6\}$ A method for generating such curves was first described by Miyaji, Nakabayashi and Takano in [95] (these are known as the MNT curves) Although there is no hard evidence, non-supersingular (a k a 'ordinary") curves are believed to be at least as safe, if not safer than supersingular curves, since they have less structure and there are a lot more of them

Finding curves with larger, but still manageable values of k is an area of great academic interest See for example the work of [15, 16] and recently, work by [17]

We now include the BKLS algorithm from [116], where Q is on the twisted curve[10]

Algorithm 3 2 BKLS algorithm for $k = 2$ computation of the Tate pairing using the Twisted Curve [116]

INPUT $P \in E(\mathbb{F}_p)[r]$, $Q \in E(\mathbb{F}_p)$
OUTPUT $t_r(P, Q)$

$f \leftarrow 1$
$A \leftarrow P$
$n \leftarrow r - 1$
for (i **in** $\lfloor \lg(r) \rfloor - 1$ **downto** 0) **do**
  $f \leftarrow f^2 \quad g(A, A, Q)$
  if ($n_i = 1$) **then**
    $f \leftarrow f \quad g(A, P, Q)$
  end if
end for
$m \leftarrow \bar{m}/m$
$m \leftarrow m^{(p+1)/r}$
return m

[10] For any curve of the form $y^2 = x^3 + Ax + B$ with $\{A, B\} \in \mathbb{Z}_r^*$, the twisted curve is given as $y^2 = x^3 + d^2Ax + d^3B$, where d is any Quadratic Non-Residue mod r

3 7 GHS Optimisations for Pairing Computation

The following three optimisations, which are also in the BKLS paper, are due to Galbraith, Harrison and Soldera in [67] They are observations on the basic Tate pairing that allow it to be implemented more efficiently

1 **Choice of Points**

Compute the pairing using $t(P, Q)$ $P \in E(\mathbb{F}_p)[r]$ Although, for the Tate pairing P does not have to be an element of $E(\mathbb{F}_p)$, making it an element of $E(\mathbb{F}_p)$ results in much smaller representation for λ, x_P, y_P and much more efficient implementation This was coined "Miller-Lite" by Solinas at ECC 2003

2 **Reduce number of $\mathbb{F}_{p^k}$ inversions**

Another implementational issue that Galbraith, Harrison and Soldera noticed is that Miller's algorithm specifies computing a function f_{n_i}/f_{d_i} at each stage and then multiplying these fractions together Obviously, this improvement cannot be used in situations where BKLS [14] denominator elimination already applies

$$f \leftarrow \frac{f_{n_1}}{f_{d_1}} \qquad \frac{f_{n_r}}{f_{d_r}} \tag{3 79}$$

This is much more efficiently implemented as

$$f_n \leftarrow f_{n_1} \qquad f_{n_r} \tag{3 80}$$

$$f_d \leftarrow f_{d_1} \qquad f_{d_r}, \tag{3 81}$$

$$f \leftarrow \frac{f_n}{f_d} \tag{3 82}$$

requiring only one division

3 **Use Faster Point Scalar Multiplication Techniques**

The third observation of Galbraith *et al* is that one can use windowing methods instead of naive bit by bit double and add The authors claim that this method does not change the number of doublings, only reducing the number of additions Therefore it would probably be of little use if using a value r of low Hamming weight

3 8 Products of Pairings

3 8 1 Solinas' Observation

As noted by Solinas at ECC 2003, it is possible to more efficiently compute the product of two or more reduced Tate pairings [129] by using the simple observation that

$$a^e \cdot b^e = (a \cdot b)^e \tag{3 83}$$

As we remarked earlier, the (reduced) Tate pairing requires an application of (some variant of) Miller's algorithm followed by a *final exponentiation* in order to get a concise result For a given curve, this final exponentiation will always be the same value, and is not in any way dependent on the inputs to the Tate pairing

We use m to denote a non-reduced Tate pairing and t to denote a full (reduced) Tate pairing

$$\begin{aligned} t(Q_0, P_0) \cdots t(Q_n, P_n) &= && (3\ 84)\\ &= \langle Q_0, P_0\rangle^e \cdots \langle Q_n, P_n\rangle^e, && (3\ 85)\\ &= (\langle Q_0, P_0\rangle \cdots \langle Q_n, P_n\rangle)^e && (3\ 86)\end{aligned}$$

3 8 2 Scott's Observation

As noted by Scott in [116], it is possible to implement multi-pairing in a manner similar to multi-exponentiation The idea here is that we only have to do one squaring of f, the

'Miller variable" as Scott calls it The basic algorithm is shown in Algorithm 3 3, where we assume that all of the points are distinct (otherwise, the points could just be added before performing the pairing)

Algorithm 3 3 Multi-Miller algorithm for computation of the product of pairings

INPUT $P_1, P_2 \in E(\mathbb{F}_{p^k})[r]$, $Q_1, Q_2 \in E(\mathbb{F}_{p^k})$
OUTPUT $t_r(P_1, Q_1)\ t_r(P_2, Q_2)$

$f \leftarrow 1$
$A_1 \leftarrow P_1$
$A_2 \leftarrow P_2$
$n \leftarrow r - 1$
for (i **in** $\lfloor \log_2(r) \rfloor - 2$ **downto** 0) **do**
  $f \leftarrow f^2\ g(A_1, A_1, Q_1)\ g(A_2, A_2, Q_2)$
  if ($n_i = 1$) **then**
    $f \leftarrow f\ g(A_1, P_1, Q_1)\ g(A_2, P_2, Q_2)$
  end if
end for
$m \leftarrow \bar{m}/m$
$m \leftarrow m^{(p/2+1)/r}$
return m

3 9 Basic Properties of Pairings

Whilst there has been a great deal of research done on the efficient implementation of pairings, as outlined in the proceeding sections of this chapter, a great many papers have been written which simply make use of an abstract bilinear map[11] Many protocols based on pairings do not require specific pairings In this section we will look briefly at the properties of the different pairings In the rest of this section let the points P and P' be two linearly dependent points which are linearly independent of the points Q and Q', which are also linearly dependent

3 9 1 The Weil Pairing

The Weil pairing satisfies the following properties

[11] As at the time of writing this thesis, the only two known bilinear maps are the Weil and Tate pairings, both of which are instanciated over elliptic curves

- **Bilinearity** For all $P, P', Q, Q' \in E[n]$,

$$e(P + P', Q) = e(P, Q) \cdot e(P', Q), \tag{3.87}$$

and

$$e(P, Q + Q') = e(P, Q) \cdot e(P, Q') \tag{3.88}$$

- **Alternating**

$$e(P, P) = 1, \tag{3.89}$$

and

$$e(P, Q) = e(Q, P)^{-1} \tag{3.90}$$

- **Non-degeneracy**

  If $e(P, Q) = 1$ for all $Q \in E[n]$ then $P = \mathcal{O}$

3.9.2 The Tate Pairing

In this section we will concentrate on the reduced Tate pairing since this is the version of the Tate pairing that is used in the construction of cryptographic protocols

The reduced Tate pairing satisfies the following properties

- **Bilinearity** For all P_1, P_2, Q_1 and Q_2 such that $P_i \in E(K)[n]$ and $Q_i \in E(K)/nE(K)$ then

$$t(P_1 + P_2, Q_1) = t(P_1, Q_1) \cdot t(P_2, Q_1), \tag{3.91}$$

and

$$t(P_1, Q_1 + Q_2) = t(P_1, Q_1) \cdot t(P_1, Q_2) \tag{3.92}$$

- **Alternating** As we have already established, if we are using the Tate pairing both points do not have to be the of the same order and so the alternating property is not defined

- **Non-degeneracy** Suppose K is a finite field For all $P \in E(K)[n]$, $P \neq \mathcal{O}$, there is some $Q \in E(K)/nE(K)$ such that $t(P,Q) \neq 1$ Similarly, for all $Q \in E(K)/nE(K)$ with $Q \notin nE(K)$ there is some $P \in E(K)[n]$ such that $t(P,Q) \neq 1$

3 9 3 The Modified Tate Pairing

- **Bilinearity** For all P_1, P_2, Q_1 and Q_2 such that $P_i \in E(K)[n]$ and $Q_i \in E(K)/nE(K)$ then

$$t(P_1 + P_2, Q_1) = t(P_1, Q_1)\ t(P_2, Q_1), \tag{3 93}$$

  and

$$t(P_1, Q_1 + Q_2) = t(P_1, Q_1)\ t(P_1, Q_2) \tag{3 94}$$

- **Alternating** Since we are now using the modified Tate pairing we have the requirement that both points be of the same order So, unlike the regular Tate pairing we can swap the order of the points For the modified Tate pairing we have the following relationship

$$e(P,Q) = e(Q,P) \tag{3 95}$$

- **Non-degeneracy** Suppose K is a finite field For all $P \in E(K)[n]$, $P \neq \mathcal{O}$, there is some $Q \in E(K)/nE(K)$ such that $t(P,Q) \neq 1$ Similarly, for all $Q \in E(K)/nE(K)$ with $Q \notin nE(K)$ there is some $P \in E(K)[n]$ such that $t(P,Q) \neq 1$

3 10 Strategies for Pairing Computation on a Smart card

In this section we look at alternative strategies that are of use for implementing pairings on smart cards, such as 'Chip & PIN" credit cards or SIM's[12] We exploit the idea of Chevallier-Mames *et al* [52]

A typical smart card has a very strictly defined API for interacting with the rest of the world The smart card should have some externally inaccessible memory locations These memory locations should be used to hold sensitive information such as private keys etc It is not possible to read memory locations directly and access to memory is via the card's API, and some logic circuitry on the card

Functions that makes use of the private data (key) should also be on the card For example, consider RSA signing, In this case, an RSA decryption exponent and modulus (d, N) must be present on the card, along with a function f that implements the signing algorithm Any application that wishes to make use of these private keys must, for example, supply all of the other arguments to f, in this case the message Therefore, any card requires (just like any computer), a certain amount of storage and a certain amount of logic circuitry

Chevallier-Mames *et al* suggested a smart card on which no computer program was implemented on the card – the card had no ROM The code was held on the (much more powerful) terminal This is elegant as exactly the same card could be used for multiple tasks depending on the program (terminal) used Any instructions that are given to the card must be signed by the program's author

In joint research with Gemplus[13], we developed a solution similar to that of Chevallier-Mames *et al* The idea here was not to disembed the program, but to go one level deeper and disembed the computationally expensive pairing Obviously our card would need to be more aware of its environment than the card they describe The two objectives of this research were

[12] SIM Subscriber Identification Module

[13] Gemplus was named the worldwide leader of the smart card industry for a seventh consecutive year with a 27% market share, according to market analysts, Gartner Inc (2005)

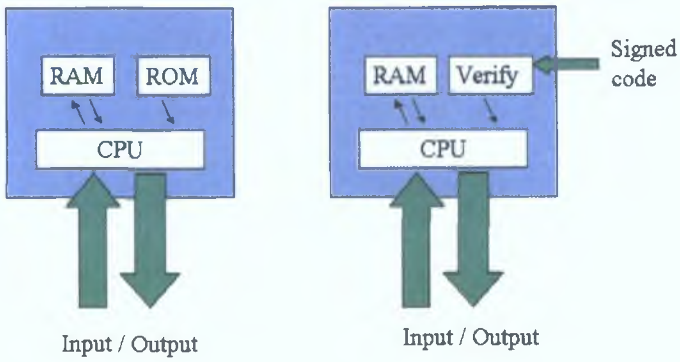


Figure 3.1: A basic interpretation of Chevallier-Mames *et al*'s idea

- Make use of existing cards that are already in production at Gemplus. Would it be possible to implement pairing based protocols on Gemplus cards designed for use with regular ECC algorithms?
- Faster pairings for smart cards. Would utilising a powerful terminal make pairings on a card faster than just implementing the pairing on the card?

We developed a protocol that was to be run between a smartcard and a terminal. The card would output a series of values to the terminal. The card would then receive responses from the terminal. The desired outcome of a run of the protocol was that the card would obtain the result of the pairing and the terminal would not obtain any secret information (such as private keys) from the card. The protocol was to be designed in such a way that:

1. The computationally expensive pairing computation was to be off-loaded to the computationally more powerful terminal. The card was to only use algorithms that it could already implement[14].

[14]This could potentially save a lot of money in the reconfiguration of a Gemplus production line.

2 The smart card would be able to detect a cheating terminal, **abort** and return the $\perp$ symbol

Formally, a protocol is said to be a secure pairing delegation protocol if the following conditions hold [51]

- **Completeness** After completion of the protocol with an honest terminal, the card obtains $e(A, B)$, except with negligible probability
- **Secrecy** A (possibly cheating) terminal should not learn any information about the secret point or points being paired More formally, for any malicious terminal $\mathcal{T}$, there exists a simulator S such that for any points A, B, the output of S is computationally indistinguishable from $\mathcal{T}$'s view S is not given A or B as input
- **Correctness** The card should be able to detect a cheating terminal, except with negligible probability More formally, for any cheating terminal $\mathcal{T}$ and for any A, B, the card outputs either $\perp$ or $e(A, B)$, except with negligible probability

We came up with a number of solutions to this problem These solutions work in a variety of situations, however, the most practical protocols are shown below

Here we show only two of the protocols that we developed

1 Two public points[15], with one constant point This is useful for encryption in Boneh and Franklin's IBE scheme (see Ch 6 for a detailed description of this scheme), where one point is public and constant (the KGC's P_{pub}), and one point is public and variable (the recipient's public key Q_{ID}) Here we reasonably assume that the ciphertext mask in Boneh and Franklin's IBE is calculated in two parts

Boneh and Franklin's IBE encryption

$$g = e(P_{pub}, Q_{ID}) \quad (3\,96)$$

$$\mathcal{M} = g^x \quad (3\,97)$$

[15]These values do not have to remain hidden from the terminal

2 The pairing of two points, one of which is public and the other of which is private and constant This is useful for Boneh and Franklin's IBE decryption or Sakai and Kasahara s IBE decryption (see Ch 6 for a detailed description of this scheme), where one point is an element of the ciphertext and the other element is a long term private key, which will remain constant over many decryptions

Boneh and Franklin's IBE decryption

$$M = e(R, sQ_{ID}) \tag{3 98}$$

Sakai and Kasahara's IBE decryption

$$M = e(R, (s + id)^{-1}Q) \tag{3 99}$$

In the first case, we propose the following protocol

3 10 1 Constant public A and public B

The card and the terminal are given as input a description of the groups $\mathcal{G}$ and μ_r, and a description of the bilinear map e $\mathcal{G} \times \mathcal{G} \rightarrow \mu_r$ Moreover, the card receives and stores the tuple $(e(A, Q), Q)$ for some random $Q \in \mathcal{G}$ These two elements are trusted to be related as described, and so are assumed to have come from a trusted party These two values will act as reference values in future calculations by the card The point Q and the value $e(A, Q)$ are kept private by the card The card is given as input the point B and must eventually output $e(A, B)$

The card generates a random $x \in \mathbb{Z}_r^*$ and queries the following pairings to the terminal

$$\alpha_1 = e(A, B), \tag{3 100}$$

$$\alpha_2 = e(A, xB + Q) \tag{3 101}$$

The card checks that

$$\alpha_1^r \ e(A,Q) = \alpha_2, \tag{3 102}$$

and that $\alpha_1^r = 1_{\mu_r}$ In this case, it outputs α_1, otherwise it outputs $\perp$

The protocol requires only one scalar multiplication and two exponentiations in μ_r, it can also make use of existing hardware that efficiently implements point scalar multiplication Efficient point scalar multiplication is a more mature area than efficient pairing implementation

Theorem 3 10 1 *The previous protocol with constant public A and public B is a secure pairing delegation protocol*

Proof We do not have to prove the secrecy property since both points being paired are public values

The completeness property is straightforward to establish The protocol's correctness is shown as follows Let b be such $B = bP$ Let q be such that $Q = qP$ Let

$$u = xb + q \mod r, \tag{3 103}$$

which gives $xB + Q = uP$ We have that the terminal's view is entirely determined by (b, u) and by the randomness used by $\mathcal{T}$ Since x and q are randomly selected from $\mathbb{Z}_r^*$, we obtain that the distribution of x is independent from the terminal's view

Let β_1, β_2 be such that

$$\alpha_1 = e(A,B) \ e(A,P)^{\beta_1}, \tag{3 104}$$

$$\alpha_2 = e(A, xB+Q) \ e(A,P)^{\beta_2} \tag{3 105}$$

We have that β_1, β_2 are a function of the terminal's view, and that $\alpha_1 = e(A,B)$ if $\beta_1 = 0$ Moreover, we obtain from 3 102 that the card outputs α_1 iff

$$x\beta_1 = \beta_2 \mod r \tag{3 106}$$

Now, we know that $\beta_1 \neq 0$ Then since β_1 and β_2 are a function of the terminal's view, and the distribution of r is independent from the terminal's view, equality (3 106) holds with probability at most $1/r$ Therefore, for any cheating terminal, the card outputs either $\perp$ or the correct $e(A, B)$, except with probability at most $1/r$ □

In the second case we have

3 10 2 Constant private A and public B

The card and the terminal are given as input a description of the groups $\mathcal{G}$ and μ_r, and a description of the bilinear map $e \ \mathcal{G} \times \mathcal{G} \to \mu_r$ Moreover, the card receives $e(A, Q)$ for some random $Q \in \mathcal{G}$ The points A, Q and the value $e(A, Q)$ are kept private by the card The card is given as input the point B and must eventually output $e(A, B)$

The card generates random $x, y, z \in \mathbb{Z}_r^*$ and queries the following pairings to the terminal

$$\alpha_1 = e(xA, B), \tag{3 107}$$

$$\alpha_2 = e(yA, z(B + Q)) \tag{3 108}$$

The card computes

$$e_{AB} = \alpha_1^{x^{-1}}, \tag{3 109}$$

$$\alpha_3 = \alpha_2^{(yz)^{-1}} \tag{3 110}$$

The card checks that

$$e_{AB} \ e(A, Q) = \alpha_3, \tag{3 111}$$

and that $e^r_{AB} = 1$ In this case, it outputs e_{AB}, otherwise it outputs $\perp$ The protocol requires only 3 scalar multiplications and 3 exponentiations in μ_r

Theorem 3 10 2 *The previous protocol with constant private A and public B is a secure pairing delegation protocol*

Proof The protocol's completeness is easily established The protocol's secrecy follows from the fact that the terminal receives only randomly distributed points The protocol's correctness is established as follows Let b be such $B = bP$ Let q be such that $Q = qP$ Let

$$u = z(b+q) \mod r, \tag{3 112}$$

which gives $z(B+Q) = uP$ The terminal's view is then entirely determined by (b, u, xA, yA) and by the randomness used by $\mathcal{T}$ Since z and q are randomly generated in $\mathbb{Z}_r^*$, we obtain that the distribution of z is independent from the terminal's view Let α_1, α_2 be such that

$$\alpha_1 = e(xA, B)^{1+\beta_1}, \tag{3 113}$$

$$\alpha_2 = e(yA, B+Q)^{1+\beta_2} \tag{3 114}$$

We have that α_1 and α_2 are a function of the terminal's view Moreover, we obtain

$$e_{AB} = e(A, B)^{1+\beta_1}, \tag{3 115}$$

$$\alpha_3 = e(A, B+Q)^{1+\beta_2} \tag{3 116}$$

Therefore, $e_{AB} = e(A, B)$ iff $\beta_1 = 0$ Moreover, we obtain from (3 111) that the card outputs e_{AB} iff

$$e(A, B+Q)^{\beta_1} = e(A, B)^{\beta_2}, \tag{3 117}$$

which gives

$$b\beta_1 = (b+q)\beta_2 \mod r \tag{3 118}$$

Then since b, β_1, β_2 are a function of the terminal's view, and the distribution of x is uniform in $\mathbb{Z}_r^*$, independent of the terminal's view, we obtain that if $\beta_1 \neq 0$, the equality 3 118 holds with probability at most $1/r$ Therefore, for any cheating terminal, the card outputs either $\perp$ or the correct $e(A, B)$, except with probability $1/r$ □

3 11 Conclusion

In this section we have given, in the Weil and Tate pairings, concrete examples of the pairings that we will be using to implement the various cryptographic protocols that we go on to describe in the following chapters We have given accompanying code in the appendices We have shown some of the tricks that can be used, in cases of cryptographic interest, and shown this to be a progressive area of research

We have shown some techniques that could be used to convert existing Gemplus smart cards into cards suitable for use with pairing based protocols Although we do not have precise timings for these results we were told that the time to implement a pairing on a card is greater than 2 seconds, whereas with our scheme it took approximately 1/2 second [96] [16]

[16] Advances in pairing implementation research suggest that it will be practical to implement pairings directly on smart cards over the next 2 - 5 years

Chapter 4

Cryptographically Hard Problems

In this chapter we explain some mathematical, complexity theoretic and number theoretic concepts These concepts are reasonably straightforward, but are sometimes clouded in mathematical language that only serves to discourage their understanding We explain what is meant by a cryptographically hard problem There are certain problems that are believed to be *intractable* Cryptographic systems can be based on these problems

These *intractable* problems are said to be *cryptographically hard* or *computationally infeasible* in certain settings The following definitions are all taken from the American government run National Institute of Standards in Technology (NIST) [20] Another useful reference for this material is [65] These definitions are for the technical meaning of these terms and may differ from those found in a non-specialist dictionary, but are appropriate for this thesis

Definition [20] *Algorithm* A computable set of steps to achieve a desired result

In layman's terms, any computer program could be described as implementing an algorithm The type of algorithms that we are interested in are those that solve *cryptographically hard problems*

Definition [20] *big-O notation* $f(n) = O(g(n))$ means there are positive constants c and k, such that $0 \leq f(n) \leq cg(n)$ for all $n \geq k$ The values of c and k must be fixed for the function f and must not depend on n.

The complexity of an algorithm is expressed using what is called "big-O" notation. Big-O notation is used to used to describe an asymptotic upper bound for the magnitude of a function in terms of another, usually simpler, function [65]. For the algorithms that we will be examining here, we are interested in limits in running time and storage.

Definition [20] *Linear time:* The measure of computation, $m(n)$ (usually execution time or memory space), is bounded by a linear function of the problem size, n. More formally $m(n) = O(n)$.

Definition [20] *Polynomial time:* When the execution time of a computation, $m(n)$, is no more than a polynomial function of the problem size, n. More formally $m(n) = O(n^k)$ where k is a constant.

Definition [20] *Exponential time algorithm:* In complexity theory, the measure of computation, $m(n)$ is bounded by an exponential function of the problem size, n. More formally if there exists a $c > 1$ such that $m(n) = O(c^n)$.

Definition [20] *Moderately (Sub) Exponential time algorithm:* The measure of computation, $m(n)$ is more than any polynomial n^k, but less than any exponential c^n where $c > 1$.

Cryptographic systems should be based on problems which are intractable:

Definition [20] *Intractable:* A problem for which no algorithm exists which computes all instances of it in polynomial time.

When we develop a cryptographic protocol, such as we do in Ch. 5, 6, 7, and 8, we wish to link the difficulty of breaking the system with the ability to solve an intractable problem. We will show this in detail when we give security arguments for the schemes that we develop.

Fundamentally there are three intractable problems that cryptosystems are based around.

- **Integer Factorisation Problem** (a k a factoring) [91, Ch 3] Given a positive integer n, find its prime factorization, that is, write $n = p_1^{e_1}, p_2^{e_2}, \quad, p_k^{e_k}$ where the p_i are pairwise distinct primes and each $e_i \geq 1$

- **(Generalised) Discrete Logarithm Problem** [91, Ch 3] Given a finite cyclic group $\mathcal{G}$ of order n, a generator α of $\mathcal{G}$, and an element $\beta \in \mathcal{G}$, find the integer x, $0 \leq x \leq n-1$, such that $\alpha^x = \beta$

- **Shortest Vector Problem** Given a lattice L, find the shortest non-zero vector contained in L There may be several vectors of the same length This is the basis of NtruEncrypt [75] and other lattice based cryptosystems

Usually we do not know if the underlying problem really is intractable But these are well studied problems, and no known efficient algorithms to solve them exist That is why they are sometimes referred to as 'assumed to be hard" problems

4 1 Cryptographically Hard Problems Over Elliptic Curves

In the specific area of pairing based cryptography the following is a list of important problems Some are intractable, and others, with current knowledge, can only be solved using bilinear maps This list is not exhaustive and the number of intractable problems in this area is growing Some work in proposing new hard problems has been done by Boneh and Boyen and others Other researchers feel uncomfortable trusting new, less well studied problems The belief that a problem is intractable grows the more that problem is studied The groups $\mathcal{G}$ and μ_r, that we refer to in the list, are those groups such that a bilinear map operates $\hat{e} \quad \mathcal{G} \times \mathcal{G} \rightarrow \mu_r$ We assume that $\langle P \rangle = \mathcal{G}$, $\langle g \rangle = \mu_r$, and that $g = \hat{e}(P, P)$

A few good references for this section are [91, ch 3], [140] and [12] In this section we concentrate on problems that require use of a distortion map (and so must be implemented over a supersingular curve) For each of these problems there is a corresponding "co" problem which can be set over non-supersingular curves

- **Bilinear Diffie-Hellman Problem** Given P, xP, yP and $zP \in \mathcal{G}$, Compute $g^{xyz} \in \mu_r$

  This problem is intractable

- **Decisional Diffie-Hellman in $\mathcal{G}$** Given P, xP, yP and $zP \in \mathcal{G}$, Decide if $xy = z$

  This problem is easy using the bilinear map Simply check the following equality

$$\hat{e}(xP, yP) \stackrel{?}{=} \hat{e}(P, zP) \tag{4.1}$$

- **Decisional Diffie-Hellman in μ_r** Given g, g^x, g^y and $Z \in \mu_r$, Decide if $Z = g^{xy}$?

  This problem is intractable

- **Computational Diffie-Hellman in $\mathcal{G}$** Given P, xP and $yP \in \mathcal{G}$, Compute xyP

  This problem is intractable

- **Computational Diffie-Hellman in μ_r** Given g, g^x and $g^y \in \mu_r$, Compute g^{xy}

  This problem is intractable

- **Discrete Logarithm Problem in $\mathcal{G}$** Given P and $xP \in \mathcal{G}$, Compute x

  This problem is intractable

- **Discrete Logarithm Problem in μ_r** Given g and $g^x \in \mu_r$, Compute x

  This problem is intractable

- **Inverse Computational Diffie-Hellman Problem in $\mathcal{G}$** Given P and $xP \in \mathcal{G}$, Compute $x^{-1}P$

  This problem is intractable

- **Inverse Computational Diffie-Hellman Problem in μ_r** Given g and $g^x \in \mu_r$, Compute $g^{x^{-1}}$

  This problem is intractable

- **Inverse Decisional Diffie-Hellman Problem in $\mathcal{G}$** Given P, xP and $Z \in G$, Decide if $Z = x^{-1}P$

  This problem can be solved using the bilinear map

$$\hat{e}(xP, Z) \stackrel{?}{=} \hat{e}(P, P) \tag{4 2}$$

- **Inverse Decisional Diffie-Hellman Problem in μ_r** Given g, g^x and $\gamma \in \mu_r$, Decide if $\gamma = g^{x^{-1}}$

  This problem is intractable

- **Divisible Computational Diffie-Hellman Problem in $\mathcal{G}$** Given P, xP and $yP \in \mathcal{G}$, Compute $(x/y)P$

  This problem is intractable

- **Divisible Computational Diffie-Hellman Problem in μ_r** Given g, g^x and $g^y \in \mu_r$, Compute $g^{x/y}$

  This problem is intractable

- **Divisible Decisional Diffie-Hellman Problem in $\mathcal{G}$** Given P, xP, yP and $Z \in \mathcal{G}$, Decide if $Z = (x/y)P$

  This problem is intractable

- **Divisible Decisional Diffie-Hellman Problem in μ_r** Given g, g^x, g^y and $Z \in \mu_r$, Compute $Z = g^{x/y}$

  This problem is intractable

- **Square Computational Diffie-Hellman Problem in $\mathcal{G}$** Given P and $xP \in \mathcal{G}$, Compute x^2P

  This problem is intractable

- **Square Computational Diffie-Hellman Problem in μ_r** Given g and $g^x \in \mu_r$, Compute g^{x^2}

This problem is intractable

- **Square Decisional Diffie-Hellman Problem in $\mathcal{G}$** Given P, xP and $Z \in \mathcal{G}$, Decide if $Z = x^2P$

  This problem can be solved using a bilinear pairing

$$e(P, x^2P) \stackrel{?}{=} e(xP, xP) \tag{4 3}$$

- **Square Decisional Diffie-Hellman Problem in μ_r** Given g, g^x and $\gamma \in \mu_r$, Decide if $\gamma = g^{x^2}$

  This problem is intractable

- **Bilinear Pairing Inversion Problem** Given $P \in G$ and $\gamma \in \mu_r$, where $\gamma = \hat{e}(P, Q) \in \mu_r$, Compute $Q \in \mathcal{G}$

  This problem is intractable

- **Bilinear Inversion Diffie-Hellman Problem** Given $P, aP, bP \in G$, Compute $e(P, P)^{a^{-1}b} \in \mu_r$

  This problem is intractable

- **q-Strong Diffie-Hellman Problem** Given the $(q + 1)$-tuple $\{P, xP, x^2P, \quad , x^qP\} \in \mathcal{G}^{q+1}$, where $q \geq 1$, Calculate a tuple $((x + y)^{-1}P, y)$

  This problem is intractable

- **q-Bilinear Diffie-Hellman Inverse problem** Given the $(q + 1)$-tuple $\{P, xP, x^2P, \quad , x^qP\} \in \mathcal{G}^{q+1}$, where $q \geq 1$, Compute $g^{x^{-1}} \in \mu_r$

  This problem is intractable

- **Decisional q-Bilinear Diffie-Hellman Inverse problem** Given the $(q+1)$-tuple $\{P, xP, x^2P, \quad , x^qP\} \in \mathcal{G}^{q+1}$, where $q \geq 1$ and $Z \in \mu_r$, Decide if $Z = g^{x^{-1}}$, where $g = e(P, P)$

  This problem is intractable

Some assumptions are said to be stronger than others, and conversely some are said to be weaker When an assumption $\mathcal{A}$ is said to be weak with respect to another assumption $\mathcal{B}$, it implies that the underlying problem of $\mathcal{A}$ is at least as difficult, if not more difficult than the problem underlying assumption $\mathcal{B}$ This is demonstrated by showing that an oracle that can break $\mathcal{A}$ can be used to break $\mathcal{B}$[1], but not being able to show the inverse

We are confident that the indicated problems are indeed intractable If the security parameter is chosen to be large enough, that is, if r, the order of the groups $\mathcal{G}$ and μ_r is a prime of at least 2^{160}, then solving the above problems is currently computationally infeasible It is extremely important that $\mathcal{G}$ and μ_r are chosen carefully, and standardisation bodies, such as NIST or IEEE usually publish suitable parameters [2] We will assume for the remainder of this thesis that r, the order of G and μ_r is prime

4 2 Methods of Solving the Discrete Logarithm Problem

We now look at some of the best methods used to attack the elliptic curve discrete logarithm problem The most important method used to attack the discrete logarithm problem *over finite fields* is the Index Calculus Attack However this method cannot be applied directly to elliptic curves We will explain the reason for this in detail later in this section, however, the important implication of this is that *ECC can use smaller key sizes than discrete logarithm systems over finite fields, for the same conjectured level of security* Since smaller key sizes generally mean less computationally expensive algorithms, this has resulted in the widespread use of ECC in constrained devices such as wireless microcontrollers and mobile phones

For clarity, we state once again the discrete logarithm problem over elliptic curves

- **EC Discrete Logarithm Problem** Given linearly dependent points[3] P and $Q \in \mathcal{G}$

[1] Remember, there may be other ways to break $\mathcal{B}$ that may not involve breaking $\mathcal{A}$

[2] This has not yet happened for pairing based cryptography as it is such a new technology, but the author is an active participant in IEEE standardisation meetings in this area The IEEE P1363 hope to propose standards in 2008

[3] Any two points P and Q are said to be linearly dependent if there is some x such that $Q = xP$

Calculate $x \in \mathbb{Z}_r^*$, such that $Q = xP$

Obviously one naive method of solving the discrete logarithm problem over elliptic curves is to try all possible values $x \in \mathbb{Z}_r^*$, where r is the order of the group $\mathcal{G}$ This is known as the exhaustive search method But there are much better algorithms for solving the EC discrete logarithm problem

4 2 1 Shank's Baby Step Giant Step Method

The Baby Step Giant Step method was developed by Shanks in [119] It is a time versus memory trade-off of the exhaustive search algorithm The idea here is to break the problem down into two smaller problems that both have $\approx \sqrt{r}$ steps, where r is the order of the group $\mathcal{G}$ One part of the algorithm takes "Giant" steps amongst elements of the group $\mathcal{G}$, whereas the other part of the algorithm takes 'Baby" steps

The algorithm proceeds as follows

Let $m = \lceil\sqrt{r}\rceil$, where r is the order of P If $Q = xP$, then x can be written as $x = im + j$, where $0 \leq i, j \leq m$ Therefore $xP = imP + jP$ This equation can be rewritten, $xP - imP = jP$ This is the basis of the Baby Step Giant Step algorithm

- Construct a table of size m and populate this table with tuples for (j, jP), for all values $0 \leq j \leq m$ Sort this table in ascending order based on the jP values
- Calculate the value imP, for $i = 0$ (this will be $\mathcal{O}$) Check if $xP - imP \stackrel{?}{=} jP$ If not, increment i, and repeat until the verification equation $xP - imP \stackrel{?}{=} jP$ is true
- Return the value $x = im + j \mod r$ This is the discrete logarithm of xP with respect to P

Shank's Baby Step Giant Step algorithm requires $O(\lceil\sqrt{r}\rceil)$ storage, and $O(\lceil\sqrt{r}\rceil)$ point scalar multiplications When $r \approx 2^{160}$, this attack would require approximately 2^{81} operations, and a table with 2^{80} storage entries and on average $1\ 5 \times 2^{80}$ point scalar multiplications Whilst being a huge improvement on the exhaustive search algorithm which would require on average $2^{159} = O(r)$ point additions, this is still impractical

4 2 2 Pollard's ρ Method

We now look at Pollard's ρ method for solving the discrete logarithm problem [105] Again we start off with the same basic problem, which is, given P and Q, such that $Q = xP$, find x The crux of Pollard's algorithm is to find two different ways of expressing any point in terms of the points P and Q Say for example we know that $R = aP + bQ$, and that $R = kP + yQ$ Then we have $aP + bQ = kP + yQ$, but we also know $Q = xP$, so we have $aP + bxP = kP + yxP$ which gives $(a - k)P = (y - b)xP$ which implies $x = (a - k)(y - b)^{-1}$ mod r

Formally, Pollard's algorithm needs a random function $f \quad \mathcal{G} \rightarrow \mathcal{G} \times \mathbb{Z}_r^* \times \mathbb{Z}_r^*$ f is a pseudo-random function That is, given the same input point, it will always return the same random output point However, we also need the function to return useful additional information about the point that is returned The function also returns two elements in $\mathbb{Z}_r^*$, these are the coefficients k and y in the equation $X = kP + yQ$, where X is the point returned by the function, and P and Q are the points for which the discrete logarithm of Q with respect to P is to be determined

If the function f is truly random, the expected running time of this algorithm is approximately $O(\sqrt{r})$ due to the birthday paradox, where r is the order of $\mathcal{G}$ All of the points that are generated $\{X_0, X_1, \quad , X_n\}$ need to be stored, and this list needs to be searched though every time to see if we have a match between the current point and any previous point

However, Floyd [63] has proposed a more elegant solution, Floyd's cycle finding algorithm, which make use of a slow moving pointer (sometimes called a tortoise) and a fast moving pointer (sometimes called a hare) proceeds as follows

- make a pointer to the first element (the hare)
- make a pointer to the first element (the tortoise)
- advance the hare by two iterations for every one iteration by the tortoise

- since the group is finite and cyclic the hare and tortoise will meet

The total amount of computation for this algorithm is $O(\sqrt{r})$ Again, the value x is recovered as $x = (a - k)(y - b)^{-1} \mod r$, where the two representations of the point recovered are $R = kP + yQ$ and $R = aP + bQ$ Pollard's ρ method is probabilistic, meaning that it is not guaranteed to finish within this computational bound, but it is expected to do so with very high probability

4 2 3 Pollard's λ Method

Pollard's λ method [105] is very similar to Pollard's ρ algorithm It relies on a similar method of finding a point that can be represented in two separate ways using the points P and Q as a basis It also uses a random function f The main idea here is that one can use several random starting points $\{P_0, \quad , P_n\}$ The name λ comes from the fact that the algorithm starts at 2 (or more) separate points and converges Once the two 'walks" meet they will coincide thereafter This is reminiscent of the greek letter λ Again this algorithm, like Pollard's ρ algorithm, is probabilistic

4 2 4 The Index Calculus Attack

The index calculus method is an ingenious way to calculate the discrete logarithm of a one element with respect to a generator element in a finite field $\mathbb{F}_p$ It is one of the most powerful attacks against the discrete logarithm problem over the finite field However, we must point out from the start that the index calculus attack cannot be used directly against elliptic curves[4] This is extremely important, as it is this fact that allows us to use much smaller key sizes for elliptic curve cryptosystem See Table 4 1 for details

The reason that the index calculus attack does not work in the elliptic curve setting is that it requires elements of the group $\mathcal{G}$ be factored If we take elements in the finite field

[4]This statement may no longer be true, due to research by Gaudry and Diem, which is not in my area of expertise [69, 57], however, their work only applies to curves over extension fields of certain degrees, and so these curves can be easily avoided

ECC key size (bits)	El Gamal key size (bits)	Ratio ECC/El Gamal
163	1024	0 159
256	3072	0 083
384	7680	0 05
512	15360	0 03

Table 4 1 Key Sizes needed for Comparable Security [40, with reference to NIST]

$\mathbb{F}_p$, they are the integers $\{0, 1, \quad, (p-1)\}$ These numbers can usually be easily factored The series of prime factors is called the *factor base*

Let p be a large prime and g be a generator element of the group $\mathbb{F}_p^*$ Then any element $h \in \{1, \quad, (p-1)\}$ can be written as

$$h = g^k \mod p \tag{4 4}$$

for some unique k with $0 \leq k \leq p-2$ k is the discrete logarithm of h with respect to the base g

Now, let h be an integer, and let $h' = g^{k'} \mod p$ be another integer Then we know that

$$h \quad h' = (g^k \quad g^{k'}) \mod p \tag{4 5}$$

or

$$h \quad h' = (g^{k+k'}) \mod p \tag{4 6}$$

We also know that $h^2 = h \quad h = g^{k+k} = g^{2k} \mod p$

Also, any integer can be expressed as

$$n = h_0^{e_0} \quad h_1^{e_1} \qquad h_n^{e_n} \tag{4 7}$$

where $\{q_0 \quad q_n\}$ are the factors of n and $e_i \geq 1$ The goal of the Index Calculus attack is to build up a table of 2-tuples (q, k), where q is a factor of n and $q = g^k \mod p$ Once we are able to express n in terms of factors for which we know the appropriate k we can solve the discrete logarithm problem

If we view (q_i, k_i, e_i) as a matching set, and n can be factored as

$$g^x = n = q_0^{e_0} \quad q_1^{e_1} \qquad q_n^{e_n} \tag{4 8}$$

which is the same as

$$g^x = n = g^{k_0 e_0} \quad g^{k_1 e_1} \qquad g^{k_n e_n} \tag{4 9}$$

then

$$x = k_0 e_0 + k_1 e_1 + \qquad + k_n e_n \mod (p-1) \tag{4 10}$$

We will now give a trivial example of the index calculus method in action Suppose one wants to find the discrete logarithm of 15 to the base 3 mod 23, i e find x such that

$$8 = 3^x \mod 23 \tag{4 11}$$

First, build up a *factor base* The factor base is a relatively small subset of the elements of $\mathcal{G}$, such that a significant fraction of elements of $\mathcal{G}$ can be efficiently expressed as products of elements from the factor base For each element in the factor base, the discrete logarithm for that element (to the base g) is known For example,

$$3 = 3^1 \mod 23 \tag{4 12}$$

$$9 = 3^2 \mod 23 \tag{4 13}$$

$$4 = 3^3 \mod 23 \tag{4 14}$$

$$12 = 3^4 \mod 23 \tag{4 15}$$

$$13 = 3^5 \mod 23 \tag{4 16}$$

$$16 = 3^6 \mod 23 \tag{4 17}$$

$$2 = 3^7 \mod 23 \tag{4 18}$$

From these equations we can build up a table of (x, g^x) pairs This is the factor base We then use these values to compute the discrete logarithm for any other element For example, we have

$$8 = 4 \ 2 = 3^3 \ 3^7 = 3^{3+7} \mod 23 \tag{4 19}$$

and so

$x = 10$

The above is a very basic example, meant only to let the reader understand the basic operation of the index calculus attacks A more complex example is given in [137]

4 2 5 The MOV Attack

Menezes, Okamoto, and Vanstone (MOV) [92] proposed the following attack that reduces the EC-DLP to a DLP in a finite field The idea is that the r^{th} roots of unity group is a subgroup of a finite field Therefore we can use the following observation to allow the application of powerful Index Calculus attacks on EC-DLP

Given P, a point of order r, and $Q = xP$, find x

First select a suitable constant point T, the second input to the bilinear pairing e Then compute the following pairing values

$$e(P,T) = g \in \mu_r \qquad (4\ 20)$$

$$e(Q,T) = e(P,T)^x = g^x \in \mu_r \qquad (4\ 21)$$

Now solve for x, using the values g and g^x

Although this looks a very similar problem, it is now set in a finite field where it can be solved using index calculus methods

Obviously for this attack to succeed, it is important that elements of μ_r can be easily manipulated and therefore that the problem be set over an elliptic curve with small embedding degree For standard elliptic curve cryptosystems we tend to avoid such curves However, we also need this property (and therefore curves of small embedding degree) for pairing based cryptography Provided we are careful in our choice of parameters pairing based cryptography is secure This means $q \geq 2^{160}$ and $q^k \geq 2^{1024}$, where k is the embedding degree of the curve

4 2 6 Using Security Definitions

We have looked at a variety of intractable problems in this chapter But why are these problems important to cryptography? The security of cryptographic protocols is often linked to one of these problems, using what is sometimes called 'proof by reduction"

The idea is to model an adversary of a particular cryptosystem, and to give that adversary every conceivable advantage to break the system in a non-trivial fashion If we are to prove that a new security protocol is secure then we should be able to show a reduction from having a non-negligible advantage in breaking our system to have a non-negligible advantage in solving one of the hard problems mentioned previously When we link a protocol to a specific hard problem, that problem is said to be the 'underlying hard problem" for the system Of course, should that hard problem be flawed, then the protocol, and any other protocols based on the same hard problem, can be broken

Proofs in this model normally proceed as follows

- **Define an adversary** E For our protocol we define an adversary by defining the scope of its powers and its goal The scope of the adversary's powers are different depending on the security objectives of the protocol For example, it might be to generate a signature without the correct private key, distinguish between the encryption of one message and another without the correct private key, or complete an authenticated key agreement without the correct private key

  As an example, for an encryption scheme we might say that we have an adversary E who

  - might have access to all public keys of the system, and all private keys of the system apart from the one which trivially decrypts the message This defines the scope of its powers
  - might wish to distinguish between the encryption of messages m_0 and m_1, encrypted under a public key for which E does not know the corresponding private key This defines it's goal

  We then define an algorithm $\mathcal{A}$ $\mathcal{A}$'s job is, by interacting with E, to solve the underlying hard problem How $\mathcal{A}$ does this is simply by imitating E's environment exactly, and getting results to particular queries back from E But $\mathcal{A}$ can store extra information, for example, $\mathcal{A}$ would be allowed to know the discrete logarithm of points that are, from E's view of the system, mapped via an idealised hash function[5], provided of course that the point is random and that the discrete logarithm is not disclosed to E It is essential that E's view of the world is exactly as he would expect if he was breaking the protocol

  $\mathcal{A}$ uses E's answers, and E's inability to distinguish between its simulated environment and the real world, to solve the underlying hard problem Since we assume that $\mathcal{A}$

[5] This is called a Random Oracle

cannot solve the hard problem Then E, who can break the protocol, cannot exist Therefore the protocol is secure

Chapter 5

Signature Schemes using Bilinear Maps

A digital signature on a message is a value, or series of values, which is generated using both a message and a private key It is important that a valid digital signature can only be created by an entity in possession of the correct private key It may be deterministic - that is given any private key and any message there is only one valid signature (the RSA signature [107] is an example of a deterministic signature), or it may be randomised - given any private key and any message there may be many valid signatures (the El Gamal signature [61] is an example of a randomised signature)

The purpose of a digital signature is to provide the following assurances

1 **Message Origin Authentication** The identity of the signer of the message is known

2 **Message Integrity** The message has not been altered since it was signed

3 **Non-Repudiation** The signer cannot later deny having signed the message

A digital signature is checked using a public key Every digitial signature verification reduces to an equation which includes the public key of the claimed signer, the signature element or elements and the message that was purportedly signed. If the verification equation is passed we can be confident that the message was signed by the holder of the private key.

$$\gamma \leftarrow V_{k_{pub}}(S_{k_{pri}}(m), m') \tag{5.1}$$

where γ is the result of the verification algorithm V, S is the signing algorithm, k_{pub} and k_{pri} are a matching key pair, m is the message that was signed and m' is the message as received by the verifier.

γ will be `true` iff $m = m'$ and $\{k_{pub}, k_{pri}\}$ is a valid key pair[1].

What is the message? By message we mean any piece of data - for example a Microsoft Word® document or an MP3 music file. Someone would want to sign a digital document for the same reasons they would want to physically sign that document once it was printed out and in the form of a hard copy. Perhaps it is a contact by which two parties agree to do business. Perhaps the person wants to claim ownership of ideas in a document or copyright of a song. This can be achieved with the assistance of a notary public.

Usually the message is pre-processed using a cryptographic hash function[2], as this produces a much smaller hash value. This hash value can then be further processed to produce the signture[3]. This is much more efficient. A hash function should have the property that it is not possible to find a message that hashes to a predetermined value - this is known as "pre-image resistance". It should also be "collision-resistant", meaning that it should not be possible to find any two messages m_0 and m_1 such that, using a hash function $\mathcal{H}$, $\mathcal{H}(m_0) = \mathcal{H}(m_1)$. This is to prevent an attack whereby one message is exchanged for another with the same hash value.

There are several non-identity based digital signatures, for example those in [107, 64, 112, 100]. In this chapter we will examine digital signature schemes that arise out of

[1] Except with negligible probability.

[2] Often the message is hashed together with a random signature element.

[3] A signature may consist of several elements.

bilinear maps. There are many different digital signature schemes that utilise pairings, some with interesting and novel properties; for example, conditionally verifiable signatures [42], aggregate signatures [32, 25, 50], multi designated verifier [80], blind [25, 144] and ring signatures [7, 53, 73, 84, 139].

We concentrate on standard identity based signature schemes in this chapter. Sakai-Ogishi-Kasahara [111] presented the first identity based signature. A more efficient scheme was proposed soon after by Paterson [102]. Cha-Cheon [41] formally defined a security model for identity based signatures, and in [50] Cheon, Kim and Yoon altered this signature to allow for batch verification. In [141] Yi proposed a signature scheme similar to that of Cha-Cheon with point reduction. Other signatures of note include [74], and two pairing based signatures by Sakai and Kasahara in [109]. A large number of identity based signatures were proved secure in a framework proposed in [18].

There are a number of important pairing based, but not identity based signatures, such as [33, 145]. The BLS and ZSNS signatures are useful because they produce the shortest secure (traditional) PKI signatures.

Recently there have been a number of non-identity based signature schemes that have been proven secure in the standard model, for example [28] and [143]. The signatures produced by these schemes are larger than the corresponding signatures produced by schemes proven secure in the Random Oracle Model (ROM), but are assumed to be "safer". At present there is a trend away from schemes proven secure in the ROM. This has been fuelled by the observation of Goldreich *et al.* that proofs in the ROM do not necessarily convert to secure schemes when the random oracles are instantiated [36]. There has also been some success in attacking modern hash functions such as SHA-1[4] [135] and MD-5 [136]. However, moving away from the random oracle model causes problems of its own. Now cryptographers generally must trust much less well studied hard problems.

For many important dealings, i.e. buying a car or house, most people would feel more comfortable with handwritten signatures on hard copies of documents, but Irish law [101]

[4]Collisions in the full SHA-1 in 2^{69} hash operations, much less than the brute-force attack of 2^{80} operations based on the hash length.

(Electronic Commerce Bill 2000), in line with EU directive 1999/93/EC [59] makes no distinction between handwritten and electronic signatures The Electronic Commerce Bill, 2000, is written in such a way as to be as flexible as possible and does not specify which algorithms must be used for the digital signature to be legally binding This leaves another caveat as the legal situation with regard to identity based signatures is somewhat unclear Identity based signature schemes inherently make use of a KGC which knows all of the private keys in the system Therefore it is trivial for the KGC to be able to forge signatures in the system A similar issue pertains for traditional PKI signatures generated using private keys that are known to more than one party Or indeed the PKI may produce false certificates and forge signatures in this manner

In this chapter we will be looking at traditional PKI signatures and identity based signatures that can be constructed using bilinear maps We will look briefly at the security models for each of these types of signatures

5.1 Definitions of PKI and IB Digital Signature Schemes

A standard PKI digital signature scheme consists of the following three algorithms **KeyGen**, **Sign**, and **Verify**

- **KeyGen** A random public key pair is produced, and the public component, along with any system parameters, is made public in an authenticated manner Often, common system parameters are used
- **Sign** Given as input a message $m \in \{0,1\}^*$ and a private key k_{pri}, a signature σ is produced
- **Verify** Given as input a public key k_{pub}, a message m, and a signature σ, verify should only output `true` if k_{pub} and k_{pri} is a matching key pair, and σ is a valid signature for m, under this key pair

A identity based digital signature scheme consists of the following four algorithms, **Setup** and **Extract**, which are common to all identity based cryptosystems, and **Sign** and **Verify** which are common to all digital signature schemes

- **Setup** The Setup algorithm is carried out by the KGC It produces **params**, the system parameters, which are distributed to the users of the system It also produces a secret key s which is known only to the KGC This is sometimes called the master secret key

- **Extract** The Extract algorithm is carried out by the KGC, and is used to produce private keys for users in the system It takes as input **params**, s and the user identity ID and produces a private key for that user d_{ID}

- **Sign** The Sign algorithm is carried out by the end users to produce a signature on a message m It takes as input **params**, d_{ID} and the message m It outputs σ, a signature on the message m

- **Verify** The Verify algorithm takes as input **params**, σ m and ID It outputs **true** only if ID and d_{ID} is a matching key pair and σ is a signature on m, by ID

5 2 Security Definitions for Signature Schemes

5 2 1 Security of a PKI Digital Signature Scheme

Existential unforgeability under a chosen message attack for a signature scheme (KeyGen, Sign, and Verify) is defined using the following game between a challenger and an adversary $\mathcal{A}$

- **Setup** The challenger runs algorithm **KeyGen** to obtain a public key K_{pub} and private key K_{pri} The adversary $\mathcal{A}$ is given K_{pub}

- **Queries** Proceeding adaptively, $\mathcal{A}$ requests signatures with K_{pub} on at most q_s messages of his choice $\{m_1, \quad , m_{q_s}\} \in \{0,1\}^*$ The challenger responds to each query with a signature $\sigma_i = \mathrm{Sign}(K_{pri}, m_i)$

- **Output** Eventually, $\mathcal{A}$ outputs a pair (m, σ^*) and wins the game if
  - m is not any of $\{m_1, \quad , m_{q_s}\}$,

    and
  - $\mathrm{Verify}(K_{pub}, m, \sigma^*) = \mathbf{true}$

We define $Adv_{Sig}\mathcal{A}$ to be the probability that $\mathcal{A}$ wins in the above game, taken over the coin tosses[5] of **KeyGen** and of $\mathcal{A}$

5 2 2 Security of an Identity Based Digital Signature Scheme

An identity based signature scheme is said to be ***strongly existentially un-forgeable*** under chosen-message attacks if no probabilistic polynomial time (PPT) adversary has a non-negligible advantage in the following game

- The challenger runs the setup algorithm to generate the system's parameters and sends them to the adversary
- The adversary $\mathcal{F}$ performs a series of queries
  - Key extraction queries $\mathcal{F}$ produces an identity ID and receives the private key d_{ID} corresponding to ID
  - Signature queries $\mathcal{F}$ produces a message m and an identity ID and receives a signature on m that was generated by the signature oracle using the private key corresponding to the identity ID
- After a polynomial number of queries, $\mathcal{F}$ produces a tuple (ID^*, m^*, σ^*) made of an identity ID^*, whose corresponding private key was never asked during the key extraction queries, and a message- signature pair (m^*, σ^*) such that σ^* was not returned

[5]We refer to an algorithms coin tosses to denote random input into these algorithms, for example, here $\mathcal{A}$ is modeled as a probabilistic polynomial time algorithm

by the signature oracle on the input (m^*, ID^*) during the signature queries for the identity ID^*

The forger $\mathcal{F}$ wins the game if the signature verification algorithm outputs **true** when it is run on the tuple (ID^*, m^*, σ^*) The forger's advantage is defined to be its probability of producing a forgery taken over the coin tosses of the challenger and $\mathcal{F}$

5 3 The BLS Short Signature Scheme

The BLS signature scheme produces the shortest secure digital signature, and is proven secure in the random oracle model It was presented in [33] Short signatures are needed in environments where there is a strong requirement that minimum bandwidth be used For example environments where digital signatures must be typed by hand, such as provably secure product licence numbers

There are also constrained wireless devices such as those developed by the DARPA funded "Smart Dust" project[6] Generally radio communication uses much more battery power than anything else a wireless device will be required to do This means that it may be acceptable to live with high computational cost as long as the signatures produced have a minimal number of bits

The BLS short signature is aimed at addressing these problems Conventional RSA digital signatures, as they are most commonly used in industry, are 1024 bits in length For the equivalent level of security, DSA signatures are 320 bits in length The BLS signature, again for the corresponding level of security, weighs in at only 160 bits Also, signature generation is relatively fast, being just a single elliptic curve point scalar multiplication Signature verification is slightly more computationally complex as it includes a computationally expensive pairing operation

The goal of the BLS algorithm is to achieve a short signature (a signature with minimal bit length) When significantly reducing the number of bits in any security protocol it is

[6] The author was a co-researcher with a similar project at the Irish "National Centre for Sensor Research"

important to be aware of the attacks against the system See Ch 4 for an overview of some attacks against cryptographic systems These attacks tell us that, as a result of Pollard's attacks in generic discrete logarithm groups, the order of the points on the elliptic curve should be at least 2^{160} Since we also require the use of bilinear maps the embedding degree should not be too large But as a result of the MOV attack we need $r^k \approx 2^{1024}$ Therefore, if we chose to use elliptic curve groups with group order $p \geq 2^{160}$ (which is secure and yet small) we should use curves of embedding degree $k \geq 6$ [33]

The BLS signature scheme is a traditional PKI style signature scheme composed of three algorithms, **Key Generation**, **Sign** and **Verify** Here we describe an implementation of the BLS algorithm over non-supersingular curves as these curves allow for the smallest representation of the signature Over non-supersingular curves no distortion map exists, therefore, the bilinear map takes elements from two linearly independent groups The authors make use of a bilinear map of the form $e\ \mathcal{G}_1 \times \mathcal{G}_2 \to \mu_r$ They use hash functions of the form $\mathcal{H}\ \{0,1\}^* \to \mathcal{G}_1$ to hash messages onto elements of the group $\mathcal{G}_1$ In the security proof these are modelled as random oracles We assume that P_2 is a generator of $\mathcal{G}_2$, and that the order of groups $\mathcal{G}_1$ and $\mathcal{G}_2$ is r

- **KeyGen** Generate a random $x \in \mathbb{Z}_r^*$ Calculate $V = xP_2$ Have this value authenticated by a TA This is the public key of the user, with x, the private key, known only to the user

- **Sign** To sign a message m, calculate $M = \mathcal{H}(m) \in \mathcal{G}_1$ The signature of the message is $S = xM$, for the public key V This signature scheme is deterministic

- **Verify** Given the message m, the public key V, and the signature S, the signature passes the verification test if

$$e(M, V) = e(S, P_2) \tag{5 2}$$

5 3 1 Security of the BLS signature scheme

The security of the BLS short signature relies on the co-BDH assumption

- **co-Bilinear Diffie-Hellman** Given $P_1, \alpha P_1 \in \mathcal{G}_1$, and $P_2 \in \mathcal{G}_2$ for unknown α, calculate $\alpha P_2 \in \mathcal{G}_2$ This problem is assumed to be hard using groups $\mathcal{G}_1$ and $\mathcal{G}_2$

The signature scheme is proven secure in the random oracle model, assuming that the co-BDH problem is intractable The authors show how an non-negligible ability to existentially forge BLS signatures can lead to an efficient algorithm to solve the co-BDH problem

5 3 2 Efficiency of the BLS signature algorithm

The BLS signature scheme has an extremely efficient signing algorithm Note that the signing algorithm is the extract algorithm for Boneh and Franklin IBE The sign algorithm consists of just one hashing, followed by one point multiplication This seems like the minimum possible effort for a secure digital signature Using the fast hashing idea (see Sec 6 2 2), when utilising the (asymmetric) Tate pairing instead of the Weil pairing, this hashing algorithm can be made very fast by removing the need for multiplication by the curve co-factor from the hashing algorithm[7]

5.4 The Identity Based Signature Scheme of Sakai, Ohgishi and Kasahara

We now look at the very first identity based signature scheme based on bilinear maps It was proposed in 2000 by Sakai, Ohgishi and Kasahara in [111] This idea was developed around the same time as the identity based encryption scheme of Boneh and Franklin It appears that some identity based cryptosystems from pairings on elliptic curves may have originally been proposed by these Japanese researchers Their cryptosystems appeared largely without any security proofs, but the following signature scheme was subsequently proven secure by

[7]The optimisation affects both the signing and verification algorithms

Libert and Quisquater in [82] We will first look at the original scheme and then briefly at the proof by Libert and Quisquater

The original Sakai, Ohgishi and Kasahara signature scheme uses the same identity based key pair as Boneh and Franklin[8] The signature scheme consists of the four algorithms common to any identity based signature scheme They are **Setup**, **Extract**, **Sign** and **Verify**

- **Setup** The setup algorithm is carried out by the KGC It outputs two groups $\mathcal{G}$ and μ_r, both of large prime order r, such that the discrete logarithm problem in the groups $\mathcal{G}$ and μ_r is computationally infeasible It produces P, a generator of $\mathcal{G}$ It also produces two hash functions, $\mathcal{H}_{ID}$ and $\mathcal{H}_M$ of the form $\mathcal{H}_{ID} \ \{0,1\}^* \rightarrow \mathcal{G}$, and $\mathcal{H}_M \ \{0,1\}^* \rightarrow \mathcal{G}$ It also produces a bilinear map of the form $e \ \mathcal{G} \times \mathcal{G} \rightarrow \mu_r$ The KGC generates a random $s \in \mathbb{Z}_r^*$ and calculates $P_{pub} = sP$ The setup algorithm outputs **params**, where

$$\texttt{params} = \{\mathcal{G}, \mu_r, e, P, P_{pub}, \mathcal{H}_{ID}, \mathcal{H}_M\} \tag{5 3}$$

  These are published by the KGC

- **Extract** The KGC first verifies that a user has a valid claim to an identity ID The KGC then calculates $Q_{ID} = \mathcal{H}_{ID}(ID)$ This is the user's public key The associated private key is calculated as sQ_{ID}

- **Sign** To sign a message m, a user first generates a random $x \in \mathbb{Z}_r^*$ The signer also calculates $M = \mathcal{H}_M(m)$ The signer then calculates the following values

$$R = xP \tag{5 4}$$

$$S = sQ_{ID} + xM \tag{5 5}$$

[8]Note Here I am careful not to call it the Boneh and Franklin identity based key pair

The signature on the message m by signer with private key sQ_{ID}, is the pair (R, S)

- **Verify** To verify a signature that was purportedly created by a signer with public key Q_{ID}, a verifier checks the following equality

$$e(S, P) \stackrel{?}{=} e(Q_{ID}, P_{pub})\ e(M, R) \tag{5 6}$$

where $M = \mathcal{H}_M(m)$

5 4 1 Security of the SOK Identity Based Signature Scheme

The security of the SOK identity based signature was demonstrated by Libert and Quisquater in [82]

Theorem 5 4 1 *[82] In the random oracle model, if a PPT forger $\mathcal{F}$ has an advantage ϵ in forging a signature in an attack modelled by the game of Sec 5 2 for proving the security of identity based signature schemes, when running in time t and asking q_{H_i} queries to random oracles $\mathcal{H}_{ID}$ and $\mathcal{H}_M$, q_E queries to the key extraction oracle and q_s queries to the signature oracle, then the Computational Diffie Hellman problem can be solved with an advantage*

$$\epsilon' > \epsilon - \frac{(q_s(q_{H_2} + q_s) + 1)/2^k}{e(q_E + 1)} \tag{5 7}$$

within a time $t' < t + (q_{H_1} + q_{H_2} + q_E + 2q_s)t_m + (q_s + 1)t_{mm}$ where e denotes the base of natural logarithms, t_m is the time to compute a scalar multiplication in $\mathcal{G}$ and t_{mm} is the time to perform a multi-exponentiation in $\mathcal{G}$

For the proof of this theorem, the reader is referred to [82], where the authors comment on the tightness of the reduction

5 5 The Identity Based Signature Scheme of Baretto *et al.*

Barreto, Libert, McCullagh[9] and Quisquater (BLMQ) propose a new identity based signature scheme based on the identity based key pair of Sakai and Kasahara The scheme they propose is the fastest provably secure identity based signature For signing the scheme requires one $\mathbb{F}_{p^k}$ exponentiation and one point scalar multiplication This is in contrast to the scheme of Cha and Cheon [41] which requires two point scalar multiplications For the most popular commercial setting of a $k = 2$ curve this will be appreciably faster For signature verification the scheme requires one pairing computation and one pairing exponentiation We give a comparison of indicative timings in Sec 5 6 The BLMQ scheme is defined as follows

- **Setup** The KGC chooses a bilinear map e $\mathcal{G}_1 \times \mathcal{G}_2 \to \mu_r$, all of large prime order r It also selects generators $Q \in \mathcal{G}_2$, $P = \psi(Q) \in \mathcal{G}_1$ where ψ is a distortion map of the form ψ $\mathcal{G}_2 \to \mathcal{G}_1$, and $g \in \mu_r$ such that $g = e(P, Q)$ It then selects a master key $s \in \mathbb{Z}_r^*$, a system-wide public key $Q_{pub} = sQ \in \mathcal{G}_2$ and hash functions $\mathcal{H}_1$ $\{0,1\}^* \to \mathbb{Z}_r^*$, $\mathcal{H}_2$ $\{0,1\}^* \times \mu_r \to \mathbb{Z}_r^*$ The public parameters are

$$\texttt{params} = \{\mathcal{G}_1, \mathcal{G}_2, \mu_r, P, Q, Q_{pub}, e, g, \psi, \mathcal{H}_1, \mathcal{H}_2\} \tag{5 8}$$

- **KeyGen** For an identity ID, the private key is calculated as $S_{ID} = \frac{1}{\mathcal{H}_1(ID)+s}P$

- **Sign** In order to sign a message $m \in \{0,1\}^*$, the signer picks a random $x \in \mathbb{Z}_r^*$ and computes the following values

$$R = g^x \tag{5 9}$$

$$h = \mathcal{H}_2(M, R) \tag{5 10}$$

$$S = (x + h)S_{\mathsf{ID}} \tag{5 11}$$

[9] The author of this thesis

The signature on M is $\sigma = (h, S) \in \mathbb{Z}_r^* \times \mathcal{G}_1$.

- **Verify:** A signature $\sigma = (h, S)$ on a message M is accepted if the following equation holds:

$$h \stackrel{?}{=} \mathcal{H}_2(M, e(S, Q_{ID})g^{-h}) \tag{5.12}$$

where $Q_{ID} = \mathcal{H}_1(ID)Q + Q_{pub}$.

A Proof of Correctness for the BLMQ Identity Based Signature

It is easy to see that all instances of a valid signature σ will be accepted by a verifier:

$$h = \mathcal{H}_2(M, R) \tag{5.13}$$

$$h = \mathcal{H}_2(M, g^x) \tag{5.14}$$

$$h = \mathcal{H}_2(M, g^{(x+h)}g^{-h}) \tag{5.15}$$

$$h = \mathcal{H}_2(M, g^{(\frac{\mathcal{H}_1(ID)+s}{\mathcal{H}_1(ID)+s}x)+h}g^{-h}) \tag{5.16}$$

$$h = \mathcal{H}_2(M, e(S, Q_{ID})g^{-h}) \tag{5.17}$$

5.5.1 Security Proof of the BLMQ identity based signature

The security proof relies on the forking lemma [103, 104]. As the security model of IBS schemes enables a forger to adaptively choose her target identity, we cannot directly apply the forking technique and we must follow the approach of [41] that first considers a weaker attack model where adversaries are challenged on a given identity selected by the challenger. In [41], an IBS scheme is said to be secure against existential forgeries on adaptively chosen message and *given* identity attacks if no adversary has a non-negligible advantage in the weaker model of attack.

Lemma 5.5.1 ([41]). *If there is a forger $\mathcal{F}_0$ for an adaptively chosen message and identity* *attack having advantage ϵ_0 against our scheme when running in a time t_0 and making $q_{\mathcal{H}_W}$ queries to random oracle $\mathcal{H}_W$, then there exists an algorithm $\mathcal{F}_1$ for an adaptively chosen message and given identity attack which has advantage $\epsilon_1 \leq \epsilon_0(1-\frac{1}{2^k})/q_{\mathcal{H}_W}$ within a running time $t_1 \leq t_0$ Moreover, $\mathcal{F}_1$ asks the same number key extraction queries, signature queries and $\mathcal{H}_{\mu_r}$-queries as $\mathcal{F}_0$ does*

Lemma 5 5 2 *Let us assume that there is an adaptively chosen message and given identity attacker $\mathcal{F}$ that makes q_{h_i} queries to random oracles H_i ($i = 1, 2$) and q_s queries to the signing oracle Assume that, within a time t, $\mathcal{F}$ produces a forgery with probability $\epsilon \geq 10(q_s+1)(q_s+q_{\mathcal{H}_{\mu_r}})/2^k$ Then, there exists an algorithm $\mathcal{B}$ that is able to solve the q-SDH Problem for $q = q_{\mathcal{H}_W}$ in an expected time*

$$t' \leq 120686 q_{\mathcal{H}_{\mu_r}}(t + O(q_s\tau_p))/(\epsilon(1 - q/2^k)) + O(q^2\tau_{mult})$$

where τ_{mult} denotes the cost of a scalar multiplication in $\mathcal{G}_2$ and τ_p is the cost of a pairing evaluation

Proof We first show how to provide the adversary with a consistent view and we then explain how to apply the forking lemma

Algorithm $\mathcal{B}$ takes as input $(P, Q, \alpha Q, \alpha^2 Q, \quad, \alpha^q Q)$ and aims to find a pair $(c, \frac{1}{c+\alpha}P)$ In a setup phase, it builds a generator $G \in \mathcal{G}_1$ such that it knows $q-1$ pairs $(w_i, \frac{1}{w_i+\alpha}G)$ for $w_1, \quad, w_{q-1} \in_R \mathbb{Z}_p^*$ To do so,

1 It picks $w_1, w_2, \quad, w_{q-1} \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and expands $f(z) = \prod_{i=1}^{q-1}(z + w_i)$ to obtain $c_0, \quad, c_{q-1} \in \mathbb{Z}_p^*$ so that $f(z) = \sum_{i=0}^{q-1} c_i z^i$

2 It sets generators $H = \sum_{i=0}^{q-1} c_i(\alpha^i Q) = f(\alpha)Q \in \mathcal{G}_2$ and $G = \psi(H) = f(\alpha)P \in \mathcal{G}_1$ The public key $H_{pub} \in \mathcal{G}_2$ is fixed to $H_{pub} = \sum_{i=1}^{q} c_{i-1}(\alpha^i Q)$ so that $H_{pub} = \alpha H$ although $\mathcal{B}$ does not know α

3 For $1 \leq i \leq q-1$, $\mathcal{B}$ expands $f_i(z) = f(z)/(z + w_i) = \sum_{i=0}^{q-2} d_i z^i$ and

$$\sum_{i=0}^{q-2} d_i \psi(\alpha^i Q) = f_i(\alpha)P = \frac{f(\alpha)}{\alpha + w_i}P = \frac{1}{\alpha + w_i}G \tag{5 18}$$

The pairs $(w_i, \frac{1}{\alpha+w_i}G)$ are computed using the left member of (5 18)

$\mathcal{B}$ is then ready to answer $\mathcal{F}$ s queries along the course of the game It first initializes a counter ℓ to 1 and launches $\mathcal{F}$ on the input (H_{pub}, ID^*) for a randomly chosen challenge identity $\mathsf{ID}^* \xleftarrow{R} \{0,1\}^*$ For simplicity, we assume that queries to $\mathcal{H}_W$ are distinct, and that any query involving an identifier ID is preceded by the random oracle query $\mathcal{H}_W(\mathsf{ID})$

- $\mathcal{H}_W$-queries on an identity $\mathsf{ID} \in \{0,1\}^*$ $\mathcal{B}$ returns a random $w^* \xleftarrow{R} \mathbb{Z}_p^*$ if $\mathsf{ID} = \mathsf{ID}^*$ Otherwise, $\mathcal{B}$ answers $w = w_\ell \in \mathbb{Z}_p^*$ and increments ℓ In both cases, $\mathcal{B}$ stores (ID, w) (where $w^* = w$ or w_ℓ) in a list L_1

- Key extraction queries on $ID \neq ID^*$ $\mathcal{B}$ recovers the matching pair (ID, w) from L_1 and returns the previously computed $(1/(\alpha + w))G$

- Signature query on a message-identity pair (M, ID) $\mathcal{B}$ picks $S \xleftarrow{R} \mathcal{G}_1$, $h \xleftarrow{R} \mathbb{Z}_p^*$, computes $r = e(S\ Q_{ID})e(G\ H)^{-h}$ where $Q_{ID} = \mathcal{H}_W(ID)H + H_{pub}$ and backpatches to define the value $\mathcal{H}_{\mu_r}(M, r)$ as $h \in \mathbb{Z}_p^*$ ($\mathcal{B}$ aborts in the unlikely event that $\mathcal{H}_{\mu_r}(M, r)$ is already defined)

We have explained how to simulate $\mathcal{F}$'s environment in a chosen-message and given identity attack We are ready to apply the forking lemma that essentially says the following consider a scheme producing signatures of the form (M, r, h, S), where each of r, h, S corresponds to one of the three moves of a honest-verifier zero-knowledge protocol Let us assume that a chosen-message attacker $\mathcal{F}$ forges a signature (M, r, h, S) in a time t with probability $\epsilon \geq 10(q_s + 1)(q_s + q_h)/2^k$ (k being a security parameter chosen so that h is uniformly taken from a set of 2^k elements) when making q_s signature queries and q_h random oracle calls If the triples (r, h, S) can be simulated without knowing the private key, then there exists a Turing machine $\mathcal{F}'$ that uses $\mathcal{F}$ to produce two valid signatures $(m, r, \mathcal{H}_W, S_1)$, $(m, r, \mathcal{H}_{\mu_r}, S_2)$, with $\mathcal{H}_W \neq \mathcal{H}_{\mu_r}$, in expected time $t' \leq 120686 q_h t/\epsilon$

In our setting, from a forger $\mathcal{F}$, we build an algorithm $\mathcal{F}'$ that replays $\mathcal{F}$ a sufficient number of times on the input (H_{pub}, ID^*) to obtain two suitable forgeries $\langle M^*, r, \mathcal{H}_W, S_1\rangle$, $\langle M^*, r, \mathcal{H}_{\mu_r}, S_2\rangle$ with $\mathcal{H}_W \neq \mathcal{H}_{\mu_r}$

The reduction then works as follows The simulator $\mathcal{B}$ runs $\mathcal{F}'$ to obtain two forgeries $\langle M^*, r, \mathcal{H}_W, S_1\rangle$, $\langle M^*, r, \mathcal{H}_{\mu_r}, S_2\rangle$ for the same message M^* and commitment r At this stage, $\mathcal{B}$ recovers the pair (ID^*, w^*) from list L_1 We note that $w^* \neq w_1, \quad , w_{q-1}$ with probability at least $1 - q/2^k$ If both forgeries satisfy the verification equation, we obtain the relations

$$e(S_1, Q_{ID^*})e(G,H)^{-\mathcal{H}_W} = e(S_2, Q_{ID^*})e(G,H)^{-\mathcal{H}_{\mu_r}},$$

with $Q_{ID^*} = \mathcal{H}_W(\mathsf{ID}^*)H + H_{pub} = (w^* + \alpha)H$ Then, it comes that

$$e((\mathcal{H}_W - \mathcal{H}_{\mu_r})^{-1}(S_1 - S_2), Q_{ID^*}) = e(G,H),$$

and hence $T^* = (\mathcal{H}_W - \mathcal{H}_{\mu_r})^{-1}(S_1 - S_2) = \frac{1}{w^* + \alpha}G$ From T^*, $\mathcal{B}$ can proceed as in [28] to extract $\sigma^* = \frac{1}{w^* + \alpha}P$ it first obtains $\gamma_{-1}, \gamma_0 \quad , \gamma_{q-2} \in \mathbb{Z}_p^*$ for which $f(z)/(z + w^*) = \gamma_{-1}/(z + w^*) + \sum_{i=0}^{q-2} \gamma_i z^i$ and eventually computes

$$\sigma^* = \left[T^* - \sum_{i=0}^{q-2} \gamma_i \psi(\alpha^i Q)\right]^{1/\gamma_{-1}} = \frac{1}{w^* + \alpha}P$$

before returning the pair (w^*, σ^*) as a result

It finally comes that, if $\mathcal{F}$ forges a signature in a time t with probability $\epsilon \geq 10(q_s + 1)(q_s + q_{\mathcal{H}_{\mu_r}})/2^k$, $\mathcal{B}$ solves the q-SDH Problem in expected time

$$t' \leq 120686 q_{\mathcal{H}_{\mu_r}}(t + O(q_s \tau_p))/(\epsilon(1 - q/2^k)) + O(q^2 \tau_{mult})$$

where the last term accounts for the cost of the preparation phase $\square$

5 6 Conclusion

In this chapter we have reviewed some important signature schemes that use bilinear maps We have seen that bilinear maps, although famous for their use in identity based cryptography, can make significant contributions to traditional public key cryptosystems Bilinear maps allow for secure signature schemes where the signature is approximately 160 bits in length This is approximately half the size of the previous shortest signature scheme

As we have seen already, bilinear maps have been the enabling tool behind efficient identity based encryption We have tracked the progress of identity based signature schemes We give a table of the comparative performance of the different signature schemes below and note that the author has been involved in the design of the fastest identity based digital signature This new, fast, identity-based signature is based on the identity-based key pair proposed by Sakai and Kasahara The timing comparisons in Table 5 1 do not take into account generation of the signer's public key from their identity

	Sign				Verify			
signature scheme	exp	mul	pairings	time (ms)	exp	mul	pairings	time (ms)
SOK	0	2	0	188	0	0	3	516
Paterson	0	4	0	376	2	0	2	354
Cha-Cheon	0	2	0	188	0	1	2	438
Hess	1	2	0	193	1	0	2	349
$SK_{(ElGamal)}$	0	3	0	282	0	2	2	532
$SK_{(Schnorr)}$	1	2	0	192	0	1	2	438
BLMQ (ours)	1	1	0	**99**	1	0	1	**177**

Table 5 1 Efficiency comparison of identity based signature schemes

The timings indicated in Table 5 1 were performed on an Athlon 64 3000+ processor, with 512MB ram and using the Java 2 Platform Standard Edition 5 0 run time environment

Some schemes can benefit from pre-computation in the verification stage We note here that ours cannot However, even when competing against schemes with pre-computation our scheme still matches the most efficient, with the added bonus that our scheme does not require any storage The competing schemes require $\mu_{r_b} n$ bits, where n is the number of users in the system with which we communicate regularly, and μ_{r_b} is the number of bits required to store an element of μ_r A comparison of timings with precomputation taken into account can be found in Appendix C

Chapter 6

Encryption Systems using Bilinear Maps

There were three papers in the early development of pairing based cryptography, that awakened cryptographer's interest in bilinear maps Firstly, there was the paper by Menezes, Okamoto and Vanstone which described an attack using the Weil pairing to efficiently convert the elliptic curve discrete logarithm problem to a discrete logarithm problem in a finite field [92] This was important, because, although the resulting finite field is larger than the original elliptic curve group, this allows the attacker to use index calculus methods to attack the EC DLP This is a destructive use of bilinear maps and it revealed that certain elliptic curves were not as secure as once thought

The second fundamental paper was by Joux [77] It was the first paper that used pairings constructively in cryptography This paper used the bilinearity of the pairing to include an extra entity in a Diffie-Hellman Key agreement Each party paired the contributions of the other two parties They then exponentiated the resulting pairing by their secret value This protocol was not without its problems It is essentially an unauthenticated three party Diffie-Hellman key agreement and as such is still subject to the 'man-in-the-middle" attack

The third seminal paper, by Boneh and Franklin [31], was the spark that really got cryptographers interested in bilinear maps It closed a long standing open problem in cryptography. The problem of constructing an efficient, secure identity based encryption (IBE) scheme had been proposed by Shamir in 1984 [118]. In his paper, Shamir proposed the first identity based signature scheme, but left the construction of identity based encryption schemes as an open problem. Seventeen years later, in 2001, an efficient solution was finally proposed by Boneh and Franklin. This solution made use of bilinear maps. The idea behind an identity based encryption scheme is that a user's online identity is used to encrypt information to them. An identity based cryptosystem (IBC) makes use of a Key Generation Centre (KGC). This substantially reduces the problems associated with key binding (certificates) in traditional PKI systems.

Since the Boneh and Franklin IBE scheme there have been many encryption schemes devised which make use of bilinear maps. Another example in their seminal paper was an escrowed El Gamal encryption scheme, which was somewhat lost in the shadow of IBE. Other examples include certificateless public key encryption [4, 49], public key encryption with keyword search [30, 9], broadcast encryption [34], hierarchical IBE [70, 76, 37, 55, 29], policy based encryption etc [3]. There are also some identity based encryption schemes that are proven secure in the standard model [1], see for example [27, 26].

NB: Around the same time as the Boneh and Franklin discovery there was concurrent research in this area by Sakai, Ohgishi and Kasahara [111] who described the first identity based key agreement protocols and signature schemes based on bilinear maps. However this research was not generally known to western researchers until after the publication of the Boneh-Franklin paper.

6.1 Identity Based Encryption

An identity-based encryption scheme E is specified by four randomized algorithms: **Setup**, **Extract**, **Encrypt** and **Decrypt**:

- **Setup:** takes as input a security parameter k. It outputs **params** (the system pa-

[1] i.e. without using random oracles in the security proofs.

rameters) and a master-key The system parameters include a description of a finite message space M, and a description of a finite ciphertext space C The system parameters will be publicly known, while the master-key will be known only to the KGC

- **Extract** takes as input **params**, the master-key, and an arbitrary $ID \in \{0,1\}^*$, and outputs a private key d Here ID is an arbitrary string that will be used as a public key, and d is the corresponding private decryption key The Extract algorithm extracts a private key from the given public key

- **Encrypt** takes as input **params**, ID, and $m \in M$ It outputs a ciphertext $c \in C$

- **Decrypt** takes as input params, $c \in C$ and a private key d It outputs $m \in M$ or, if the decryption fails, $\perp$

6 1 1 Security Definition for Identity Based Encryption

Chosen ciphertext security (IND-CCA) is the standard notion of security for a public key encryption scheme Hence, it is natural to require that an identity-based encryption scheme also satisfy this strong notion of security However, the definition of chosen ciphertext security must be strengthened a bit The reason is that when an adversary attacks a public key ID in an identity-based system, the adversary might already possess the private keys of users $\{ID_0, \quad, ID_n\} \mid ID \notin \{ID_0, \quad, ID_n\}$ of her choice The system should remain secure under such an attack Hence, the definition of chosen ciphertext security must allow the adversary to obtain the private key associated with any identity ID_i of her choice (other than the public key ID being attacked) We refer to such queries as private key extraction queries Another difference is that the adversary is challenged on a public key ID of her choice (as opposed to a random public key)

We say that an identity-based encryption scheme E is semantically secure against an adaptive chosen ciphertext attack (IND-ID-CCA) if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the Challenger in the following IND-ID-CCA game

- **Setup** The challenger takes a security parameter k and runs the Setup algorithm It gives the adversary the resulting system parameters **params** It keeps the master-key secret

- **Phase 1** The adversary issues queries $\{q_1, \quad, q_m\}$ where query q_i is one of

  - Extraction query $\langle ID_i \rangle$ The challenger responds by running algorithm **Extract** to generate the private key d_i corresponding to the public key $\langle ID_i \rangle$ It sends d_i to the adversary

  - Decryption query $\langle ID_i, C_i \rangle$ The challenger responds by running algorithm Extract to generate the private key d_i corresponding to ID_i It then runs algorithm Decrypt to decrypt the ciphertext C_i using the private key d_i It sends the resulting plaintext to the adversary

  These queries may be asked adaptively, that is, each query q_i may depend on the replies to $\{q_1, \quad, q_{i-1}\}$

- **Challenge** Once the adversary decides that Phase 1 is over it outputs two equal length plaintexts $\{m_0, m_1\} \in M$ and an identity ID^* on which it wishes to be challenged The only constraint is that ID^* did not appear in any private key extraction query in Phase 1 The challenger picks a random bit $b \in \{0,1\}$ and sets $C^* =$ Encrypt(**params**, ID^*, m_b) It sends C^* as the challenge to the adversary

- **Phase 2** The adversary issues more queries $\{q_{m+1}, \quad, q_n\}$ where query q_i is one of

  - Extraction query $\langle ID_i \rangle$ where $ID_i \neq ID^*$ Challenger responds as in Phase 1
  - Decryption query $\langle ID_i, C_i \rangle \neq \langle ID^*, C^* \rangle$ Challenger responds as in Phase 1

  These queries may be asked adaptively as in Phase 1

- **Guess** Finally, the adversary outputs a guess $b' \in \{0,1\}$ and wins the game if $b' = b$

We refer to such an adversary $\mathcal{A}$ as an IND-ID-CCA adversary. We define adversary $\mathcal{A}$'s advantage in attacking the scheme E as the following function of the security parameter k (k is given as input to the challenger):

$$Adv_{\mathcal{A}}(k) = |Pr[b' = b] - 1/2|. \tag{6.1}$$

The probability is over the random bits used by the challenger and the adversary.

6.2 Boneh and Franklin's Identity Based Encryption Scheme

Boneh and Franklin's IBE system consists of the following four algorithms: **Setup** and **Extract** which are performed by the KGC, and **Encrypt** and **Decrypt** which are performed by the clients.

Boneh and Franklin's identity based key pair generation algorithms, Setup and Extract, have been used by many identity based cryptosystems, such as those in [41, 46, 81, 86, 114, 127]. It is the first[2] of the two identity based key pair derivation algorithms for IBC systems based on bilinear maps, the other being from Sakai and Kasahara.

- **Setup:** The setup algorithm is carried out by the KGC. It takes a security parameter k, and outputs two groups $\mathcal{G}$ and μ_r, both of large prime order r, such that the discrete logarithm problem in the groups $\mathcal{G}$ and μ_r is computationally infeasible. The KGC produces P, a generator of $\mathcal{G}$, four hash functions; $\mathcal{H}_{ID}$ of the form $\mathcal{H}_{ID} : \{0,1\}^* \rightarrow \mathcal{G}$, $\mathcal{H}_{\mu_r}$ of the form $\mathcal{H}_{\mu_r} : \mu_r \rightarrow \{0,1\}^n$, $\mathcal{H}_r$ of the form $\mathcal{H}_r : \{0,1\}^n \times \{0,1\}^* \rightarrow \mathbb{Z}_r^*$ and $\mathcal{H}_v$ of the form $\mathcal{H}_v : \{0,1\}^n \rightarrow \{0,1\}^n$. It also produces a bilinear map of the form $e : \mathcal{G} \times \mathcal{G} \rightarrow \mu_r$. The KGC generates a random secret $s \in \mathbb{Z}_r^*$ and calculates $P_{pub} = sP$. The setup algorithm outputs `params`, where

[2] Earlier work by Sakai, Ohgishi and Kasahara, used this same IBC key pair, but it was unknown by western researchers until later, see [111], consequently this IBC key pair has become known as the Boneh and Franklin key pair.

$$\textsf{params} = \{\mathcal{G}, \mu_r, e, P, P_{pub}, \mathcal{H}_{ID}, \mathcal{H}_{\mu_r}, \mathcal{H}_r, \mathcal{H}_v\} \tag{6 2}$$

The KGC publishes **params**

- **Extract** The KGC first verifies that a user has a valid claim to an online identity ID The KGC then calculates $Q_{ID} = \mathcal{H}_{ID}(ID)$ This is the user's public key The associated private key is calculated as sQ_{ID}

- **Encrypt** A user encrypts a message $m \in \{0,1\}^*$ to a recipient with identity ID and private key sQ_{ID} using the following probabilistic encryption algorithm

  Choose a random $\sigma \in \{0,1\}^n$ and compute the following values

$$x = \mathcal{H}_r(\sigma, m) \tag{6 3}$$

$$R = xP \tag{6 4}$$

$$g_{ID} = e(P_{pub}, Q_{ID}) \tag{6 5}$$

$$\mathcal{M} = \mathcal{H}_{\mu_r}(g_{ID}^x) \tag{6 6}$$

$$V = \mathcal{M} \oplus \sigma \tag{6 7}$$

$$C = \mathcal{H}_v(\sigma) \oplus m \tag{6 8}$$

  The resulting ciphertext is (R, C, V) It should be noted that at this stage g_{ID} will not change for repeated encryptions to the same identity ID It is therefore advisable, if storage limitations permit, to compute and cache the value g_{ID}

- **Decrypt** A user with private key sQ_{ID}, who receives a ciphertext (R, C, V) intended for him, calculates the following values to recover the message m The receiver first checks that $R \in \mathcal{G}^3$, then the user computes the following values

[3]As Scott points out [116] this is "free" when computing the pairing operation, if not using the private key as a BKLS fixed base Using R as the first argument of the pairing implicitly performs a rP multiplication To check membership of $\mathcal{G}$, simply check that $rP = \mathcal{O}$

$$\mathcal{M} = \mathcal{H}_{\mu_r}(e(R, sQ_{ID})) \quad (6\ 9)$$

$$\sigma = V \oplus \mathcal{M} \quad (6\ 10)$$

$$m = \mathcal{H}_v(\sigma) \oplus C \quad (6\ 11)$$

$$x' = \mathcal{H}_r(\sigma, m) \quad (6\ 12)$$

And performs the following check

$$R \stackrel{?}{=} x'P \quad (6\ 13)$$

If the above check holds then the ciphertext is accepted as being valid, otherwise the ciphertext is rejected

6 2 1 The Security of Boneh and Franklin's IBE scheme

The security of the Boneh and Franklin scheme rests on the difficulty of the BDH problem Though we will not go into the detail of the security arguments here, we note that an identity based system needs a new type of security model Boneh and Franklin address this issue by constructing the security proof in two parts

1 Construct a public key encryption scheme from an identity based encryption scheme by providing a fixed identity as part of the system parameters[4] Prove the security of this scheme

2 Show how an advantage in breaking the equivalent identity based scheme can be transformed into an advantage in breaking the public key encryption scheme

[4]This is called BasicPub in [31]

6 2 2 Implementational Improvements to Boneh and Franklin's IBE

As part of my Ph D work I have implemented several identity based cryptosystems[5] I now note two improvements that I have observed Both of these ideas are of implementational importance, and can significantly reduce the time taken to perform IBE However, they are not substantial enough to warrant papers in themselves One of these has been published as a small section of an CT-RSA paper by Scott[6] [116], with reference to a personal communication The other idea remains unpublished

McCullagh's Observation on the Boneh and Franklin Key Pair Derivation Algorithm

It is often reported in literature that, when doing identity based encryption, the most computationally expensive process is actually computing the pairing When we implemented the Boneh and Franklin IBE system on a mobile phone we inserted many timing logs into the program so we could identify the bottlenecks Somewhat to our surprise[7] we discovered that public key generation from an identity took twice as long as a pairing calculation We then looked closely at the structure of the public key

The 'Map To Point" algorithm of Boneh and Franklin mandates that the public key is generated as follows

$$y = \mathcal{H}_{\bar{ID}}(id) \quad (6\ 14)$$

$$Q'_{pub} = (x, y) \in E \quad (6\ 15)$$

$$Q_{pub} = lQ'_{pub} \quad (6\ 16)$$

This algorithm has three steps

1 Hashing, to produce an integer $y \in \mathbb{Z}_r^*$, which is relatively efficient

[5]Using the programming languages Java, C and C++
[6]The author's Ph D supervisor
[7]In the literature, pairing is alway is always mooted at the computationally expensive operation

2. Solving the curve equation to obtain a point on the curve of unknown order. Again, this is reasonably efficient and, in the case of the curve recommended in [31] is deterministic, so should run in a reasonably quick time.

3. Multiplication of the point by an element l. In the case of the curves used in [31] $l = (p+1)/r$. Therefore, in the popular setting of $p = 512, k = 2$, which seems to be gaining favour as the curve specification to implement, this is $\approx 2^{512}/2^{160}$. Therefore l is a 352 bit number. This is substantially larger than the usual 160 bit integers that we associate with point scalar multiplication. This is obviously the bottleneck.

The reason for multiplication by l is to ensure the point is of order r. When working with the Weil pairing both points must be of order r. Boneh and Franklin's paper concentrated on the use of the Weil pairing. However, it soon became apparent that the reduced Tate pairing provided much better performance than the Weil pairing. The first commercial applications are using the Tate pairing in place of the Weil pairing. However, they have kept the public key generation algorithm unchanged[8].

Since the Tate pairing is preferred, the pairing need no longer be symmetric. Only the first argument of the pairing must be a point of order r. Therefore, as noted by Scott, the public and private keys, if they are used as the second argument to the pairing, need no longer be points of order r [115] – we can use Q'_{pub} in place of Q_{pub}. Unfortunately, with Scott's fast key pair generation method we have lost compatibility with the Boneh and Franklin key server[9]. This is a problem in the commercial world, if not in the academic world. Ideally we wish to use Scott's optimisation, whilst still being Boneh and Franklin IBE "standards compliant".

If we look at the use of the public key we see that Boneh and Franklin encryption comes down to the equality:

$$e(xsP, Q_{pub}) = e(xP, Q_{pri}) \tag{6.17}$$

[8] See `http://www.voltage.com`.

[9] Such a Key Server is used by `http://www.voltage.com` and has, as a result, become the *de facto* commercial standard.

where the left hand side is the basis of the sender's computation, and the right hand side is the basis of the recipient's computation

Expanding this equation a little we have

$$e(xsP\ Q_{pub}) = e(xsP, lQ'_{pub}) \quad (6\ 18)$$

$$e(lxsP, Q'_{pub}) = e(xP, Q_{pri}) \quad (6\ 19)$$

Therefore, we can simply replace the point sP with the point lsP This is the co-factor multiplication that we identified as the bottleneck above, however this only needs to be done once, and so can be amortized over the lifetime of the system Indeed, this new value can be distributed with the system parameters and so need not be calculated by the client at all This very small change results in approximately 10 to 20 times faster public key generation on the client[10], see Table 6 1, whilst maintaining full compliance with the Key Server[11]

Boneh and Franklin's hash and map	The faster Boneh and Franklin compliant hash and map
328ms	16ms

Table 6 1 Timings for Java Implementation

The code used for this test is available in Appendix B

McCullagh's Observation on Boneh and Franklin Private Key Distribution for Low Powered Constrained Devices

As should already be apparent, a user in an identity based cryptosystem does not have any PKI certificate The assurances in a PKI come from the fact that a Certificate Authority (CA) has publicly certified that a user is linked to a particular public key We expect the CA to perform appropriate checks when certifying that a public key belongs to a user

[10] Approx 20 times faster in Java code, on AMD 64 3000+
[11] The Key Server continues to issue the same keys as before

Similarly a user in an identity based system is certified by a KGC We assume that the fact that a user has a private key implies that user has been authenticated by the KGC A user should not be able to generate a private key themselves

As part of my research, I was a member of a team which implemented identity based cryptographic solutions on very restricted devices Whilst it is quite common for high-end mobile phone platforms to support SSL, we were interested in developing a system for issuing private keys that requires very low bandwidth Ideally this solution should be restricted to the set of operations that is inherently needed to perform identity based encryption (on elliptic curves), so as to shrink the size and power consumption of the processor We looked at the possibility of using an SSL scheme which specified elliptic curve El Gamal, but we have come up with a slightly more streamlined solution which uses less computation and about half the bits of elliptic curve El Gamal – even before the excess overhead of SSL is removed The main performance improvements come from the observation that a Boneh-Franklin private key is a BLS signature by the KGC on the client's identity BLS signatures are explained in more detail in Ch 5

When developing for wireless devices, such as sensor networks, it is important that the absolute minimum number of bits is transmitted, since radio is the most power hungry resource on these devices Battery life can be dramatically increased if the use of radio is minimised

We note that previous work has been done in this area by Lee *et al* but in their scheme the end user is not verified by the KGC [12] A more complex multi-KGC variant that they propose, was, on another level, broken by Chunxiang *et al* in [56] There is also previous work by Sui *et al* [130] However, their scheme is computationally more complex and requires twice the bandwidth from user to KGC than our proposed solution It also uses a password as opposed to a digital signature for authentication We note that unlike these preceeding schemes our scheme is not annoymous An eavesdropper can determine the origin and authenticity of each message in the protocol

[12] It is assumed that the client uses some seperate means to verify themselves

A user, at registration time, has a long term public key that is given to the KGC in an authenticated manner This is an El Gamal public key, based on the same elliptic curve E and generator point P as specified by the KGC for use with pairings The key pair is of the form $\{x, xP\}$ where the integer $x \in \mathbb{Z}_r^*$ is the private component and xP is the public component

Likewise, the KGC has a similar key pair, $\{s, P_{pub} = sP\}$, were P_{pub} is the master public key as distributed in the IBE `params` At each time period, every user in the system should be able to generate every identity in the system, this being the fundamental point of an identity based cryptosystem It should be noted that a user is able to generate their own public key for the next time period, using whatever rules the KGC has set out

Our key issuing protocol is shown in Table 6 2

Client		KGC
$Q_{ID} \leftarrow \mathcal{H}_{ID}(\text{identity} \parallel \text{time period})$		
$V = xQ_{ID}$	$\rightarrow$	
		verify BLS(ID, xP, V, P)
	$\leftarrow$	$S = sV$ (which is sxQ_{ID})
$Q'_{ID} = x^{-1}S$		
verify BLS(ID, Q'_{ID}, P, sP)		
$Q_{ID_{pri}} = Q'_{ID}$		

Table 6 2 An Efficient Protocol for Private Key Distribution

Where verify BLS means simply to run the BLS verification algorithm on the inputs

Heuristic Security Arguments for the Security of the Key Distribution Protocol The key issuing protocol exploits the fact that both the client and the KGC can generate the public key for the next period solely from knowledge of the ID and the public key construction algorithm[13] as defined by the KGC The request for a new key starts with a BLS signature on the identity using the client's long term PKI key pair which has been authenticated by the KGC This ensures the authenticity of the claimant each time a new key is issued Any non-negligible ability by an adversary to fool the KGC at this stage

[13]Called "map to point" in Boneh and Franklin's IBE

implies an ability to forge BLS signatures

If the BLS signature passes the verification stage, then the KGC uses its public point P_{pub} as a regular PKI public key - it is, after all, a valid EC El Gamal public key It then BLS signs the value that was given to it by the client At this stage, the resulting value can be viewed as a blinded BLS signature by the KGC on the identity's public key This blinding is important, since the BLS signature by the KGC on the identity is the private key

The client, who knows the value x, can, at the last stage, unblind the signature By doing this it will obtain the client's private key (or the KGC's signature on the identity)

An eavesdropper can obviously check the validity of the messages that are being sent back and forth as they are just signatures on known messages by known entities (actually, this depends on whether or not the user's public key is made truly public or just known to the KGC)

The BLS check by the client at the end of the protocol ensures that they have received the valid private key This check is important to ensure that an adversary does not inject a false value for the private key into the protocol

6 3 Sakai and Kasahara's Identity Based Encryption Scheme

The original Sakai and Kasahara scheme was an 'ID based public key cryptosystem with Authentication" described in [110, Sec 3] This is effectively a signcryption scheme Whilst both the signature scheme and the encryption scheme appear secure (the authors did not present proofs of security), there is a problem with the way that they aggregate the encryption and signature schemes, as pointed out by McCullagh and Barreto [88], which is an adaptation of an attack by Libert and Quisquater [81] on Malone-Lee's Signcryption scheme [86] This does not detract from the importance of the Sakai and Kasahara IBE scheme We will just look at the encryption scheme here

The encryption scheme defined here, which was never formally defined by Sakai and Kasahara, is their basic scheme with the Fujisaki-Okamoto transform [66] applied This is the same mechanism by which Boneh and Franklin transformed their 'Basic Ident" scheme into their 'Full Ident" scheme in [31] Chen and Cheng have recently proved the security of this scheme in [43]

- **Setup** The setup algorithm is carried out by the KGC It takes a security parameter k, and outputs two groups $\mathcal{G}$ and μ_r, both of large prime order r, such that the discrete logarithm problem in the groups $\mathcal{G}$ and μ_r is computationally infeasible The KGC produces P, a generator of $\mathcal{G}$, g, a generator of μ_r, such that $g = e(P, P)$, four hash functions, $\mathcal{H}_{ID}$ of the form $\mathcal{H}_{ID} \ \{0,1\}^* \to \mathbb{Z}_r^*$, $\mathcal{H}_{\mu_r}$ of the form $\mathcal{H}_{\mu_r} \ \mu_r \to \{0,1\}^n$, $\mathcal{H}_r$ of the form $\mathcal{H}_r \ \{0,1\}^n \times \{0,1\}^* \to \mathbb{Z}_r^*$ and $\mathcal{H}_v$ of the form $\mathcal{H}_v \ \{0,1\}^n \to \{0,1\}^n$ It also produces a bilinear map of the form $e \ \mathcal{G} \times \mathcal{G} \to \mu_r$ The KGC generates a random secret $s \in \mathbb{Z}_r^*$ and calculates $P_{pub} = sP$ The setup algorithm outputs **params**, where

$$\textbf{params} = \{\mathcal{G}, \mu_r, e, P, P_{pub}, \mathcal{H}_{ID}, \mathcal{H}_{\mu_r}, \mathcal{H}_r, \mathcal{H}_v\} \qquad (6\ 20)$$

- **Extract** To generate a private key for a client of the system, the KGC verifies the end user is entitled to a particular online identity, $ID \in \{0,1\}^*$, and generates the user's key pair, first by calculating $\mathcal{H}_{ID}(ID) \to a \in \mathbb{Z}_r^*$, and then computing the user's public key as $sP + aP = (s + a)P$, whilst the user's private key is $(s + a)^{-1}P$

- **Encrypt** To encrypt a message $m \in \{0,1\}^*$, to a user with identity ID, a user generates a random $\sigma \in \{0,1\}^n$ and calculates the following values

$$x = \mathcal{H}_r(\sigma, m) \tag{6 21}$$

$$R = x(s + \mathcal{H}_{ID}(ID))P \tag{6 22}$$

$$\mathcal{M} = g^x \tag{6 23}$$

$$S = \sigma \oplus \mathcal{H}_{\mu_r}(M) \tag{6 24}$$

$$C = m \oplus \mathcal{H}_v(\sigma) \tag{6 25}$$

The ciphertext is the tuple (R, S, C) It should be noted at this stage that encryption does not require a pairing calculation and so is more efficient than the identity based encryption scheme proposed by Boneh and Franklin

- **Decrypt** To decrypt a ciphertext (R, S, C), a user with private key $(s + a)^{-1}P$ computes the following values

$$\mathcal{M} = e(R, (s + a)^{-1}P) \tag{6 26}$$

$$\sigma = S \oplus \mathcal{H}_{\mu_r}(\mathcal{M}) \tag{6 27}$$

$$m = C \oplus \mathcal{H}_v(\sigma) \tag{6 28}$$

$$x' = \mathcal{H}_r(\sigma, m) \tag{6 29}$$

And check if the following test holds

$$x'P \stackrel{?}{=} R \tag{6 30}$$

The ciphertext is accepted if the equality above holds, otherwise the ciphertext is rejected

This IBE scheme is attracting a lot of attention from both the academic and industrial communities not only because it is more efficient than the Boneh and Franklin scheme, but also for commercial reasons[14]

6 4 Public Key Encryption with Keyword Search

The idea of Public key Encryption with Keyword Search (PEKS), which was introduced by Boneh *et al* in [30] is that a specified user, who might not ordinarily be allowed to read encrypted data, is able to test if a specific word is present in the data This encryption scheme is based on public key encryption methods and so is not applicable to large volumes of data, but may be appropriate for encrypting small amounts of data such as email headers The example, given by the authors, was to alert a largely untrusted email gateway to forward messages that were marked urgent (for example to a BlackBerry device), whilst not allowing the device to read any of the encrypted message Another example may be to allow clerks in the military to effectively handle data which is classified above their security clearance Private decryption keys can be tailored to allow for the searching of any particular word, and only that word Obviously PEKS schemes must resist dictionary attacks

6 4 1 Definition of a Public Key Encryption with Keyword Search Scheme

In a PEKS scheme "public key" refers to the fact that ciphertexts are created by various people using Alice's public key, in the same way as a normal public key encryption scheme Suppose Bob wants to send an encrypted message m to Alice with the keywords $\{W_1, \quad, W_k\}$ (we assume that k is small) Bob then sends the following message

$$[E_{A_{pub}}[m], PEKS(A_{pub}, W_1), \quad, PEKS(A_{pub}, W_k)] \tag{6 31}$$

where A_{pub} is Alice's public key and m is the email body We assume that this information is to pass though a mail gateway that is trusted to redirect messages containing specific keywords, but which otherwise is not authorised to see the message

[14] The Boneh and Franklin scheme is subject to patent protection, owned by Stanford University and Voltage Security Inc, a Stanford University startup company

The goal of a PEKS scheme is to enable Alice to send a short secret key (a k a a trapdoor) T_W to the mail gateway that will enable the gateway to locate all messages containing the keyword W, but learn nothing else about the messages Alice produces this trapdoor T_W using her private key The server simply sends the relevant emails back to Alice Such a scheme is called a non-interactive public key encryption with keyword search, or as a shorthand, a 'searchable public-key encryption"

A PEKS scheme consists of the following algorithms

1 **KeyGen(s)** Takes a security parameter, k, and generates a public/private key pair (A_{pub}, A_{pri})

2 **PEKS**(A_{pub}, W) For a public key A_{pub} and a word W, produces a searchable encryption of W

3 **Trapdoor**(A_{pri}, W) Given Alice's private key and a word W produces a trapdoor T_W

4 **Test**(A_{pub}, S, T_W) Given Alice's public key, a searchable encryption S = PEKS(A_{pub}, W_0), and a trapdoor T_W = Trapdoor(A_{pri}, W), outputs `true` iff $W = W_0$ and $\perp$ otherwise

Alice runs the KeyGen algorithm In typical PKI fashion she publishes her public key and keeps her private key secret It is assumed that all users in the system have access to an authenticated copy of Alice's public key With knowledge of A_{pri} and her choice of word W, she uses the algorithm Trapdoor to produce T_W, a trapdoor corresponding to to her public key and the word W T_W is then given to the third party (in this case the email gateway) The gateway can now check for the existence of the word W in a given message

An important point is that $PEKS(A_{pub}, W)$ must not reveal any information about the existence of the keyword W unless T_W is available

6 4 2 The security model for PEKS schemes

We define security against an active attacker who is able to obtain trapdoors T_W for any W of his choice Even under such attack the attacker should not be able to distinguish an encryption of a keyword W_0 from an encryption of a keyword W_1 for which he did not obtain the trapdoor Formally, we define security against an active attacker $\mathcal{A}$ using the following game between a challenger and the attacker

PEKS Security game

1 The challenger runs the KeyGen(k) algorithm to generate A_{pub} and A_{priv} It gives A_{pub} to the attacker

2 The attacker can adaptively ask the challenger for the trapdoor T_W for any keyword $W \in \{0,1\}^*$ of his choice

3 At some point, the attacker $\mathcal{A}$ sends the challenger two words W_0, W_1 on which it wishes to be challenged The only restriction is that the attacker did not previously ask for the trapdoors T_{W_0} or T_{W_1} The challenger picks a random $b \in \{0,1\}$ and gives the attacker $C = \text{PEKS}(A_{pub}, W_b)$ We refer to C as the challenge PEKS

4 The attacker can continue to ask for trapdoors T_W for any keyword W of his choice as long as $W \neq W_0, W_1$

5 Eventually, the attacker $\mathcal{A}$ outputs $b' \in \{0,1\}$ and wins the game if $b = b'$ We define $\mathcal{A}'s$ advantage in breaking the PEKS as

$$Adv_{\mathcal{A}}(s) = |Pr[b = b'] - 1/2| \tag{6 32}$$

Definition A PEKS is semantically secure against an adaptive chosen keyword attack if for any polynomial time attacker $\mathcal{A}$ we have that $Adv_{\mathcal{A}}(s)$ is negligible

6 4 3 Boneh *et al*'s Public Key Encryption with Keyword Search Scheme

The original PEKS scheme was proposed by Boneh *et al* in [30] This scheme exploits the fact that in IBE cryptosystems identities are, after all, only words Therefore the authors of [30] observed that they could create a PEKS scheme from the Boneh and Franklin IBE scheme In fact, as subsequently pointed out in [2], we can adapt any anonymous[15] IBE scheme, by replacing the identity with a keyword The transformation is more complex, but this is the basic idea

In Boneh *et al*'s scheme the length of the ciphertext of the PEKS increments with each key word appended It is assumed that PEKS will be used as part of a hybrid encryption scheme with a large symmetrically encrypted component Therefore a small increment in the size of the ciphertext is of no concern to the authors

In Boneh *et al*'s PEKS scheme the four algorithms defined above are implemented as follows

- **KeyGen** This is a standard EC El Gamal public key generation algorithm over a group suitable for pairing based cryptography A suitable group $\mathcal{G}$ of large prime order r is chosen and P a generator of the group $\mathcal{G}$ is picked A suitable bilinear map $e \quad \mathcal{G} \times \mathcal{G} \rightarrow \mu_r$ is selected Two hash functions are chosen, $\mathcal{H} \quad \{0,1\}^* \rightarrow \mathcal{G}$, and $\mathcal{H}_{\mu_r} \quad \mu_r \rightarrow \{0,1\}^n$ The user generates a random $\alpha \in \mathbb{Z}_r^*$ and computes the public key pair $(K_{pri}, K_{pub}) = (\alpha, \alpha P)$ As with standard EC El Gamal it is not necessary to pick a unique generator each time The user publishes the system parameters as

$$\texttt{params} = \{\mathcal{G}, \mu_r, e, P, K_{pub}, \mathcal{H}_W, \mathcal{H}_{\mu_r}\} \tag{6 33}$$

- **PEKS** To compute the PEKS of the keyword W, the user, using the recipients public key K_{pub}, first calculates $t = e(\mathcal{H}_W(W), K_{pub})^x$ for a random $x \in \mathbb{Z}_r^*$ It calculates $H = \mathcal{H}_{\mu_r}(t)$ and the point $S = xP$, and outputs the tuple (S, H)

[15] An anonymous PKI scheme is one in which the identity of the recipient is not obvious from the ciphertext

- **Trapdoor:** To generate T_W, the trapdoor information for the keyword W, a user with private key K_{pri} computes the value $K_{pri}\mathcal{H}_W(W)$[16].

- **Test:** This is used to test whether a keyword is included in a ciphertext. Given a PEKS ciphertext, W a keyword to search for, and T_W Trapdoor information relating to W, Test performs the following check:

$$\mathcal{H}_{\mu_r}(e(T_W, S)) \stackrel{?}{=} H \tag{6.34}$$

  If the test passes then it is accepted that W is in the list of encrypted keywords.

Theorem 6.4.1. *The non-interactive searchable encryption scheme (PEKS) above is semantically secure against a chosen keyword attack in the random oracle model assuming BDH is intractable [30].*

6.5 LMQ PEKS: A PEKS based on Sakai and Kasahara IBE

In [2] Abdalla *et al.* show that any annoymous IBE scheme can be transformed into a PEKS. In a new result we (Libert, McCullagh and Quisquater[17]) show the PEKS scheme resulting from Sakai and Kasahara's IBE. This is the most efficient PEKS scheme known, as in common with most Sakai and Kasahara identity-based cryptosystems it does not require a pairing computation in the ciphertext generation stage. It should be noted that in Boneh *et al*'s scheme a pairing computation is required for every keyword that is included in the ciphertext. The scheme described in this section is otherwise very similar to Boneh *et al*'s scheme.

The scheme consists of the same four algorithms that comprise any PEKS: **KeyGen**, **PEKS**, **Trapdoor** and **Test**. In our scheme they are instanciated as follows:

- **KeyGen:** This is a standard EC El Gamal public key generation algorithm over a

[16] This is similar to a BF identity based private key for the identity W.

[17] This was joint work with Libert and Quisquater which was never published.

group suitable for pairing based cryptography Two suitable groups $\mathcal{G}$ and μ_r of large prime order r are chosen and P a generator of the group $\mathcal{G}$, and $g = e(P, P) \in \mu_r$ a generator of μ_r, are picked A suitable bilinear map e $\mathcal{G} \times \mathcal{G} \to \mu_r$ is selected Two hash functions are chosen $\mathcal{H}_W$ $\{0,1\}^* \to \mathbb{Z}_r^*$, and $\mathcal{H}_{\mu_r}$ $\mu_r \to \{0,1\}^n$ The user generates a random $\alpha \in \mathbb{Z}_r^*$ and computes the public key pair $(K_{pri}, K_{pub}) = (\alpha, \alpha P)$ The user publishes their public key and system parameters as

$$\texttt{params} = \{\mathcal{G}, \mu_r, e, P, g, K_{pub}, \mathcal{H}_W, \mathcal{H}_{\mu_r}\} \tag{6 35}$$

- **PEKS** To compute the PEKS of the keyword W, a user, using the recipients' public key and parameters, first calculates $t = g^x$ for a random $x \in \mathbb{Z}_r^*$ They calculate $H = \mathcal{H}_{\mu_r}(t)$ and the point $S = x(\alpha + \mathcal{H}_W(W))P$, and output the tuple (S, H)

- **Trapdoor** To generate T_W, the trapdoor information for the keyword W, a user with private key $K_{pri} = \alpha$ computes the value $(\alpha + \mathcal{H}_W(W))^{-1}P$ This is distributed to the third party, for example a mail gateway

- **Test** This is used to test whether a keyword is included in a ciphertext Given a PEKS ciphertext (S, H), W (a keyword to search for) and T_W (trapdoor information relating to W), the third party checks the following

$$\mathcal{H}_{\mu_r}(e(S, T_W)) \stackrel{?}{=} H \tag{6 36}$$

If the test passes then it is accepted that W is in the list of encrypted keywords

Theorem 6 5 1 *Using the same security model as defined by Boneh* et al , *the PEKS defined in this section is semantically secure against chosen-keyword attacks if the p-BDHI problem is intractable The security of the scheme is shown using points from linearly independent groups*

6 6 Security Proof of the LMQ PEKS

Theorem 6 6 1 *The PEKS is semantically secure against chosen-keyword attacks if the p-BDHI problem is intractable*

Proof Algorithm $\mathcal{B}$ takes as input $\langle P, Q, \alpha Q, \alpha^2 Q, \quad, \alpha^p Q \rangle$, where P and Q are from linearly independent groups, and attempts to extract $e(P,Q)^{1/\alpha}$ from its interaction with $\mathcal{A}$

In a preparation phase, $\mathcal{B}$ selects at random an index $\ell \overset{R}{\leftarrow} \{1, \quad, q_{\mathcal{H}_W}\}$, elements $I_\ell \overset{R}{\leftarrow} \mathbb{Z}_q^*$ and $w_1, \quad, w_{\ell-1}, w_{\ell+1} \quad, w_{q_{\mathcal{H}_W}} \overset{R}{\leftarrow} \mathbb{Z}_q^*$ For $i = 1, \quad, \ell-1, \ell+1, \quad, q_{\mathcal{H}_W}$, it computes $I_i = I_\ell - w_i$ As in the technique of Boneh-Boyen, it sets up generators $G_2 \in \mathcal{G}_2$, $G_1 = \psi(G_2) \in \mathcal{G}_1$, where ψ is a distortion map from $\mathcal{G}_2$ to $\mathcal{G}_1$, and another $\mathcal{G}_2$ element $U = \alpha G_2$ such that it knows $q_{\mathcal{H}_W} - 1$ pairs $(w_i, H_i = (1/(w_i + \alpha))G_2)$ for $i \in \{1, \quad, q_{\mathcal{H}_W}\} \backslash \{\ell\}$ The public key Q_{pub} is chosen as

$$Q_{pub} = -U - I_\ell G_2 = (-\alpha - I_\ell)G_2$$

so that its (unknown) private key is implicitly set to $x = -\alpha - I_\ell \in \mathbb{Z}_q^*$ For all $i \in \{1, \quad, q_{\mathcal{H}_W}\} \backslash \{\ell\}$, we have $(I_i, -H_i) = (I_i, (1/(I_i + x))G_2)$

$\mathcal{B}$ then initializes a counter ν to 1 and starts the adversary $\mathcal{A}$ on input of (G_1, G_2, Q_{pub}) Throughout the game, we assume that $\mathcal{H}_W$-queries are distinct, that the target keywords W_0^*, W_1^* are submitted to $\mathcal{H}_W$ at some point and that any query involving a keyword comes after a $\mathcal{H}_W$-query on it

- $\mathcal{H}_W$-queries (let us call W_ν the input of the ν^{th} one such query) $\mathcal{B}$ answers I_ν and increments ν

- $\mathcal{H}_{\mu_r}$-queries on input $\gamma_j \in G_T$ $\mathcal{B}$ returns a random $B_j \overset{R}{\leftarrow} \{0,1\}^n$ and stores the pair (γ, B_j) in list L_2

- Trapdoor queries on an input of a keyword W_ν if $\nu = \ell$, then the simulator fails

Otherwise, it knows that $\mathcal{H}_W(W_\nu) = I_\nu$ and returns $-H_\nu = (1/(I_\nu + x))\, G_2 \in \mathcal{G}_2$

At the challenge phase, $\mathcal{A}$ outputs two distinct keywords (W_0^*, W_1^*) for which she never obtained the trapdoors If $W_0^*, W_1^* \neq W_\ell$, $\mathcal{B}$ aborts Otherwise, we may assume that $W_0^* = W_\ell$ (the case $W_1^* = W_\ell$ is treated in the same way) $\mathcal{B}$ picks $\xi \xleftarrow{R} \mathbb{Z}_q^*$ and $B^* \xleftarrow{R} \{0,1\}^n$ to return the challenge $S^* = [A^*, B^*]$ where $A^* = -\xi G_1 \in \mathcal{G}_1$ If we define $\rho = \xi/\alpha$ and since $x = -\alpha - I_\ell$, we can check that

$$A^* = -\xi G_1 = -\alpha\rho G_1 = (I_\ell + x)\rho G_1 = \rho I_\ell G_1 + \rho\psi(Q_{pub}) \tag{6 37}$$

$\mathcal{A}$ cannot recognize that S^* is not a proper ciphertext unless she queries $\mathcal{H}_{\mu_r}$ on $e(A^*, G_2^{(1/(x+\mathcal{H}_W(W_0^*))}) = e(G_1, G_2)^\rho$ or $e(A^*, G_2^{(1/(x+\mathcal{H}_W(W_1^*))})$ Along the second stage, her view is simulated as before and her eventual output is ignored Standard arguments can show that a successful $\mathcal{A}$ is very likely to query $\mathcal{H}_{\mu_r}$ on either $e(A^*, G_2^{(1/(x+\mathcal{H}_W(W_0^*))}) = e(G_1, G_2)^\rho$ or $e(A^*, G_2^{(1/(x+\mathcal{H}_W(W_1^*))})$ if the simulation is indistinguishable from a real attack environment Let AskH_2 denote this event

In a real attack, we have

$$\Pr[\mathcal{A} \text{ wins}] \leq \Pr[\mathcal{A} \text{ wins}|\neg\mathsf{AskH}_2]\Pr[\neg\mathsf{AskH}_2] + \Pr[\mathsf{AskH}_2] \tag{6 38}$$

Clearly, $\Pr[\mathcal{A} \text{ wins}|\neg\mathsf{AskH}_2] = 1/2$ and $\Pr[\mathcal{A} \text{ wins}] \leq 1/2 + (1/2)\Pr[\mathsf{AskH}_2]$ On the other hand, we have

$$\Pr[\mathcal{A} \text{ wins}] \geq \Pr[\mathcal{A} \text{ wins}|\neg\mathsf{AskH}_2](1 - \Pr[\mathsf{AskH}_2]) = \frac{1}{2} - \frac{1}{2}\Pr[\mathsf{AskH}_2]$$

It comes that $\epsilon \leq |\Pr[\mathcal{A} \text{ wins}] - 1/2| \leq \frac{1}{2}\Pr[\mathsf{AskH}_2]$ and thus $\Pr[\mathsf{AskH}_2] \geq 2\epsilon$ This shows that, provided the simulation is consistent, $\mathcal{A}$ issues a $\mathcal{H}_{\mu_r}$-query on either $e(A^*, G_2^{(1/(x+\mathcal{H}_W(W_0^*))})$ or $e(A^*, G_2^{(1/(x+\mathcal{H}_W(W_1^*))})$ at some point of the game with probability at least ϵ With probability ϵ, a $\mathcal{H}_{\mu_r}$-query involving $e(A^*, G_2^{(1/(x+\mathcal{H}_W(W_0^*))}) = e(G_1, G_2)^\rho$ will be issued To produce a result, $\mathcal{B}$ fetches a random record from the lists L_2 With probability $1/q_{\mathcal{H}_{\mu_r}}$, the chosen record contains the right element $r = e(G_1, G_2)^\rho = e(P, Q)^{f(\alpha)^2\xi/\alpha}$, where $f(z) = \sum_{i=0}^{p-1} c_i z^i$ is the polynomial for which $G_2 = f(\alpha)Q$. The p-BDHIP solution can be extracted by noting that, if $\gamma^* = e(P, Q)^{1/\alpha}$, then

$$e(G_1, G_2)^{1/\alpha} = \gamma^{*(c_0^2)} e\Big(\sum_{i=0}^{p-2} c_{i+1}(\alpha^i P), c_0 Q\Big) e\Big(G_1, \sum_{j=0}^{p-2} c_{j+1}(\alpha^j) Q\Big).$$

In an analysis of $\mathcal{B}$'s advantage, we note that it only fails in providing a consistent simulation because of one of the following independent events:

E_1: $W_0^*, W_1^* \neq W_\ell$.

E_2: $\mathcal{B}$ aborts when answering a trapdoor query.

We clearly have $\Pr[\neg E_1] = (q_{\mathcal{H}_W} - 1)/\binom{q_{\mathcal{H}_W}}{2} = 2/q_{\mathcal{H}_W}$ and we know that $\neg E_1$ implies $\neg E_2$. We thus find $\Pr[\neg E_1 \wedge \neg E_2] = 2/q_{\mathcal{H}_W}$. It follows that $\mathcal{B}$ outputs the correct result with probability $2\epsilon/(q_{\mathcal{H}_W} q_{\mathcal{H}_{\mu_r}})$. □

6.7 Optimisations

In this section we look at the optimisations of Baek *et al.* [9] and consider their applicability to our scheme.

6.7.1 Refreshing Keywords

Obviously one of the problems with a PEKS system is that, for the system to become operational, the keywords must come from a relatively small set which we assume is publicly known. For example one might want to forward emails from known email addresses `ceo@company.com`, or emails that mention a certain term such as *new contract*. Baek *et al.* [9] propose refreshing the keywords by appending date information in much the same way that identities are given validity windows in IBE [31]. Using this method the current date is appended to the keyword. This method seems reasonable as information on how to construct keywords can be distributed with public keys.

Another obvious method is the use of ephemeral public keys, which are signed using a long term private key Since this is an encryption scheme we assume that lookups of public keys are not an inconvenience to the sender Also, the sender will only have to check the last link in the certificate chain We note that this method is more efficient than the method of appending dates to the the keywords, as, if the keywords do not change, then we can store keyword hashes and do not have to repeatedly perform 'hash and map"

Baek *et al* [9] seem to imply that the trapdoor information should only be released to the gateway at the start of its validity period, in much the same way as a private key is only distributed to a user of an identity based system at the start of the validity period for the corresponding public key We note here that this does not have to be the case for the distribution of trapdoor information The recipient could easily publish all of its intended public keys ahead of time, and at the same time give the gateway all of the corresponding trapdoor information The gateway could then store all of the trapdoor information and discard them when they expire

6 7 2 Removal of the Secure Channel

Another idea Baek *et al* [9] suggested was the removal of the secure channel for the distribution of the trapdoor information from the user to the gateway This incurred a penalty of one extra exponentiation in the group μ_r for the sender We observe that with our scheme we can remove the secure channel without any additional burden on any of the users in the system The modified system is only slightly different from the original scheme that we propose in section 6 5 and is outlined here

A PEKS scheme with removal of the secure channel between the public key owner and the third party requires five algorithms **KeyGen User, KeyGen GW, Encrypt, Trapdoor, Test** KeyGen User is a public key generation algorithm carried out by the recipient KeyGen GW is an algorithm carried out by the mail gateway Encrypt is carried out by the sender Trapdoor is carried out by the recipient to produce the trapdoor information which is given to the gateway Test is carried out by the gateway

- **KeyGen User** This is a standard EC El Gamal public key generation algorithm over a group suitable for pairing based cryptography Two suitable groups $\mathcal{G}$ and μ_r of large prime order r are chosen and P a generator of the group $\mathcal{G}$, and g a generator of μ_r such that $g = e(P, P)$, are picked A suitable bilinear map $e \quad \mathcal{G} \times \mathcal{G} \to \mu_r$ is selected Two hash functions are chosen $\mathcal{H}_W \quad \{0,1\}^* \to \mathbb{Z}_r^*$, and $\mathcal{H}_{\mu_r} \quad \mu_r \to \{0,1\}^k$ The user generates a random $\alpha \in \mathbb{Z}_r^*$ and computes the public key pair $(K_{pri}, K_{pub}) = (\alpha, \alpha P)$ The user publishes their public key and system parameters as

$$\texttt{params} = \{\mathcal{G}, \mu_r, e, P, g, K_{pub}, \mathcal{H}_W, \mathcal{H}_{\mu_r}\} \tag{6 39}$$

- **KeyGen GW** Using **params** the gateway generates a random value $y \in \mathbb{Z}_r^*$ and computes $g_{gw} = g^y \in \mathcal{G}$

- **PEKS** To compute the PEKS of the keyword W, a sender, using the recipients' public key and parameters, and the value g_{gw} obtained from the gateway, first calculates $t = g_{gw}^x$ for a random $x \in \mathbb{Z}_r^*$ They calculate $H = \mathcal{H}_{\mu_r}(t)$ and the point $S = x(\alpha + \mathcal{H}_W(W))P$, and output the tuple (S, H)

- **Trapdoor** To generate T_W, the trapdoor information for the keyword W, a user with private key $K_{pri} = \alpha$ computes the value $(\alpha + \mathcal{H}_W(W))^{-1}P$ This information can be passed to the gateway in the clear

- **Test** This is used to test whether a keyword is included in a ciphertext Given a PEKS ciphertext (S, H), W (a keyword to search for) and T_W (trapdoor information relating to W), and the gateway's secret value y the mail gateway checks the following

$$\mathcal{H}_{\mu_r}(e(S, yT_W)) \stackrel{?}{=} H \tag{6 40}$$

The trapdoor information can now be distributed to the gateway in the clear, since the Test algorithm now requires knowledge of y, which is known only to the gateway

If the test passes then it is accepted that W is in the list of encrypted keywords

The Security of the "No Secure Channel" Scheme

Theorem 6 7 1 *The PEKS with secure channel removed is semantically secure against chosen-keyword attacks if the p-BDHI problem is intractable*

The proof is very similar to the proof given for the scheme described above, and is included in appendix B 1

6 7 3 Randomness Re-use

Baek *et al* suggest randomness re-use for use with the Boneh *et al* scheme However, we note here that randomness re-use is not possible with our scheme Randomness re-use, where the same $t = g^x$ is used for two different encryptions introduces the following vulnerability

Say an attacker *guesses* two popular keywords, he can check for their presence by doing the following test

Let W_0, W_1 represent the guessed keywords respectively Then, if the attacker's guesses are correct and randomness re-use is used, the resulting ciphertext will include $S = x(\alpha + H_W(W_0))P$ and $S' = x(\alpha + H_W(W_1))P$ $H = \mathcal{H}_{\mu_r}(g^x)$ will be the same for both encrypted keywords, due to randomness re-use

$$xP = (H_W(W_0) - H_W(W_1))^{-1}(S - S') \tag{6 41}$$

$$g^x = e(P, xP) \tag{6 42}$$

The attacker then carries out the following test

$$\mathrm{H} \stackrel{?}{=} \mathcal{H}_{\mu_r}(g^x) \tag{6 43}$$

If the test is passed, the attacker knows that the two keywords were present This is a very real attack on a PEKS system, since the keywords are likely to come from a small, well defined dictionary

6.8 Efficiency of the Sakai and Kasahara PEKS Scheme

We now look at the efficiency of our scheme with comparison to the Boneh *et al* scheme [30] We will then look at the various modifications that can be made to that scheme and see how they may be applied to the scheme which we present - some of these where suggested by Baek *et al* in [9] The figures in brackets represent the timings when using Scott's faster hash and map algorithm

# keywords	Boneh *et al* (naive)	Boneh *et al* (Randomness Re-use)	**ours**
1	599ms (302ms)	599ms (302ms)	**188ms**
5	2995ms (1510ms)	2619ms (1134ms)	**940ms**
10	5990ms (3020ms)	5144ms (2174ms)	**1880ms**

Table 6 3 Comparison of our scheme with that of Boneh *et al*

As we can see the new scheme is faster, but due to the fact that we cannot make use of randomness re-use, it does not manage to significantly outperform the Boneh et al scheme at higher numbers of keywords as might be expected

6 9 Conclusion

We have seen in this chapter that IBE is not the only talent of pairings We have given a review of a few of the more interesting IBE schemes, such as the seminal Boneh and Franklin IBE and the more efficient Sakai and Kasahara IBE scheme We have also looked at Public key Encryption with Keyword Search, and presented the fastest known scheme This just gives a flavour of the types of encryption schemes that are possible with pairings

Chapter 7

Two-Party Identity-Based Key Agreements Protocols

Key agreement protocols are fundamental to the study of asymmetric cryptography. Key agreement protocols that are based on the discrete logarithm problem are closely related to public key encryption. The idea of a key agreement scheme is to allow two entities to share a common ephemeral (session) key. The process of establishing a session key is called *key establishment*. There are two ways to achieve a shared key, one being a *key transportation* protocol, where one entity is trusted with generating a key and transporting it securely to the other user. For example by encrypting it using the public key of the recipient, or by encrypting it using a symmetric encryption algorithm under a master key that is shared by both users. This is sometimes referred to as a digital envelope [79]. Some key transportation algorithms make use of a third party, for example the key agreement protocol in the Kerberos network authentication system [132]. Another way of generating a shared session key is that the two parties generate tokens that they swap. This is called a *key agreement* protocol. These tokens allow the users to create a common shared secret. For a good general reference see [91, Ch. 12].

The most famous key agreement protocol is the *Diffie-Hellman* protocol. It was presented in 1976, in the ground-breaking paper 'New Directions in Cryptography" [58]. It allows two users who have not previously shared information to establish a shared secret session key in the presence of passive eavesdroppers The Diffie-Hellman key agreement, as shown in Table 7 1 was extremely important, because it proposed something that was so counter-intuitive Up until this point it was assumed that if you wanted to engage in a cryptographic protocol with another party you must have previously established some common shared secret with them (symmetric cryptography) It layed the foundation stone for public key cryptography

The Diffie-Hellman key agreement has two system parameters p and g Parameter p is a prime number and parameter g is a generator of a large prime order subgroup of order r α and β are two number drawn at random from the set of integers less than r

Alice		Bob
$T_A = g^{\alpha}$	$\rightarrow$	
	$\leftarrow$	$T_B = g^{\beta}$
$K_A = T_B^{\alpha}$		$K_B = T_A^{\beta}$
$K_A = g^{\alpha\beta}$		$K_B = g^{\alpha\beta}$

Table 7 1 The Diffie Hellman Key Agreement

The Diffie-Hellman key agreement is not perfect however It suffers from what is called the "man-in-the-middle" attack This derives from the fact that the parties are not authenticated in any way during the protocol This is quite a critical flaw The idea behind the attack is that you can create a shared secret with someone, but you do not know for sure with whom you are communicating The protocol itself is secure, but you do not know if you are talking with your intended recipient, or if you are talking directly to an eavesdropper If the eavesdropper manages to dupe Alice and Bob into talking directly with him then he can relay (and read) all of the messages between them The eavesdropper becomes the 'man in the middle"[1] The eavesdropper does this by negotiating two seperate session keys, one with Alice and the other with Bob This is shown clearly in Table 7 2 The eavesdropper can now decrypt messages from Alice and re-encrypt them and forward them on to Bob

Alice		Eve		Bob
$K_A = g^{\alpha}$	$\rightarrow$			
	$\leftarrow$	$K_{E_B} = g^{\beta_E}$		
$\mathsf{key}_{AE} = K_{E_B}^{\alpha}$		$\mathsf{key}_{AE} = K_A^{\beta_E}$		
$\mathsf{key}_{AE} = g^{\alpha\beta_E}$		$\mathsf{key}_{AE} = g^{\alpha\beta_E}$		
		$K_{E_A} = g^{\alpha_E}$	$\rightarrow$	
			$\leftarrow$	$K_B = g^{\beta}$
		$\mathsf{key}_{BE} = K_B^{\alpha_E}$		$\mathsf{key}_{BE} = K_{E_A}^{\beta}$
		$\mathsf{key}_{BE} = g^{\alpha_E\beta}$		$\mathsf{key}_{BE} = g^{\alpha_E\beta}$

Table 7.2 A Man in the Middle Attack on the Diffie-Hellman key Agreement

We have seen that the Diffie-Hellman protocol is extremely elegant. However, we have also seen that it does not have any real practical application as it stands. If we assume that we only use cryptography to keep secrets then we would also assume that we want to know with confidence who we are telling those secrets to. This leads to the obvious question: What properties should we expect of a key agreement protocol?

7.1 Definition of an Identity Based Key Agreement Protocol

A two party identity based key agreement protocol contains the algorithms **Setup**, **Extract** and the protocol **Key Agreement**. Setup and Extract are carried out by the KGC and are common to all identity based cryptosystems. Key Agreement, which is common to all key agreement protocols, is carried out by the two end users.

- **Setup** takes as input a security parameter k. It outputs system wide **params**, which are made public. It also produces a master secret key s, which is known only to the KGC.

- **Extract** takes as input **params**, s, and the identity of a user ID. It outputs a private key d for this user.

- **Key Agreement** is carried out between two end users. The result of this algorithm is that both parties obtain a shared secret value.

7 2 Properties of Key Agreement Protocols

Properties of Key Agreements [44, 8]

- **Known Key Security** Each run of the protocol should result in a fresh, unique, randomly distributed session key Recovery of arbitrarily many previous session keys should not help an attacker in determining the currently agreed session key
- **Forward Secure** A key agreement is said to be forward secure if knowledge of all long term private keys does not compromise previously established session keys A scheme is said to have **partial forward secrecy** if knowledge of all of the private keys of the communicating entities is required before previous session keys can be recovered
- **Key Compromise Impersonation Resilience** Compromise of Alice s long term private key will (obviously) allow an attacker to impersonate Alice to other entities However it is desirable that this does not allow the attacker to impersonate other entities to Alice
- **Unknown Key Share Resilience** This is an attack whereby an entity A finishes an execution of a key agreement protocol believing that a common key is shared with an entity B (this is in fact the case) but B falsely believes that the key is shared with another entity E ($\neq A$)
- **Key Control** Neither party should be able to force the agreed session key to be a certain value, or to be in a certain small subset of the key space

A key agreement protocol is said to provide key authentication if entity A is assured that no other entity apart from a specifically identified entity B can possibly learn the value of the shared secret key It is an 'Authenticated Key Agreement" (AK) protocol This does not guarantee that entity B knows a particular shared secret, it only guarantees that no-one else knows it

This gives rise to a further definition, an authenticated key agreement protocol in which entity A is assured that entity B has a particular secret value is called an 'Authenticated Key Agreement with Key Confirmation" (AKC) It is easy to convert an Authenticated Key Agreement into an Authenticated Key Agreement with Key Confirmation The basis of this transformation is to add another pass to the protocol in which the agreed session key is used to MAC[1] some data that contains redundancy

Often, if we wish to use the secret value as a key to encrypt a message that contains redundancy, for example a message written in the English language or a real-time voice call, we do not need to add key confirmation The key will be confirmed by the fact that a message with the expected redundancy was recovered If the decryption reveals a random binary string or the phone call just contains 'white noise" then we can assume that the secret value was not transmitted correctly

Other desirable attributes of AK and AKC protocols include

- **Small Number of Passes** A pass in a protocol is a token (message) sent from entity A to entity B or visa versa

- **Small Number of Rounds** A new round is classified by its dependence on information exchanged in a previous round For example in a tripartite key agreement an entity A might send different messages to entities B and C However, if these can both be sent at the same time, we say that this is one round of the protocol We would classify Joux's key agreement protocol [77] as a one round protocol, since the information that any entity sends is independent of the information sent to them from other entities Many two party key agreements are one round protocols

- **Small Computational Complexity** The computational complexity is the amount of work done by the communicating entities in order to successfully share a secret value

[1] MAC Message Authentication Code, similar to a digital signature, but does not offer non-repudiation

- **Role Symmetry:** Do all of the parties in the protocol carry out identitical computations? If they do then the key agreement is *role symmetric*. This may be advantageous if both entities have the same computational resources, or not, if the entities have very different computational resources (for example a smart card / terminal key agreement).

There are many key agreement protocols based on bilinear maps, and many have subsequently been broken. One of the first applications of pairing based cryptography was a tripartite key agreement protocol by Joux [77]. This protocol does not authenticate the users, and thus is susceptible to the man-in-the-middle attack. However, it was a significant step in the development of pairing based cryptography. This original scheme was not identity-based.

Many key agreement protocols from bilinear maps have been since proposed. Smart [127], and Chen and Kudla [44] have proposed two-party key agreement protocols, neither of which have been broken. Nalla proposes a tripartite identity-based key agreement in [97], and Nalla and Reddy propose a scheme in [99], but both have been broken [47, 121]. Shim presents two key agreements [123, 122], but both these schemes have been broken by Sun and Hsieh [131]. Another set of authenticated tripartite key agreements proposed by Al-Riyami and Paterson [5] were attacked by Shim [120], with one being broken. The non-interactive identity based scheme of Sakai, Ohgishi and Kasahara [111], and the scheme of Scott [114] both suffer from key compromise impersonation.

Most identity-based key agreement protocols have the property of *key escrow*: the trusted authority that issues private keys can recover the agreed session key. This feature is either acceptable, unacceptable, or desirable depending on the circumstances. For example, escrow is essential in situations where confidentiality as well as an audit trail is a legal requirement, as in confidential communication in the health care profession. There are other examples, such as personal communications, where it would be advantageous to turn escrow off.

The two-party key agreements proposed by Smart and by Chen and Kudla are escrowed schemes by default. A modification suggested by Chen and Kudla [44] to remove escrow can also be applied to Smart's scheme. However, this modification creates additional computational overhead. Scott's scheme does not allow escrow, and there seems no obvious way to introduce this feature, bar one party in the protocol sending a third party a copy of the agreed key.

If all parties in an identity based key agreement protocol have had their private keys issued by the same KGC then we say that they are all members of the same domain. If a key agreement protocol requires that both users have keys issued by the same KGC [111, 114] then this, for example, might mean that two workers from the same company would be able to generate a shared secret. However employees from two different companies would not be able to generate such a shared secret. Chen and Kudla proposed a solution to this problem in [44].

7.3 Security Models for Identity Based Key Agreements

We adopt the security model proposed by Bellare and Rogaway [19], modified by Blake-Wilson *et al.* [23], and used in proving the security of the key agreement protocols introduced in [44] and [89].

The model includes a set of parties, each modelled by an oracle. We use the notation $\prod_{i,j}^{n}$, meaning a participant/oracle i believing that it is participating in the n-th run of the protocol with j. Oracles keep transcripts of all communications in which they have been involved. Each oracle has a secret private key, issued by a KGC, which has run a BDH parameter generator $\mathcal{B}$ and published groups $\mathcal{G}$ and μ_r, a bilinear map of the form $e : \mathcal{G} \times \mathcal{G} \rightarrow \mu_r$, a group generator P of $\mathcal{G}$, and a master public key sP.

The model contains an adversary E which has access to all message flows in the system. E is not a (legitimate) user or KGC[2]. All oracles only communicate with each other via E. E can replay, modify, delay, interleave or delete messages. E is benign if it acts like a wire

[2]Does not hold a private key of the target identity or the master secret key.

and does not modify communication between oracles From [19], if two oracles receive, via the adversary, property formatted messages that have been generated exclusively by the other oracle, and both oracles accept[3], we say that these two oracles have had a matching conversation

The adversary E at any time can make the following queries

- **Create** E sets up a new oracle in the system that has public key ID, of E's choosing E has access to the identity / public key of the oracle The private key is obtained from the KGC

- **Send** E sends a message of his choice to an oracle i, $\prod_{i,j}^{n}$, in which case i assumed that the message came from j E can also instruct the actual oracle j to start a new run of the protocol with i by sending a λ_j signal to j Using the terminology of [23] an oracle is an *initiator oracle* if the first message that it receives is λ, otherwise it is a *responder oracle*

- **Reveal** E receives the session key that is currently being held by a particular oracle

- **Corrupt** E receives the long term private key being held by a particular oracle

- **Test** E receives either the session key or a random value from a particular oracle Specifically, to answer the query the oracle flips a fair coin $c \in \{0,1\}$, if the answer is 0 it outputs the agreed session key, and if the answer is 1 it outputs a random element of $\{0,1\}^k$ E then must decide whether c is 0 or 1, call this prediction c' E's advantage in distinguishing the actual session key held by an uncorrupted party from a key sampled at random from $\{0,1\}^k$ in this game, with respect to the security parameter k, is given by

$$Advantage^E(k) \quad = \quad |Pr[c' = c] - 1/2| \qquad (7\ 1)$$

[3]The oracles enter the accepted state as defined in [19]

The Test query can be performed only once, against an oracle that is in the Accepted state (see below), and which has not previously been asked a Reveal or Corrupt query

An oracle may be in one of the following states (it cannot be in more than one state)

Accepted If the oracle decides to accept a session key, after receipt of properly formatted messages

Rejected If the oracle decides not to accept and aborts the run of the protocol

* If the oracle has yet to decide whether to accept to reject for this run of the protocol We assume that there is some time-out on this state

Opened If a Reveal query has been performed against this oracle for its last run of the protocol (its current session key is revealed)

Corrupted If a Corrupt query has ever been performed against this oracle

Definition [23] A protocol is an AK protocol if

- In the presence of the benign adversary on $\prod_{i,j}^{n}$ and $\prod_{j,i}^{t}$, both oracles always accept holding the same session key, and this key is distributed uniformly at random on $\{0,1\}^k$, if for every adversary E
  - If uncorrupted oracles $\prod_{i,j}^{n}$ and $\prod_{j,i}^{t}$ have matching conversations then both oracles accept and hold the same session key,
  - $Advantage^E(k)$ is negligible

7 4 The Non-interactive Identity Based Key Agreement Protocol of Sakai, Ohgishi and Kasahara

As mentioned in the introduction to this chapter, the Diffie-Hellman key agreement protocol and the paper "New Directions in Cryptography" [58] laid the foundation stone for asymmetric cryptography However, identity based key agreement protocols using pairings are a much more recent discovery The first such identity based key agreement protocol was proposed by Sakai, Ohgishi and Kasahara in 2000 [111] As an added bonus this scheme is also non-interactive, and is one of the simplest key agreement schemes in existence

The protocol proceeds as follows

- **Setup** The KGC chooses an appropriate group $\mathcal{G}$ of order r and selects a generator of that group P Therefore we have $\langle P \rangle = \mathcal{G}$ The KGC generates a random $s \in_R \mathbb{Z}_r^*$ The KGC calculates $P_{pub} = sP$ The KGC publishes descriptions of hash functions $\mathcal{H}_k \ \ \mu_r \rightarrow \{0,1\}^k, \mathcal{H}_{ID} \ \ \{0,1\}^k \rightarrow \mathcal{G}$, and a bilinear map $e \ \ \mathcal{G} \times \mathcal{G} \rightarrow \mu_r, \mathcal{G}, P_{pub}$ and μ_r

- **Extract** The KGC issues private keys to users, first by checking that they have a legitimate claim on ID, the identity for which they wish to receive the private key The KGC generates their private key as sQ_{ID} where $Q_{ID} = \mathcal{H}_{ID}(ID) \in \mathcal{G}$

- **Key Agreement**

  Suppose the user with identity ID_A and public key Q_{ID_A}, wishes to set up a shared secret with the user with identity ID_B, and corresponding public key Q_{ID_B} The shared secret is calculated as $\mathcal{H}_k(e(sQ_{ID_A}, Q_{ID_B}))$

  Suppose the user with identity ID_B and public key Q_{ID_B}, wishes to setup a shared secret with a user with identity ID_A, and corresponding public key Q_{ID_A} The shared secret is calculated $\mathcal{H}_k(e(Q_{ID_A}, sQ_{ID_B}))$

  From bilinearity, it can be observed that

$$e(sQ_{ID_A}, Q_{ID_B}) = e(Q_{ID_A}, sQ_{ID_B}) = e(Q_{ID_A}, Q_{ID_B})^s \qquad (7\ 2)$$

  and therefore both users have agreed the same shared secret, without interaction

7 5 The Identity Based Key Agreement Protocol of Smart

Smart's key agreement [127], like all identity based key agreements, contains the two algorithms **Setup**, **Extract** and and the protocol **Key Agreement** Smart's key agreement makes use of a group $\mathcal{G}$ and a bilinear map of the form $e \quad \mathcal{G} \times \mathcal{G} \rightarrow \mu_r$, where solving the discrete logarithm problem in the groups $\mathcal{G}$ and μ_r is computationally infeasible We denote the order of the groups by r It also makes use of a session key derivation function $\mathcal{H}_k \quad \mu_r \rightarrow \{0,1\}^k$, and a hash function $\mathcal{H}_{ID} \quad \{0,1\}^* \rightarrow \mathcal{G}$ (as described by Boneh and Franklin) to map identities to elements of the group $\mathcal{G}$

The key agreement proceeds as follows

- **Setup** and **Extract** are identitical to the Setup and Extract algorithms specified by Boneh and Franklin
- **Key Agreement** We describe the key agreement between two users, Alice and Bob, who have public keys Q_A and Q_B and private keys sQ_A and sQ_B respectively Alice generates a random $\alpha \in \mathbb{Z}_r^*$ and likewise, Bob generates a random $\beta \in \mathbb{Z}_r^*$ Now the protocol proceeds as shown in Table 7 3

Alice		Bob
αP	$\rightarrow$	
	$\leftarrow$	βP
$K_A = \mathcal{H}_k(e(sQ_A, \beta P) \quad e(Q_B, \alpha sP))$		$K_B = \mathcal{H}_k(e(Q_A, \beta sP) \quad e(sQ_B, \alpha P))$

Table 7 3 Smart's Identity Based Key Agreement

Smart also proposes a Authenticated Key Agreement Scheme with Key Confirmation (AKC), by applying a simple transformation using the key that was exchanged in the key agreement above with a MAC on some redundant data This idea was explored in detail in [23] The key derivation function is now $\mathcal{H}_k \quad \mu_r \rightarrow \{0,1\}^k \times \{0,1\}^k$ This produces two k bit keys, one being used to key the MAC, and therefore for providing confirmation, and the other being the actual session key

Smart's Authenticated Key Agreement with Key Confirmation proceeds as in Table 7 4

Alice		Bob
αP	$\rightarrow$	
		$R = e(Q_A, \beta s P)\ e(sQ_B, \alpha P)$ $(k, k') = \mathcal{H}_k(R)$
	$\leftarrow$	$\begin{cases} \beta P \\ M_1 = MAC'_k(2, Q_B, Q_A, R) \end{cases}$
$R = e(sQ_A, \beta P)\ e(Q_B, \alpha s P)$ $(k, k') = \mathcal{H}_k(R)$ $MAC_{k'}(2, Q_B, Q_A, R) \stackrel{?}{=} M_1$		
$M_2 = MAC_{k'}(3, Q_A, Q_B, R)$ $K_A = k$	$\rightarrow$	$MAC_{k'}(3, Q_A, Q_B, R) \stackrel{?}{=} M_2$ $K_B = k$

Table 7 4 Smart's Identity Based Key Agreement Protocol with Key Confirmation

Provided that both of the verification equations are passed then the agreed session key is k

In his original paper, Smart gives informal security arguments for the security of his scheme, but in a new result we prove it secure in the random oracle model, using a modified version of the security model of Bellare and Rogaway [19] in which reveal queries are not allowed

7 5 1 The Security of Smart's Key Agreement Protocol

The proof of security of the above algorithm relies on the conjectured intractability of the Bilinear Diffie-Hellman Problem The Bilinear Diffie-Hellman Problem is Given $P, aP, bP, cP \in \mathcal{G}$ compute $g^{abc} \in \mu_r$ where $g = e(P, P)$

Assuming that the BDHP is hard (with respect to the security parameter k), we now demonstrate the security of Smart's key agreement protocol

Theorem 7 5 1 *Smart's key agreement protocol is a secure AK protocol, assuming that E does not make any reveal queries and that the hash functions used are modelled as random oracles, and that the BDHP is hard*

See appendix D for the proof. This *original work* has been put in the Appendix, as it simply an adaptation of the security proof which Chen and Kulda gave for their identity based key agreement [44].

7.5.2 Efficiency of Smart's Identity Based Key Agreement Protocol

We now look at the efficiency of Smart's key agreement protocol. Firstly, the AK protocol presented by Smart is role symmetric. This means that both parties to the agreement incur the same computational and bandwidth costs. We see that, without precomputation, the computational cost for each participant is one point scalar multiplication, two pairings and an exponentiation in μ_r. With precomputation, we see that if entity A was to repeatedly communicate with entity B, then the pairing $\gamma_B = e(Q_B, sP)$ could be precomputed and stored. This would mean that A could then complete the key agreement as

$$K_A = \mathcal{H}_k(e(sQ_A, \beta P) \cdot \gamma_B^{\alpha}) \qquad (7.3)$$

This reduces the computational load placed on A to one point scalar multiplication, one pairing and one pairing exponentiation. Since pairing exponentiation is much faster than pairing computation over $k = 2$ curves, this change will achieve a significant increase in the performance of the key agreement.

7.6 The Identity Based Key Agreement of McCullagh and Barreto

We now describe the identity based AK protocol that has been presented by McCullagh and Barreto in [89]. This key agreement protocol, unlike the previous AK protocols of Smart and Chen and Kudla, does not make use of the identity based public key pair of Boneh and Franklin. Instead we use the identity based key pair developed by Sakai and Kasahara [109]. Like the previous schemes, this scheme consists of two algorithms, **Setup** and **Extract**, and the **Key Agreement** itself. Obviously the modifications that were proposed by Chen and

Kudla in [44] also apply We will look at this in more detail later

This algorithm makes use of two groups $\mathcal{G}$ and μ_r of prime order r P is a generator of the group $\mathcal{G}$ It also makes use of two random oracles, $\mathcal{H}_{ID}$ $ID \rightarrow \mathbb{Z}_r^*$ and $\mathcal{H}_K$ $\mu_r \rightarrow \{0,1\}^k$ A bilinear map of the form e $\mathcal{G} \times \mathcal{G} \rightarrow \mu_r$ is selected This scheme also uses $g = e(P, P) \in \mu_r$ g is a generator of μ_r

- **Setup** The KGC generates a random element $s \in \mathbb{Z}_r^*$ The KGC publishes $\mathcal{G}$, μ_r, e $\mathcal{G} \times \mathcal{G} \rightarrow \mu_r$, $\mathcal{H}_{ID}$, $\mathcal{H}_K$, P and sP

- **Extract** The KGC validates that the user requesting the private key is associated with a certain ID The public key for this user is $sP + iP = (s + i)P$, where $i = \mathcal{H}_{ID}(ID) \in \mathbb{Z}_r^*$ The corresponding private key, which requires knowledge of s to compute, is calculated by the KGC as $(s + i)^{-1}P$

- **Key Agreement** First, users Alice and Bob, who have public key pairs $\{(s+a)P, (s+a)^{-1}P\}$ and $\{(s+b)P, (s+b)^{-1}P\}$, generate random α and $\beta \in \mathbb{Z}_r^*$ respectively They then complete the key agreement as shown in Table 7 5

Alice		Bob
$\alpha(s+b)P$	$\rightarrow$	
	$\leftarrow$	$\beta(s+a)P$
key $= \mathcal{H}_K(g^{\alpha} \cdot e(\beta(s+a)P, (s+a)^{-1}P))$		key $= \mathcal{H}_K(g^{\beta} \cdot e(\alpha(s+b)P, (s+b)^{-1}P))$

Table 7 5 McCullagh and Barreto's Authenticated Key Agreement

For clarification, the agreed session key is $\mathcal{H}_K(g^{\alpha+\beta})$ The computational cost associated with this key agreement is one pairing, one pairing exponentiation and one point scalar multiplication We also note that apart from storing long term public keys (which would increase performance), there are no storage overheads with this key agreement protocol

7 6 1 The Security of the Identity Based Key Agreement Protocol of McCullagh and Barreto

The original security proof supplied with the McCullagh and Barreto key agreement was flawed, in that an adversary could tell the difference between the simulated environment and the real world This was a flaw in the security proof only In [48] Cheng and Chen provided a new security proof which relied on a new hard problem, which they introduced, called the k-EBCAA$_1$ assumption

Definition k-EBCAA$_1$ Assumption For an integer k, and $x, y \in_R \mathbb{Z}_r^*, P \in \mathcal{G}, e \quad \mathcal{G} \times \mathcal{G} \to \mu_r$, given $hP, xP, h_0, (h_1, (h_1 + x)^{-1}P), \quad , (h_k, (h_k + x)^{-1}P), yP)$ where $h_i \in_R \mathbb{Z}_r^*$ are different from each other for $0 \leq i \leq k$, to compute $e(P, P)^{y(h_0+x)^{-1}}$ is hard

They then proceeded to provide a proof for the McCullagh and Barreto key agreement assuming this assumption is sound The proof, included in Appendix E, is taken from [48]

7 6 2 Applying Chen and Kudla's modifications to McCullagh and Barreto's Key Agreement Protocol

In [44] Chen and Kudla proposed modifications to their key agreement protocol and Smart's key agreement protocol to add the following properties removal of KGC escrow, key agreement between domains and addition of a key confirmation stage

Since most of these are generic techniques we now look at how they can be applied to the authenticated key agreement protocol of McCullagh and Barreto Firstly we will look at the removal of escrow We actually see that in the McCullagh and Barreto scheme, we can use a technique similar to that of Chen and Kulda Again we modify the key derivation function $\mathcal{H}_K$ This time, however, we do not use exactly the same key derivation function that they use Instead, we use a function of the form $\mathcal{H}_K \quad \mu_r \times \mu_r \to \{0,1\}^k$ We also modify the **Setup** and **Extract** algorithms These modifications will be explained in more detail later

- **Setup** The KGC picks two groups $\mathcal{G}$ and μ_r, both of large prime order r, such that the discrete logarithm problem in these groups is computationally infeasible The KGC makes public hash functions $\mathcal{H}_{ID}$ $\{0,1\}^* \rightarrow \mathbb{Z}_r^*$, $\mathcal{H}_Q$ $\{0,1\}^* \rightarrow \mathcal{G}$ and $\mathcal{H}_K$ $\mu_r \times \mu_r \rightarrow \{0,1\}^k$ The KGC also publishes details of a bilinear map of the form e $\mathcal{G} \times \mathcal{G} \rightarrow \mu_r$ Here we let $g = e(P,Q)$ where P and Q are taken from the same group, and Q is some unknown multiple of P The KGC picks two random public strings (for example the first ten digits of π and the first ten digits of G, the gravitational constant) Interestingly the KGC does not publish generator points in this system, but shows how these points can be generated Two generator points P and Q are required, derived from the constant strings as follows $P = H_Q(\pi)$ and $Q = H_Q(G)$ *This is to inspire confidence that the KGC does not know the value x, such that $P = xQ$, as this could lead to an attack by the KGC* The KGC generates a random $s \in \mathbb{Z}_r^*$ and publishes the point sP

- **Extract** The KGC generates users' public keys using the same Extract algorithm as before, except that the private keys are now generated using the point Q To generate a private key for user ID, the KGC first generates $i = \mathcal{H}_{ID}(ID) \in \mathbb{Z}_r^*$ The public key for this user is $sP + iP = (s+i)P$, the private key is now $(s+i)^{-1}Q$

- **Key Agreement** Two users Alice and Bob, who have public key pairs $\{(s+a)P, (s+a)^{-1}Q\}$ and $\{(s+b)P, (s+b)^{-1}Q\}$ respectively, now generate random secret α and $\beta \in \mathbb{Z}_r^*$ respectively and perform the key agreement as shown in Table 7 6 2

Alice		Bob
$\alpha(s+b)P$	$\rightarrow$	
	$\leftarrow$	$\beta(s+a)P$
$\begin{cases} R_A = e(\beta(s+a)P, (s+a)^{-1}Q) \\ K_A = \mathcal{H}_K(g^{\alpha} \ R_A, R_A^{\alpha}) \end{cases}$		$\begin{cases} R_B = e(\alpha(s+b)P, (s+b)^{-1}Q) \\ K_B = \mathcal{H}_K(g^{\beta} \ R_B, R_B^{\beta}) \end{cases}$

Table 7 6 McCullagh and Barreto's Authenticated Key Agreement protocol with No Escrow

For clarity the shared secret key is now $\mathcal{H}_K(g^{\alpha+\beta}, g^{\alpha\beta})$ Since the secret is processed using a random oracle, this time the adversary E must have advantage in finding both parts of the input to $\mathcal{H}_K$ We proved earlier that this was not possible if we use the same point in the generation of both the public and private keys It is also not possible if we use points for which the discrete logarithm is unknown All that is required is for the challenger C to answer the random oracle queries with two points for which C knows the discrete logarithm between them, whilst not revealing this discrete logarithm to E

We also notice that in this situation the KGC can recover the values g^{α} and g^{β} and thus the first input into the oracle $\mathcal{H}_K$ However, the KGC cannot recover the value $g^{\alpha\beta}$ This would imply a non-negligible advantage in solving the DHP over the group μ_r This was first proposed by McCullagh and Barreto at CT-RSA on 15th Feb 2005 However a similar scheme has since appeared in a separate paper on the IACR Cryptology eprint Archive See [138] for more details

We now look at Chen and Kudla's second modification to Smart's protocol which allowed key agreement between domains We notice that their scheme is not immediately applicable to the McCullagh and Barreto AK protocol, since the shared secret that the McCullagh and Barreto protocol generates does not depend on any way on the master secret of the KGC (it is annulled by the pairing of the received point and the private key) Therefore, all that is needed is that the KGCs agree on the same groups, pairing implementation and point P

We assume that Alice has obtained her private key from KGC_1, which has as its master secret s_1 and which publishes the point s_1P Therefore Alice's public key pair is $\{(s_1 + a)P, (s_1 + a)^{-1}P\}$ Likewise, Bob has obtained his public key from KGC_2, which has the master secret s_2 and which has published s_2P Bob's key pair therefore is $\{(s_2 + b)P, (s_2 + b)^{-1}P\}$ The key agreement protocol proceeds as shown in Table 7 7

A More Flexible Approach to Key Agreement Between Domains

Another new way to implement key agreement between domains is just to use a key derivation function $\mathcal{H}_K$ $\mu_{r_1} \times \mu_{r_2} \to \{0,1\}^k$, where μ_{r_1} is the group used by KGC_1 and μ_{r_2} is the group used by KGC_2 Then we can combine any of the above key agreement protocols

Alice		Bob
$\alpha(s_2+b)P$	$\rightarrow$	
	$\leftarrow$	$\beta(s_1+a)P$
$\begin{cases} R_A = e(\beta(s_1+a)P, (s_1+a)^{-1}P) \\ K_A = \mathcal{H}_K(g^{\alpha} \; R_A)) \end{cases}$		$\begin{cases} R_B = e(\alpha(s_2+b)P, (s_2+b)^{-1}P) \\ K_B = \mathcal{H}_K(g^{\beta} \; R_B)) \end{cases}$

Table 7 7 McCullagh and Barreto's Authenticated Key Agreement Between Domains

with each other Importantly, we can now enable members of a domain who use Boneh and Franklin identity based key pairs (as used by Smart), communicate with other users who have Sakai and Kasahara identity based key pairs (as used by McCullagh and Barreto) This is easily accomplished as follows

Let Alice have a Boneh-Franklin identity based key pair, issued by KGC_1 That is Alice's public key is P_A, her private key is s_1P_A and the KGC's master public key is s_1P Her KGC specifies an appropriate bilinear map e_1 Bob has a Sakai and Kasahara key pair, issued by KGC_2 That is Bob's public key is $(s_2+b)Q$, and his private key is $(s_2+b)^{-1}Q$, where $b = \mathcal{H}_{ID}(ID_B) \in \mathbb{Z}_r^*$ His KGC specifies an appropriate bilinear map e_2 KGC_2 issues the point s_2Q The points P and Q may be totally unrelated and belong to different elliptic curves Let g_1 denote $e_1(P,P)$ and g_2 denote $e_2(P_A,P)$

Alice and Bob can execute the key agreement protocol as shown in Table 7 8

Alice		Bob
$\alpha(s_2+b)Q$	$\rightarrow$	
	$\leftarrow$	βP
$K_A = \mathcal{H}_K(g_2^{\alpha}, e_1(s_1P_A, \beta P))$		$K_B = \mathcal{H}_K(e_2(\alpha(s_2+b)Q, (s_2+b)^{-1}Q), e_1(P_A, s_1P)^{\beta})$
$K_A = \mathcal{H}_K(g_2^{\alpha}, g_1^{s_1\beta})$		$K_B = \mathcal{H}_K(g_2^{\alpha}, g_1^{s_1\beta})$

Table 7 8 A New Method for Key Agreement Between Domains

We now look at Chen and Kudla's third modification This modification is a generic modification, which they take from [23], and allows the transformation of any AK protocol into the corresponding AKC protocol This modification makes use of a new key derivation function of the form $\mathcal{H}_K \;\; \mu_r \rightarrow \{0,1\}^k \times \{0,1\}^k$ Because of the generic nature of this transformation we will only describe the key agreement stage in Table 7 9

$$
\begin{array}{ccc}
\text{Alice} & & \text{Bob} \\
R_A = \alpha(s+b)P & \rightarrow & \\
 & & R_{SK} = e(R_A, (s+b)^{-1}P)\ g^{\beta} \\
 & & (k, k') = \mathcal{H}(R_{SK}) \\
 & \leftarrow & \left\{ \begin{array}{c} R_B = \beta(s+a)P \\ M_1 = MAC_{k'}(2, b, a, R_A, R_B) \end{array} \right. \\
R_{SK} = e(R_B, (s+a)^{-1}P)\ g^{\alpha} & & \\
(k, k') = \mathcal{H}(R_{SK}) & & \\
M_2 = MAC_{k'}(3, a, b, R_A, R_B) & \rightarrow & \\
K_A = k & & K_B = k
\end{array}
$$

Table 7 9 McCullagh and Barreto Identity Based Key Agreement Protocol with Key Confirmation

7 6 3 Efficiency of the McCullagh and Barreto Identity Based Key Agreement Protocol

We have already seen the efficiency gains that Chen and Kudla manage to achieve over the scheme of Smart We now look at the efficiency gains that are made in the McCullagh and Barreto scheme Firstly, each participant in the scheme incurs one pairing, one point scalar multiplication and one pairing exponentiation The amount of computation incurred in the Chen and Kudla scheme is two point scalar multiplications and one pairing Therefore, in the popular setting of a $k = 2$ curve, our scheme will be faster than the Chen and Kudla scheme We note that their scheme can achieve the same level of performance as the McCullagh and Barreto scheme if there is enough storage to allow for precomputation The McCullagh and Barreto Key Agreement algorithm (the third of the three algorithms which comprise the scheme) does not appear to benefit from precomputation The only benefit seems to be the precomputation and storage of public keys We also note that to remove escrow we do exponentiation in the group μ_r whereas Chen and Kudla's modification of Smart's scheme does point scalar multiplication, so in common settings McCullagh and Barreto's protocol will again be faster

	Basic		Precomputation		Properties				
	Op count	Time	Op count	Time	KKS	PFS	KCIR	UKSR	KC
SOK	1p	172	-	-	○	○	○	●	●
Scott	1p+2pe	182	2pe	10	●	●⋆	○	●	●
Smart	2p+1psm+1pe	443	1p+1psm+1pe	271	●	●‡	●	●	●
C-K	1p+2psm	360	1p+1psm+1pe	271	●	●‡	●	●	●
M-B (ours)	1p+1psm+1pe	**271**	1p+1psm+1pe	**271**	●	●‡	●	●	●

Table 7 10 A Comparison of Key Agreement Protocols and their Claimed Properties

- Time is in milliseconds and is based on operation counts In reality times will be slower due to network constraints
- **KKS** Known Key Security
- **PFS** Partial Forward Secrecy
- **KCIR** Key Compromise Impersonation Resilience
- **UKSR** Unknown Key Share Resilience
- **KC** Key Control
- **Computational Cost**
  - **p** pairing operation
  - **psm** point scalar multiplication
  - **pe** pairing exponentiation operation
- ⋆ This scheme has full forward secrecy
- ‡ These schemes can be modified to have full forward secrecy

We note that while using precomputation the Smart, Chen and Kudla and McCullagh and Barreto algorithms require exactly the same computational cost, the McCullagh and Barreto scheme has no storage requirements, whereas Smart and Chen and Kudla both require storage of $\mu_{r_b}n$ bits, where n is the number of users with which we wish to perform key agreements and μ_{r_b} is the number of bits required to store one element in μ_r

7 7 Conclusion

The original work in the area of two party identity based key agreements from pairings was done by Sakai, Ohgishi and Kasahara in [111], and was improved upon by Smart in [127] Smart give heuristic arguments for the security of his scheme In this thesis, in a minor result, we prove the security of Smart's scheme, in the security model proposed in [23]

The work of Smart was improved upon by Chen and Kudla Chen and Kudla proposed a new key agreement which was faster than that proposed by Smart They also introduced, to identity based cryptography, the rigorous security frameworks of [19] and [23] which were originally designed for non-identity based public key cryptosystems This is a important contribution of their work

We then went on to describe the identity based key agreement protocol of McCullagh and Barreto This key agreement protocol manages to achieve the same performance without precomputation as the previous schemes only managed to achieve with precomputation We note that with precomputation Smart's scheme Chen and Kudla's scheme and McCullagh and Barreto's scheme all have similar performance characteristics This is illustrated in Table 7 10, along with the security properties that the various scheme are believed to possess

In another result we show how to agree a shared secret between users of an identity based system which uses Boneh and Franklin key pairs [31] and Sakai and Kasahara [109] key pairs

Chapter 8

Identity Based Signcryption

Two fundamental services of public key cryptography are confidentiality and authentication. Public key encryption schemes aim at providing confidentiality whereas digital signatures must provide authentication and non-repudiation. Nowadays, noticeably, many real-world cryptographic applications require these distinct goals to be achieved simultaneously. This motivated Zheng [146] to provide the cryptographer's toolbox with a novel cryptographic primitive which he called "signcryption." The purpose of this cryptographic primitive is to both encrypt and sign data in a single operation which has a computational cost less than that of doing both operations sequentially. Signcryption schemes should provide confidentiality as well as authentication and non-repudiation. As with conventional encryption schemes, recovering the plaintext from a signcrypted message must be computationally infeasible without the recipient's private key; as with conventional digital signature schemes, it must be computationally infeasible to create signcrypted texts without the sender's private key. The area of combining signature (or other authentication) with encryption has been extensively researched, see for example [147, 113, 6, 10, 11, 85].

8 1 Definition of an Identity Based Signcryption Scheme

The formal structure that we use for defining the security of our identity-based signcryption scheme is the following

Setup is a probabilistic algorithm run by a key generation centre (KGC) that takes as input a security parameter k, and outputs public parameters **params**, which are made public, and a master key mk that is kept secret by the KGC

KeyGen is a key generation algorithm run by the KGC on input of **params**, an identity ID and the master key mk, and outputs the private key S_{ID} associated with the identity ID

Sign/Encrypt is a probabilistic algorithm that takes as input public parameters **params**, a plaintext message m, the recipient's identity ID_B, the sender's private key S_{ID_A}, and outputs a ciphertext $\sigma = \text{Sign/Encrypt}(m, S_{ID_A}, ID_B)$

Decrypt/Verify is a deterministic decryption algorithm that takes as input a ciphertext σ public parameters **params**, the receiver's private key S_{ID_B} and (optionally)[1] a sender's identity ID_A before returning a valid message-signature pair (m, s) or a distinguished symbol $\perp$ if σ does not decrypt into a message bearing signer ID_A's signature

8 2 Properties of a Signcryption Scheme

The following, which were taken from [35] are some of the properties that we use to classify signcryption schemes

1 **Message Confidentiality** allows the communicating parties to preserve the secrecy of their exchange, if they choose to

[1]The senders identity may be sent as part of the ciphertext or may be recovered during the early stages of the Decrypt/Verify algorithm

2. **Signature non-repudiation:** makes it universally verifiable that a message speaks in the name of the signer (regardless of the ciphertext used to convey it, if any). This implies message authentication and integrity.

3. **Ciphertext unlinkability:** allows the sender to disavow creating a ciphertext for any given recipient, even though he or she remains bound to the valid signed message it contains.

4. **Ciphertext authentication:** allows the legitimate recipient, alone, to be convinced that the ciphertext and the signed message it contains were crafted by the same entity. This implies ciphertext integrity. It also reassures the recipient that the communication was indeed secured end-to-end.

5. **Ciphertext anonymity:** makes the ciphertext appear anonymous (hiding both the sender and the recipient identities) to anyone who does not possess the recipient decryption key.

Prior to the work of Barreto *et al.* [13], several identity-based signcryption algorithms had been proposed, e.g. [35, 45, 54, 81, 86, 98, 109, 142]. There is also an interesting hierarchical scheme [55]. Within this handful of results, only the authors of [35, 45, 54, 55, 81, 142] consider schemes supported by formal models and security proofs in the random oracle model [19]. Amongst them Chen and Malone-Lee's proposal [45] yields the most efficient construction.

In this chapter we outline some of the important advances in the development of identity based signcryption protocols. We introduce a designated verifier variant of the Malone-Lee's signcryption scheme, which resists the attack by Libert *et al.* on Malone-Lee's original scheme. We classify a new type of attack against some pairing based cryptosystems[2] and apply this attack to an identity based signcryption scheme by Sakai and Kasahara. We finish with the work of Barreto *et al.*, which was co-written by the author of this thesis.

[2] This was joint work by the author and Baretto.

We do a comparison of many important identity based signcryption protocols, in terms of properties and performance We see that our protocol is substantially faster than any of the competing schemes, whilst maintaining many desirable properties

8 3 Security Definitions for Identity Based Signcryption Schemes

Definition [35] An identity-based signcryption scheme (IBSC) satisfies the *message confidentiality* property (or adaptive chosen-ciphertext security IND-IBSC-CCA) if no PPT adversary, denoted $\mathcal{A}$, has a non-negligible advantage in the following game

1 The challenger runs the **Setup** algorithm on input of a security parameter k and sends the domain-wide parameters **params** to the $\mathcal{A}$

2 In a *find* stage, $\mathcal{A}$ queries the following oracles

- **KeyGen** returns private keys associated to arbitrary identities
- **Sign/Encrypt** given a pair of identities ID_A, ID_B and a plaintext m, this oracle returns an encryption under the receiver's identity ID_B of the message m signed in the name of the sender ID_A
- **Decrypt/Verify** given a pair of identities (ID_A, ID_B) and a ciphertext σ, it generates the receiver's private key S_{ID_B} = KeyGen(ID_B) and returns either a valid message-signature pair (m, s) for the sender's identity ID_A or the $\perp$ symbol if, under the private key S_{ID_B}, σ does not decrypt into a valid message-signature pair

3 $\mathcal{A}$ produces two plaintexts $m_0, m_1 \in \mathcal{M}$ and identities ID_A^* and ID_B^* She must not have extracted the private key of ID_B^* and she obtains C =Sign/Encrypt$(m_b, S_{ID_A}, ID_B^*, \mathbf{params})$ for a random a bit $b \xleftarrow{R} \{0,1\}$

4 In the *guess* stage, $\mathcal{A}$ asks new queries as in the *find* stage This time, she may not issue

a key extraction request on ID_B^* and she cannot submit C to the **Decrypt/Verify** oracle for the target identity ID_B^*

5 Finally, $\mathcal{A}$ outputs a bit b' and wins if $b' = b$

$\mathcal{A}$'s advantage is defined as $Adv(\mathcal{A}) = |2 \times \Pr[b' = b] - 1|$

The next definition, given in [35], considers non-repudiation with respect to signatures embedded in ciphertexts rather than with respect to ciphertexts themselves

Definition [35] An identity-based signcryption scheme (IBSC) is said to be *existentially signature-unforgeable* against adaptive chosen messages and ciphertexts attacks (ESUF-IBSC-CMA) if no PPT adversary can succeed in the following game with a non-negligible advantage

1 The challenger runs the **Setup** algorithm on input k and gives the params to the adversary $\mathcal{F}$

2 $\mathcal{F}$ issues a number of queries as in the previous definition

3 Finally, $\mathcal{F}$ outputs a triple $(\sigma^*, ID_A^*, ID_B^*)$ and wins the game if the sender's identity ID_A^* was not corrupted and if the result of the Decrypt/Verify oracle on the ciphertext σ^* under the private key associated to ID_B^* is a valid message-signature pair (m^*, S^*) such that no Sign/Encrypt query involved m^*, ID_A^* and some receiver ID_B' (possibly different from ID_B^*) and resulted in a ciphertext σ' whose decryption under the private key $S_{ID_B'}$ is the alleged forgery (m^*, s^*, ID_A^*)

The adversary's advantage is its probability of success in the above game

In both of these definitions, we consider insider attacks [6] Namely, in the definition of message confidentiality, the adversary is able to be challenged on a ciphertext created using a corrupted sender's private key, whereas in the notion of signature non-repudiation, the forger may output a ciphertext computed under a corrupted receiving identity

8.4 The Identity Based Signcryption Scheme of Malone-Lee

In [86] Malone-Lee introduced the first identity based signcryption scheme An important contribution of this work was formally redefining the existing notions of signcryption schemes for the identity based setting[3] His scheme has the same setup and extract algorithms as specified by Boneh and Franklin (see Sec 6 2) We only reproduce the Signcrypt and Unsigncrypt algorithms here

We assume that all participants to the protocol have access to hash functions $\mathcal{H}_1$ $\{0,1\}^* \to \mathbb{Z}_r^*$ and $\mathcal{H}_2$ $\mu_r \to \{0,1\}^n$ Where n is the length, in bits, of the message m

- **Signcryption** To perform signcryption to a user with public key Q_{ID_B} a sender, with key pair $\{Q_{ID_A}, sQ_{ID_A}\}$ generates a random $x \in \mathbb{Z}_r^*$ and computes the following values

$$U = xP \tag{8 1}$$

$$h = \mathcal{H}_1(U\|m) \tag{8 2}$$

$$V = hsQ_{ID_A} + xP_{pub} \tag{8 3}$$

$$C = \mathcal{H}_2(e(xP_{pub}, Q_{ID_B})) \oplus m \tag{8 4}$$

  The resulting ciphertext is the tuple (U, V, C)

- **Unsigncryption** To unsigncrypt the ciphertext (U, V, C) from the user with public key Q_{ID_A} a user with key pair $\{Q_{ID_B}, sQ_{ID_B}\}$ computes the following values

$$m \leftarrow \mathcal{H}_2(e(U, sQ_{ID_B})) \oplus C \tag{8 5}$$

$$h \leftarrow \mathcal{H}_1(U\|m) \tag{8 6}$$

[3] Although Malone-Lee's work was pioneering, the formal model that he favours now appears to be that of Boyen [46]

and then performs the following test

$$e(V, P) \stackrel{?}{=} e(hQ_{ID_A} + U, P_{pub}) \tag{8 7}$$

Malone-Lee compares his scheme with sequential use of both a Cha and Cheon signature scheme, followed by the Boneh and Franklin IBE scheme His scheme saves one μ_r exponentiation in the sign/encrypt stage, whilst trading two point scalar multiplications for a pairing and a μ_r exponentiation in the decrypt/verify stage (which take approximately the same time) We note that it is possible to turn this scheme into a designated verifier scheme, by computing $U = xQ_{ID}$ and $V = (h + x)sQ_{ID}$ rather than $hsQ_{ID} + xP_{pub}$ This achieves the indistinguishability of ciphertexts property at the cost of universal verification We do note however, that Malone-Lee's scheme does reduce the bandwidth of sending both encryption and signature separately, by one element in $\mathcal{G}$ and n bits, where n is the length of the message m in bits

If using the designated verifier variant we see that signcryption and unsigncryption now become

- **Signcryption** To perform signcryption to a user with public key Q_{ID_B} a sender, with key pair $\{Q_{ID_A}, sQ_{ID_A}\}$ generates a random $x \in Zr^*$ and computes the following values

$$U \leftarrow xQ_{ID_A} \tag{8 8}$$

$$h \leftarrow \mathcal{H}_1(U \| m) \tag{8 9}$$

$$V \leftarrow (x + h)sQ_{ID_A} \tag{8 10}$$

$$C \leftarrow \mathcal{H}_2(e(xsQ_{ID_A}, Q_{ID_B})) \oplus m \tag{8 11}$$

The resulting ciphertext is the tuple $(U\ V, C)$

- **Unsigncryption** To unsigncrypt the ciphertext (U, V, C) from the user with public key Q_{ID_A} a user with key pair $\{Q_{ID_B}, sQ_{ID_B}\}$ computes the following values

$$m \leftarrow \mathcal{H}_2(U, sQ_{ID_B}) \oplus C \quad (8\ 12)$$

$$h \leftarrow \mathcal{H}_1(U \| m) \quad (8\ 13)$$

and then performs the following test

$$e(V, Q_{ID_B}) \stackrel{?}{=} e(hQ_{ID_A} + U, sQ_{ID_B}) \quad (8\ 14)$$

8 4 1 Security of Malone-Lee's Signcryption Scheme

Malone-Lee defines the notion of *indistinguishability of identity-based signcryptions under chosen ciphertext attack* However, as Libert and Quisquater point out, Malone-Lee's scheme, as specified, does not have this property This is because in the original scheme the ciphertext contains the signature on the plaintext Given a ciphertext U, V, C and a message $m \in \{m_0, m_1\}$, the message can be determined as follows

$$h \leftarrow \mathcal{H}_1(U \| m_0) \quad (8\ 15)$$

$$e(V, P) \stackrel{?}{=} e(hQ_{ID_A} + U, P_{pub}) \quad (8\ 16)$$

If the equation verifies then the message was m_0 otherwise it was m_1

We note that this is not the case with our designated verifier variant, since the value sQ_{ID_B} (the receiver's private key) is not a publicly available value, whereas the value P_{pub} is However, the ciphertext can only be verified by the intended receiver, and therefore has lost its universal verifiability property For the purposes of non-repudiation the receiver would have to surrender her private key, which is a poor result Indeed as Shin *et al* [124] point out, universal verifiability hampers resistance to chosen ciphertext attack

8 5 The Identity Based Signcryption Scheme of Sakai and Kasahara

We now look at the Sakai and Kasahara identity based signcryption scheme They call this scheme an 'ID-Based Public Key Cryptosystem with Authentication" in [109] The paper introduces a number of efficient schemes However, with current knowledge, these schemes can only be implemented using the Weil pairing and so, although they require fewer pairings, they are not actually more efficient The paper is quite complex to understand, but it is an extremely important paper as this is the paper in which Sakai and Kasahara introduced their new identity based key pair

Contrary to other methods, the Sakai and Kasahara signcryption scheme depends on the availability of a pairing $e \colon \mathcal{G}_1 \times \mathcal{G}_2 \to \mu_r$ where $\mathcal{G}_1$ and $\mathcal{G}_2$ are two distinct subgroups We denote $\mathcal{G}_1 = \langle P \rangle$ and $\mathcal{G}_2 = \langle Q \rangle$ Importantly, it also requires a pairing $e \colon \mathcal{G}_1 \times \mathcal{G}_2 \to \mu_r$, such that $e(P + Q, Q) = e(P, Q)$ and $e(P + Q, P) = e(Q, P)$, which implies it can only be instantiated using the Weil pairing Let $g = e(P, Q) = \langle \mu_r \rangle \in \mu_r$

- **Setup** The KGC generates a random secret polynomial $s(x) = \sum_{i=0}^{d} s_i x^i \in \mathbb{Z}_r[x]$ which acts as its private master key The simplest choice is $d = 1$, $s_1 = 1$, so the secret key reduces to the single $\mathbb{Z}_r^*$ value s_0 The KGC publishes the points P, Q, $g = e(P, Q)$, and $s_i Q$ for $i = 0, \ldots, d$ It also publishes descriptions of two hash functions $\mathcal{H}_0 \colon \{0,1\}^* \to \mathbb{Z}_r^*$ and $\mathcal{H}_1 \colon \mu_r \to \{0,1\}^*$

- **KeyGen** A user identity is a public element $u \in \mathbb{Z}_r^*$ The KGC computes a user's private key as $P_u = s(u)^{-1} P$, where the inverse is computed modulo r The corresponding public key can be (publicly) computed from u and the points $s_i Q$ as $Q_u = \sum_{i=0}^{d} u^i (s_i Q) = s(u) Q$ Let Alice's identity be a and Bob's identity be b

- **Sign/Encrypt** To signcrypt a message m to Bob, Alice generates a random integer

$x \in \mathbb{Z}_r^*$ and computes

$$R = g^x \tag{8 17}$$

$$h = \mathcal{H}_0(m) \tag{8 18}$$

$$c = \mathcal{H}_1(R^{(1+h)}) \oplus m \tag{8 19}$$

$$S = x(hP_a + Q_b) \tag{8 20}$$

The signcrypted message is (c, S)

- **Decrypt/Verify** Upon reception of the above pair, Bob computes

$$R = e(P_b, S) \tag{8 21}$$

$$W = e(S, Q_a) \tag{8 22}$$

$$m = \mathcal{H}_1(RW) \oplus c \tag{8 23}$$

$$h = \mathcal{H}_0(m) \tag{8 24}$$

Bob then verifies that $W = R^h$

8 5 1 An Attack on Sakai and Kasahara's Signcryption Scheme

The scheme proposed by Sakai and Kasahara makes it possible to distinguish between a number of possible plaintexts given only the ciphertext, the public identity of the sender, and the KGC's public key This also happens in Malone-Lee's scheme, as pointed out by Libert and Quisquater [81]

The attack we now describe against Sakai and Kasahara's scheme is a variant of the attack of Libert and Quisquater against Malone-Lee's scheme and proceeds as follows The ciphertext is (c, S) We assume that Carol knows that the plaintext m that Alice sent to Bob is one of the messages in a set $\{m_0, m_1\}$ Carol computes $W \leftarrow e(S, Q_a)$ and then

$$h_0 \leftarrow \mathcal{H}_0(m_0) \tag{8 25}$$

$$R_0 \leftarrow W^{h_0^{-1} \bmod r} \tag{8 26}$$

And then the test

$$c \stackrel{?}{=} \mathcal{H}_1(R_0 W) \oplus m_0 \tag{8 27}$$

If the equation validates then the message m is equal to m_0, otherwise it is equal to m_1 Therefore, the signcryption scheme of Sakai and Kasahara does not satisfy the IND-IDSC-CCA (*indistinguishability of signcryptions*) property

8 5 2 Projection Attacks Against the Sakai and Kasahara Signcryption Scheme

The original description of the scheme by Sakai and Kasahara does not impose any restriction upon the groups over which it is defined, assuming only the existence of a bilinear, non-degenerate, efficiently computable pairing on those groups

As it turns out, the group choice seriously affects the security of the Sakai and Kasahara scheme, in the sense that the scheme structure implicitly uses the relationship between $\langle P \rangle$ and $\langle Q \rangle$ for the security purpose of concealing the signer's private key In particular, when implemented on a large class of groups where the Tate or Weil pairing is especially efficient, it allows the recipient of a signcrypted message to obtain sufficient information to impersonate the sender as we show next

Definition The *Frobenius endomorphism* is the mapping Φ $E(\mathbb{F}_{p^k}) \to E(\mathbb{F}_{p^k})$, $(X, Y) \mapsto (X^p, Y^p)$

Definition The *trace map* is the mapping tr $E(\mathbb{F}_{p^k}) \to E(\mathbb{F}_p)$ defined as $\mathrm{tr}(P) = P + \Phi(P) + \Phi^2(P) + \quad + \Phi^{k-1}(P)$

We see that $\mathrm{tr}(\Phi(P)) = \Phi(\mathrm{tr}(P)) = \mathrm{tr}(P)$ for any $P \in E(\mathbb{F}_{p^k})$

Definition The *trace-zero subgroup* or *trace kernel* is the subgroup $\mathcal{T} = \{Q \in E(\mathbb{F}_{p^k}) \mid \mathrm{tr}(Q) = \mathcal{O}\}$

The following maps

$$\pi_0 \quad E(\mathbb{F}_{p^k}) \to \mathcal{T},\ \pi_0(Q) = Q - k^{-1}\,\mathrm{tr}(Q), \tag{8 28}$$

$$\pi_1 \quad E(\mathbb{F}_{p^k}) \to E(\mathbb{F}_p),\ \pi_1(P) = k^{-1}\,\mathrm{tr}(P), \tag{8 29}$$

where k^{-1} is computed modulo r, satisfy $\pi_0(Q) = Q$ for any $Q \in \mathcal{T}$ and $\pi_1(P) = P$ for any $P \in E(\mathbb{F}_p)[r]$ Notice that any point $R \in E(\mathbb{F}_{p^k})[r]$ can be written $R = \pi_0(R) + \pi_1(R)$

With these tools, we can mount a forgery attack against the Sakai and Kasahara scheme The crucial assumption is that the KGC chooses a point $Q \in \mathcal{T}$ (the trace zero subgroup) This is the case if the implementation is based on certain supersingular curves as described in [14, 67, 68] (such as curves of form $y^2 = x^3 + ax$ over $\mathbb{F}_p$ with $p \equiv 3 \pmod 4$, or curves of the form $y^2 = x^3 - x \pm 1$ over $\mathbb{F}_{3^m}$), or ordinary curves as suggested in [16] These are all popular choices, as they favour efficient implementation of the Tate or Weil pairing as well as other arithmetic operations[4]

The basic attack allows the legitimate receiver of a signcrypted message to fake other signcryptions from the same sender This attack proceeds as follows Bob unsigncrypts the received message (c, S), obtaining R and h Let m' be the message he wants to pretend was

[4]However, since we are using the Weil pairing we do not have to use the trace zero group, we can pick P and Q as generators of any two linearly independent subgroups of order r in $\mathbb{F}_{p^k}$

sent by Alice He computes

$$U \leftarrow h^{-1}\pi_1(S) \; [= xP_a] \tag{8 30}$$

$$V \leftarrow \pi_0(S) \; [= rQ_b] \tag{8 31}$$

$$h' \leftarrow \mathcal{H}_0(m') \tag{8 32}$$

$$c' \leftarrow \mathcal{H}_1(R^{1+h'}) \oplus m' \tag{8 33}$$

$$S' \leftarrow h'U + V \tag{8 34}$$

Now Bob can use the pair (c', S') as evidence that Alice sent him m' rather than m He can even further disguise his ruse by using a different x, say $x' = \alpha x$ All he has to do is to set $R' \leftarrow R^{\alpha}$, $U' \leftarrow \alpha U$, and $V' \leftarrow \alpha V$ and use these values instead

This attack is especially annoying because, if the plaintext of any signcrypted message m from Alice to Bob is compromised, then a third party, Carol, can impersonate Alice and forge new signcrypted messages to Bob Carol simply computes $h \leftarrow \mathcal{H}_0(m)$, $R = e(h^{-1}S, Q_a)$, and proceeds as above We see that, in fact, Carol needs only h, not m itself

8 6 The Identity Based Signcryption Scheme of Barreto *et al.*

We now look at the signcryption scheme of Barreto, Libert, McCullagh and Quisquater (BLMQ), to be presented at Asiacrypt '05 [13] Unlike recent works of [35, 45] that present two-layer designs of probabilistic signature followed by a deterministic encryption, our construction is a single-layer construction jointly achieving signature and encryption on one side and decryption and verification on the other Although the description of our scheme could be modified to fit a two-layer formalism, we kept the monolithic presentation without hampering the non-repudiation property as, similar to [35, 45], our construction enables an ordinary signature on the plaintext to be extracted from any properly formed ciphertext using the recipient's private key The extracted message-signature pair can be forwarded to any third party in such a way that a sender remains committed to the content of the plaintext

Unlike models of [35, 45] that consider anonymous ciphertexts, the above assumes that senders' identities are sent in the clear along with ciphertexts Actually, receivers do not need to have any a priori knowledge as to from whom the ciphertext emanates in our scheme but this simply allows more efficient reductions in the security proofs A simple modification of our scheme yields anonymous ciphertexts and enables senders' identities to be recovered by the **Decrypt/Verify** algorithm (which only then takes a ciphertext and the recipient's private key as input)

8 6 1 The BLMQ Signcryption Scheme

Setup Given k, the PKG chooses bilinear map groups $(\mathcal{G}_1, \mathcal{G}_2, \mu_r)$ of prime order $r > 2^k$ and generators $Q \in \mathcal{G}_2$, $P = \psi(Q) \in \mathcal{G}_1$, where ψ is an efficiently computable distortion map from $\mathcal{G}_2$ to $\mathcal{G}_1$, $g = e(P, Q) \in \mu_r$ It then chooses a random master key $s \in \mathbb{Z}_r^*$, a system-wide public key $Q_{pub} = sQ \in \mathcal{G}_2$ and hash functions $H_1 \ \{0,1\}^* \to \mathbb{Z}_r^*$, $H_2 \ \{0,1\}^* \times \mu_r \to \mathbb{Z}_r^*$ and $H_3 \ \mu_r \to \{0,1\}^n$ The public parameters are

$$\texttt{params} = \{\mathcal{G}_1, \mathcal{G}_2, \mu_r, P, Q, g, Q_{pub}, e, \psi, H_1, H_2, H_3\}$$

KeyGen for an identity ID, the private key is $S_{ID} = \frac{1}{H_1(ID)+s} Q \in \mathcal{G}_2$

Sign/Encrypt given a message $m \in \{0,1\}^n$, a receiver's identity ID_B and a sender's private key S_{ID_A},

1 Pick a random $x \in \mathbb{Z}_r^*$, compute $R = g^x$ and $c = m \oplus H_3(R) \in \{0,1\}^n$

2 Set $h = H_2(m, R) \in \mathbb{Z}_r^*$

3 Compute $S = (x + h)\psi(S_{ID_A})$

4 Compute $T = x(H_1(ID_B)P + \psi(Q_{pub}))$

The ciphertext is $\sigma = \langle c, S, T\rangle \in \{0,1\}^n \times \mathcal{G}_1 \times \mathcal{G}_1$

Decrypt/Verify given $\sigma = \langle c\ S, T\rangle$ and some sender's identity ID_A,

1 Compute $R = e(T, S_{ID_B})$, $m = c \oplus H_3(R)$, and $h = H_2(m, R)$

2 Accept the message iff $R = e(S, H_1(ID_A)Q + Q_{pub})g^{-h}$ If this condition holds, return the message m together with the signature $(h, S) \in \mathbb{Z}_r^* \times \mathcal{G}_1$

If required, the anonymity property is obtained by scrambling the sender's identity ID_A together with the message at step 1 of Sign/Encrypt in such a way that the recipient retrieves it at the first step of the reverse operation This change does not imply any computational penalty in practice but induces more expensive security reductions In order for the proof to hold, ID_A must be appended to the inputs of H_2

8 6 2 Security results

The following theorems prove the security of the scheme in the random oracle model under the same irreflexivity assumption[5] as Boyen's scheme [35] the Sign/Encrypt algorithm is assumed to always take distinct identities as inputs (in other words, a principal never encrypts a message bearing his signature using his own identity)

Theorem 8 6 1 *Assume that an IND-IDSC-CCA adversary $\mathcal{A}$ has an advantage ϵ against our scheme when running in time τ, asking q_{h_i} queries to random oracles H_i $(i = 1, 2, 3)$, q_{se} signature/encryption queries and q_{dv} queries to the decryption/verification oracle Then there is an algorithm $\mathcal{B}$ to solve the q-BDHIP for $q = q_{h_1}$ with probability*

$$\epsilon' > \frac{\epsilon}{q_{h_1}(2q_{h_2} + q_{h_3})}\left(1 - q_{se}\frac{q_{se} + q_{h_2}}{2^k}\right)\left(1 - \frac{q_{dv}}{2^k}\right)$$

[5] Irreflexivity assumption A term coined by Boyen meaning that the sender and reciever identities cannot be the same

within a time $\tau' < \tau + O(q_{se} + q_{dv})\tau_p + O(q_{h_1}^2)\tau_{mult} + O(q_{dv}q_{h_2})\tau_{exp}$ where τ_{exp} and τ_{mult} are respectively the costs of an exponentiation in $\mathcal{G}_T$ and a multiplication in $\mathcal{G}_2$ whereas τ_p is the complexity of a pairing computation.

Proof. Algorithm $\mathcal{B}$ takes as input $\langle P, Q, \alpha Q, \alpha^2 Q, \ldots, \alpha^q Q\rangle$ and attempts to extract $e(P,Q)^{1/\alpha}$ from its interaction with $\mathcal{A}$.

In a preparation phase, $\mathcal{B}$ selects $\ell \xleftarrow{R} \{1, \ldots, q_{\mathcal{H}_W}\}$, elements $I_\ell \xleftarrow{R} \mathbb{Z}_p^*$ and $w_1, \ldots, w_{\ell-1}, w_{\ell+1}, \ldots, w_q \xleftarrow{R} \mathbb{Z}_p^*$. For $i = 1, \ldots, \ell-1, \ell+1, \ldots, q$, it computes $I_i = I_\ell - w_i$. As in the technique of [28] and in lemma 5.5.2, it sets up generators $G_2 \in \mathcal{G}_2$, $G_1 = \psi(G_2) \in \mathcal{G}_1$ and another $\mathcal{G}_2$ element $U = \alpha G_2$ such that it knows $q-1$ pairs $(w_i, H_i = (1/(w_i + \alpha))G_2)$ for $i \in \{1, \ldots, q\}\backslash\{\ell\}$. The system-wide public key Q_{pub} is chosen as

$$Q_{pub} = -U - I_\ell G_2 = (-\alpha - I_\ell)G_2$$

so that its (unknown) private key is implicitly set to $x = -\alpha - I_\ell \in \mathbb{Z}_p^*$. For all $i \in \{1, \ldots, q\}\backslash\{\ell\}$, we have $(I_i, -H_i) = (I_i, (1/(I_i + x))G_2)$.

$\mathcal{B}$ then initializes a counter ν to 1 and starts $\mathcal{A}$ on input of (G_1, G_2, Q_{pub}). Throughout the game, we assume that $\mathcal{H}_W$-queries are distinct, that the target identity ID_B^* is submitted to $\mathcal{H}_W$ at some point and that any query involving an identity ID comes after a $\mathcal{H}_W$-query on ID.

- $\mathcal{H}_W$-queries (let us call ID_ν the input of the ν^{th} one of such queries). $\mathcal{B}$ answers I_ν and increments ν.

- $\mathcal{H}_{\mu_r}$-queries on input (M, r). $\mathcal{B}$ returns the defined value if it exists and a random $\mathcal{H}_{\mu_r} \xleftarrow{R} \mathbb{Z}_p^*$ otherwise. To anticipate possible subsequent Decrypt/Verify requests, $\mathcal{B}$ additionally simulates random oracle H_3 on its own to obtain $h_3 = H_3(r) \in \{0,1\}^n$ and stores the information $(M, r, \mathcal{H}_{\mu_r}, c = M \oplus h_3, \gamma = r \cdot e(G_1, G_2)^{\mathcal{H}_{\mu_r}})$ in L_2.

- H_3-queries for an input $r \in \mathcal{G}_T$. $\mathcal{B}$ returns the previously assigned value if it exists and a random $h_3 \xleftarrow{R} \{0,1\}^n$ otherwise In the latter case, the input r and the response h_3 are stored in a list L_3

- KeyGen queries on an input ID_ν if $\nu = \ell$, then $\mathcal{B}$ fails Otherwise, it knows that $\mathcal{H}_W(ID_\nu) = I_\nu$ and returns $-H_\nu = (1/(I_\nu + x))\, G_2 \in \mathcal{G}_2$

- Sign/Encrypt queries for a plaintext M and identities $(ID_A, ID_B) = (ID_\mu, ID_\nu)$ for $\mu, \nu \in \{1, \quad , q_{\mathcal{H}_W}\}$ we observe that, if $\mu \neq \ell$, $\mathcal{B}$ knows the sender s private key $S_{ID_\mu} = -H_\mu$ and can answer the query according to the specification of Sign/Encrypt We thus assume $\mu = \ell$ and hence $\nu \neq \ell$ by the irreflexivity assumption Observe that $\mathcal{B}$ knows the receiver's private key $S_{ID_\nu} = -H_\nu$ by construction The difficulty is to find a random triple $(S, T, h) \in \mathcal{G}_1 \times \mathcal{G}_1 \times \mathbb{Z}_p^*$ for which

$$e(T\ S_{ID_\nu}) = e(S, Q_{ID_\ell}) e(G_1, G_2)^{-h} \tag{8 35}$$

where $Q_{ID_\ell} = I_\ell G_2 + Q_{pub}$ To do so, $\mathcal{B}$ randomly chooses $t, h \xleftarrow{R} \mathbb{Z}_p^*$ and computes $S = t\psi(S_{ID_\nu}) = -t\psi(H_\nu)$, $T = t\psi(Q_{ID_\ell}) - h\psi(Q_{ID_\nu})$ where $Q_{ID_\nu} = I_\nu G_2 + Q_{pub}$ in order to obtain the desired equality $r = e(T, S_{ID_\nu}) = e(S, Q_{ID_\ell}) e(G_1, G_2)^{-h} = e(\psi(S_{ID_\nu}), Q_{ID_\ell})^t e(G_1, G_2)^{-h}$ before patching the hash value $\mathcal{H}_{\mu_r}(M, r)$ to h ($\mathcal{B}$ fails if $\mathcal{H}_{\mu_r}$ is already defined but this only happens with probability $(q_{se} + q_{\mathcal{H}_{\mu_r}})/2^k$) The ciphertext $\sigma = \langle M \oplus H_3(r), S, T \rangle$ is returned

- Decrypt/Verify queries on a ciphertext $\sigma = \langle c, S, T \rangle$ for identities $(ID_A, ID_B) = (ID_\mu, ID_\nu)$ we assume that $\nu = \ell$ (and hence $\mu \neq \ell$ by the irreflexivity assumption), because otherwise $\mathcal{B}$ knows the receiver's private key $S_{ID_\nu} = -H_\nu$ and can normally run the Decrypt/Verify algorithm Since $\mu \neq \ell$, $\mathcal{B}$ has the sender's private key S_{ID_μ} and also knows that, for all valid ciphertexts, $\log_{S_{ID_\mu}}(\psi^{-1}(S) - hS_{ID_\mu}) = \log_{\psi(Q_{ID_\nu})}(T)$, where $h = \mathcal{H}_{\mu_r}(M, r)$ is the hash value obtained in the Sign/Encrypt algorithm and

$Q_{ID_\nu} = I_\nu G_2 + Q_{pub}$ Hence, we have the relation

$$e(T, S_{ID_\mu}) = e(\psi(Q_{ID_\nu}), \psi^{-1}(S) - hS_{ID_\mu}) \tag{8 36}$$

which yields $e(T, S_{ID_\mu}) = e(\psi(Q_{ID_\nu}), \psi^{-1}(S))e(\psi(Q_{ID_\nu}), S_{ID_\mu})^{-h}$ We observe that the latter equality can be tested without inverting ψ as $e(\psi(Q_{ID_\nu}), \psi^{-1}(S)) = e(S, Q_{ID_\nu})$ The query is thus handled by computing $\gamma = e(S, Q_{ID_\mu})$, where $Q_{ID_\mu} = I_\mu G_2 + Q_{pub}$, and searching through list L_2 for entries of the form $(M_i, r_i, h_{2,i}, c, \gamma)$ indexed by $i \in \{1, \quad , q_{\mathcal{H}_{\mu_r}}\}$ If none is found, σ is rejected Otherwise, each one of them is further examined for the corresponding indexes, $\mathcal{B}$ checks if

$$e(T, S_{ID_\mu})/e(S\ Q_{ID_\nu}) = e(\psi(Q_{ID_\nu}), S_{ID_\mu})^{-h_{2\ i}} \tag{8 37}$$

(the pairings are computed only once and at most $q_{\mathcal{H}_{\mu_r}}$ exponentiations are needed), meaning that (8 36) is satisfied If the unique $i \in \{1, \quad , q_{\mathcal{H}_{\mu_r}}\}$ satisfying (8 37) is detected, the matching pair $(M_i, \langle h_{2,i}, S\rangle)$ is returned Otherwise, σ is rejected Overall, an inappropriate rejection occurs with probability smaller than $q_{dv}/2^k$ across the whole game

At the challenge phase, $\mathcal{A}$ outputs messages (M_0, M_1) and identities (ID_A, ID_B) for which she never obtained ID_B's private key If $ID_B \neq ID_\ell$, $\mathcal{B}$ aborts Otherwise, it picks $\xi \stackrel{R}{\leftarrow} \mathbb{Z}_p^*$, $c \stackrel{R}{\leftarrow} \{0,1\}^n$ and $S \stackrel{R}{\leftarrow} \mathcal{G}_1$ to return the challenge $\sigma^* = \langle c, S, T\rangle$ where $T = -\xi G_1 \in \mathcal{G}_1$ If we define $\rho = \xi/\alpha$ and since $x = -\alpha - I_\ell$, we can check that

$$T = -\xi G_1 = -\alpha\rho G_1 = (I_\ell + x)\rho G_1 = \rho I_\ell G_1 + \rho\psi(Q_{pub}) \tag{8 38}$$

$\mathcal{A}$ cannot recognize that σ^* is not a proper ciphertext unless she queries $\mathcal{H}_{\mu_r}$ or H_3 on $e(G_1, G_2)^\rho$ At the guess stage, her view is simulated as before and her eventual output is ignored Standard arguments can show that a successful $\mathcal{A}$ is very likely to query $\mathcal{H}_{\mu_r}$ or H_3 on the input $e(G_1, G_2)^\rho$ if the simulation is indistinguishable from a real attack environment

To produce a result, $\mathcal{B}$ fetches a random entry $(M, r, \mathcal{H}_{\mu_r}, c, \gamma)$ or $\langle r, \ \rangle$ from the lists L_2 or L_3 With probability $1/(2q_{\mathcal{H}_{\mu_r}} + q_{h_3})$ (as L_3 contains no more than $q_{\mathcal{H}_{\mu_r}} + q_{h_3}$ records by construction), the chosen entry will contain the right element $r = e(G_1, G_2)^{\rho} = e(P, Q)^{f(\alpha)^2 \xi/\alpha}$, where $f(z) = \sum_{i=0}^{q-1} c_i z^i$ is the polynomial for which $G_2 = f(\alpha)Q$ The q-BDHIP solution can be extracted by noting that, if $\gamma^* = e(P, Q)^{1/\alpha}$, then

$$e(G_1, G_2)^{1/\alpha} = \gamma^{*(c_0^2)} e\Big(\sum_{i=0}^{q-2} c_{i+1}(\alpha^i P), c_0 Q\Big) e\Big(G_1, \sum_{j=0}^{q-2} c_{j+1}(\alpha^j)Q\Big)$$

In an analysis of $\mathcal{B}$'s advantage, we note that it only fails in providing a consistent simulation because one of the following independent events

E_1 $\mathcal{A}$ does not choose to be challenged on ID_{ℓ}

E_2 $\mathcal{B}$ aborts in a Sign/Encrypt query because of a collision on $\mathcal{H}_{\mu_r}$

E_3 $\mathcal{B}$ rejects a valid ciphertext at some point of the game

We clearly have $\Pr[\neg E_1] = 1/q_{\mathcal{H}_W}$ and we already observed that $\Pr[E_2] \le q_{se}(q_{se} + q_{\mathcal{H}_{\mu_r}})/2^k$ and $\Pr[E_3] \le q_{dv}/2^k$ We thus find that

$$\Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3] \ge \frac{1}{q_{\mathcal{H}_W}} \left(1 - q_{se} \frac{q_{se} + q_{\mathcal{H}_{\mu_r}}}{2^k}\right) \left(1 - \frac{q_{dv}}{2^k}\right)$$

We obtain the announced bound by noting that $\mathcal{B}$ selects the correct element from L_2 or L_3 with probability $1/(2q_{\mathcal{H}_{\mu_r}} + q_{h_3})$ Its workload is dominated by $O(q_{\mathcal{H}_W}^2)$ multiplications in the preparation phase, $O(q_{se} + q_{dv})$ pairing calculations and $O(q_{dv} q_{\mathcal{H}_{\mu_r}})$ exponentiations in $\mathcal{G}_T$ in its emulation of the Sign/Encrypt and Decrypt/Verify oracles □

Theorem 8 6 2 *Assume there exists an ESUF-IBSC-CMA attacker $\mathcal{A}$ that makes q_{h_i} queries to random oracles H_i ($i = 1, 2, 3$), q_{se} signature/ encryption queries and q_{dv} queries to the decryption/verification oracle Assume also that, within a time τ, $\mathcal{A}$ produces a forgery with probability $\epsilon \ge 10(q_{se} + 1)(q_{se} + q_{h_2})/2^k$ Then, there is an algorithm $\mathcal{B}$ that is* *able to solve the q-SDHP for $q = q_{h_1}$ in expected time*

$$\tau' \leq 120686 q_{h_1} q_{h_2} \frac{\tau + O((q_{se} + q_{dv})\tau_p) + q_{dv} q_{h_2} \tau_{exp}}{\epsilon(1 - 1/2^k)(1 - q/2^k)} + O(q^2 \tau_{mult})$$

where τ_{mult}, τ_{exp} and τ_p denote the same quantities as in theorem 8.6.1.

Proof. The proof is similar to the one of theorem **??**. Namely, it shows that a forger in the ESUF-IBSC-CMA game implies a forger in a chosen-message and *given* identity attack. Using the forking lemma [103, 104], the latter is in turn shown to imply an algorithm to solve the q-Strong Diffie-Hellman problem. More precisely, queries to the Sign/Encrypt and Decrypt/Verify oracles are answered as in the proof of theorem 8.6.1 and, at the outset of the game, the simulator chooses public parameters in such a way that it can extract private keys associated to any identity but the one which is given as a challenge to the adversary. By doing so, thanks to the irreflexivity assumption, it is able to extract clear message-signature pairs from ciphertexts produced by the forger (as it knows the private key of the receiving identity ID_B^*). □

We now restate theorem 8.6.1 for the variant of our scheme with anonymous ciphertexts. The simulator's worst-case running time is affected by the fact that, when handling **Decrypt/Verify** requests, senders' identities are not known in advance. The reduction involves a number of pairing calculations which is quadratic in the number of adversarial queries.

Theorem 8.6.3. *Assume that an IND-IDSC-CCA adversary $\mathcal{A}$ has an advantage ϵ against our scheme when running in time τ, asking q_{h_i} queries to random oracles H_i ($i = 1, 2, 3$), q_{se} signature/encryption queries and q_{dv} queries to the decryption/verification oracle. Then there is an algorithm $\mathcal{B}$ to solve the q-BDHIP for $q = q_{h_1}$ with probability*

$$\epsilon' > \frac{\epsilon}{q_{h_1}(2q_{h_2} + q_{h_3})} \left(1 - q_{se}\frac{q_{se} + q_{h_2}}{2^k}\right)\left(1 - \frac{q_{dv}}{2^k}\right)$$

within a time $\tau' < \tau + O(q_{se} + q_{dv} q_{h_2})\tau_p + O(q_{h_1}^2)\tau_{mult} + O(q_{dv} q_{h_2})\tau_{exp}$ where τ_{exp}, τ_{mult} and

τ_p *denote the same quantities as in previous theorems*

Proof The simulator is the same as in theorem 8 6 1 with the following differences (recall that senders' identities are provided as inputs to $\mathcal{H}_{\mu_r}$)

- $\mathcal{H}_{\mu_r}$-queries on input (ID_A, M, r) $\mathcal{B}$ returns the previously defined value if it exists and a random $\mathcal{H}_{\mu_r} \xleftarrow{R} \mathbb{Z}_p^*$ otherwise To anticipate subsequent Decrypt/Verify requests, $\mathcal{B}$ simulates oracle H_3 to obtain $h_3 = H_3(r) \in \{0,1\}^{n+n_0}$ (where n_0 is the maximum length of identity strings) and stores $(ID_A, M, r, \mathcal{H}_{\mu_r}, c = (M\|ID_S ID_A) \oplus h_3, \gamma = r\ e(G_1, G_2)^{\mathcal{H}_{\mu_r}})$ in list L_2

- Decrypt/Verify queries given a ciphertext $\sigma = \langle c, S, T\rangle$ and a receiver's identity $ID_B = ID_\nu$ we assume that $\nu = \ell$ because otherwise $\mathcal{B}$ knows the receiver's private key The simulator $\mathcal{B}$ does not know the sender's identity ID_A but knows that $ID_A \neq ID_\nu$ It also knows that, for the private key S_{ID_S}, $\log_{S_{ID_S}}(\psi^{-1}(S) - hS_{ID_S}) = \log_{\psi(Q_{ID_\nu})}(T)$, and hence

$$e(T, S_{ID_S}) = e(\psi(Q_{ID_\nu}), \psi^{-1}(S) - hS_{ID_S}), \qquad (8\ 39)$$

where $h = \mathcal{H}_{\mu_r}(ID_A, M, r)$ is the hash value obtained in the Sign/ Encrypt algorithm and $Q_{ID_\nu} = I_\nu G_2 + Q_{pub}$ The query is handled by searching through list L_2 for entries of the form $(ID_{S,i}, M_i, r_i, h_{2,i}\ c, \gamma_i)$ indexed by $i \in \{1, \quad , q_{\mathcal{H}_{\mu_r}}\}$ If none is found, the ciphertext is rejected Otherwise, each one of these entries for which $ID_{S,i} \neq ID_\nu$ is further examined by checking whether $\gamma_i = e(S, \mathcal{H}_W(ID_{S,i})Q + Q_{pub})$ and

$$e(T, S_{ID_S\,i})/e(S, Q_{ID_\nu}) = e(\psi(Q_{ID_\nu}), S_{ID_S\,i})^{-h_{2\,i}} \qquad (8\ 40)$$

(at most $3q_{\mathcal{H}_{\mu_r}} + 1$ pairings and $q_{\mathcal{H}_{\mu_r}}$ exponentiations must be computed), meaning that equation (8 39) is satisfied and that the ciphertext contains a valid message signature pair if both relations hold If $\mathcal{B}$ detects an index $i \in \{1, \quad , q_{\mathcal{H}_{\mu_r}}\}$ satisfying them, the matching pair $(M_i, \langle h_{2,i}, S\rangle)$ is returned Otherwise, σ is rejected and such a wrong rejection again occurs with an overall probability smaller than $q_{dv}/2^k$

□

Theorem 8 6 2 can be similarly restated as its reduction cost is affected in the same way

8 7 Conclusion

In this section we have looked at a number of signcryption schemes The properties that various signcryption schemes offer are quite varied, and the term "signcryption" can only be loosely defined in reality There is still debate over which properties are advantageous, and this probably comes down to the requirements of the individual application There are several identity and non-identity based signcryption schemes, including a non-identity based signcryption scheme broken by the author of this thesis in a personal communication with its authors [71]

In this review we have concentrated on identity based signcryption schemes We have introduced a new signcryption scheme based on the identity based key pair of Sakai and Kasahara and we note that in performance terms it ranks well with its peers This is demonstrated in Table 8 1

	Sign/Encrypt				Decrypt/Verify			
signcryption scheme	exp	mul	pairings	time (ms)	exp	mul	pairings	time (ms)
Boyen	1	3	1	459	0	2	4	876
Chow-Yiu-Hui-Chow	0	2	2	532	0	1	4	782
Libert-Quisquater (basic)	0	2	2	532	0	1	4	782
Libert-Quisquater (short)	0	3	1	454	0	1	2	438
Malone-Lee	0	3	1	454	0	1	3	610
Chen-Malone-Lee	0	3	1	454	0	1	3	610
Sakai-Kasahara†	2	$1+1^{\S}$	0	204	1	0	2	570
BLMQ (ours)	1	2	0	**193**	1	0	2	**349**

Table 8 1 Comparison of Signcryption Schemes

(†) This scheme requires the Weil pairing

(§) One PSM is in $\mathbb{F}_{p^k}$, though this can be made efficient by choosing the trace zero group

Bibliography

[1] 1363, I. 2000: Standard Specifications for Public Key Cryptography, 2000. (page 45.)

[2] Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., and Shi, H. Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. Advances in Cryptology – Crypto '05 (To Appear), 2005. Available online at http://eprint.iacr.org/2005/254. (pages 132 and 133.)

[3] Al-Riyami, S., Malone-Lee, J., and Smart, N. Escrow-Free Encryption Supporting Cryptographic Workflow. Cryptology ePrint Archive, Report 2004/258, 2004. http://eprint.iacr.org/2004/258. (page 115.)

[4] Al-Riyami, S. S., and Paterson, K. G. Certificateless Public Key Cryptography. In *Advances in Cryptology – Asiacrypt 2003* (2003), vol. 2894 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 452–473. (page 115.)

[5] Al-Riyami, S. S., and Paterson, K. G. Tripartite Authenticated Key Agreement Protocols from Pairings. In *IMA Conference on Cryptography and Coding* (2003), vol. 2898 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 332–359. (page 147.)

[6] An, J.-H., Dodis, Y., and Rabin, T. On the Security of Joint Signature and Encryption. In *Advances in Cryptography – Eurocrypt'2002* (2002), vol. 2332 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 83–107. (pages 163 and 167.)

[7] Awasthi, A., and Lal, S. ID-based Ring Signature and Proxy Ring Signature Schemes from Bilinear Pairings. ePrint Archive, Report 2004/184, 2004. Available at http://eprint.iacr.org/2004/184. (page 98.)

[8] Baek, J., and Kim, G. Remarks on the Unknown Key-Share Attacks. In *IEICE Trans. Fundamentals* (2000), vol. E83-A, No. 12. (page 145.)

[9] Baek, J., Safavi-Naini, R., and Susilo, W. Public Key Encryption with Keyword Search Revisited. Cryptology ePrint Archive, Report 2005/191, 2005. http://eprint.iacr.org/2005/191. (pages 115, 137, 138 and 141.)

[10] Baek, J., Steinfeld, R., and Zheng, Y. Formal Proofs for the Security of Signcryption. In *International workshop on Public Key Cryptography (PKC'02)* (2002), vol. 2274 of *Lecture Notes in Computer Science*, Springer, pp. 81–98. (page 163.)

[11] Bao, F., and Deng, R. A Signcryption Scheme with Signature Directly Verifiable by Public Key. In *Practice and Theory of Public Key Cryptography PKC '98* (1998), vol. 1431 of *Lecture Notes in Computer Science*, Springer, pp. 55–59. (page 163.)

[12] Bao, F., Deng, R., and Zhu, H. Variations of Diffie-Hellman Problem. In *International Conference on Information and Communications Security (ICICS)* (2003), vol. 2836 of *Lecture Notes in Computer Science*, Springer-Verlag. (page 82.)

[13] Barreto, P. L. S. M., Libert, B., McCullagh, N., and Quisquater, J.-J. Efficient and Provably-Secure Identity-Based Signatures and Signcryption. In *Advances in Cryptography - Asiacrypt 2005* (2005), Lecture Notes in Computer Science, Springer, p. to appear. (pages 165 and 175.)

[14] Barreto, P. S. L. M., Kim, H. Y., Lynn, B., and Scott, M. Efficient Algorithms for Pairing-Based Cryptosystems. In *Advances in Cryptology – Crypto 2002* (2002), vol. 2442 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 354–368. (pages iv, 48, 63, 64, 67 and 174.)

[15] Barreto, P. S. L. M., Lynn, B., and Scott, M. Constructing Elliptic Curves with Pescribed Embedding Degrees. In *Security in Communication Networks – SCN 2002* (2002), vol. 2576 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 263–273. (pages 35, 48 and 66.)

[16] Barreto, P. S. L. M., Lynn, B., and Scott, M. On the Selection of Pairing-Fiendly Groups. In *Selected Areas in Cryptography – SAC 2003* (2003). to appear. (pages 35, 66 and 174.)

[17] Barreto, P. S. L. M., and Naehrig, M. Pairing-friendly elliptic curves of prime order. Cryptology ePrint Archive, Report 2005/133, 2005. Available online at http://eprint.iacr.org/2005/133. (page 66.)

[18] Bellare, M., Namprempre, C., and Neven, G. Security Proofs for Identity-Based Identification and Signature Schemes. In *Advances in Cryptology – Eurocrypt 2004* (2004), vol. 3027 of *Lecture Notes in Computer Science*, Springer, pp. 268–286. (page 98.)

[19] Bellare, M., and Rogaway, P. Entity Authentication and Key Distribution. In *Advances in Cryptology – Crypto '93* (1994), vol. 773 of *Lecture Notes in Computer Science*, Springer-Verlag. (pages iv, 148, 149, 153, 162 and 165.)

[20] Black, P. Dictionary of Algorithms and Data Structures, 2005. Available online at http://www.nist.gov/dads/. (pages 80 and 81.)

[21] Blake, I., Seroussi, G., and Smart, N., Eds. *Elliptic Curves in Cryptography*, vol. 265 of *London Mathematical Society: Lecture Note Series*. Cambridge University Press, 1999. (page 26.)

[22] Blake, I., Seroussi, G., and Smart, N., Eds. *Advances in Elliptic Curve Cryptography*, vol. 317 of *London Mathematical Society: Lecture Note Series*. Cambridge University Press, 2005. (pages 26, 55, 57, 59 and 60.)

[23] Blake-Wilson, S., Johnson, D., and Menezes, A. Key Agreement Protocols and their Security Analysis. In *6th IMA International Conference on Cryptography and Coding* (1997), vol. 1355 of *Lecture Notes In Computer Science*, Springer-Verlag, pp. 30–45. (pages 148, 149, 150, 152, 159 and 162.)

[24] Blum, L., Blum, M., and Shub, M. A Simple Unpredictable Pseudo-Random Number Generator. In *SIAM Journal on Computing* (1986), vol. 15, pp. 364–383. (page 16.)

[25] Boldyreva, A. Efficient Threshold Signatures, Multisignatures and Blind Signatures based on the Gap-Diffie-Hellman-group Signature Scheme. In *International workshop on Public Key Cryptography - PKC 2003* (2003), vol. 2567 of *Lecture Notes in Computer Science*, Springer-Verlag. (page 98.)

[26] Boneh, D., and Boyen, X. Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles. In *Advances in Cryptology - Eurocrypt 2004* (2004), vol. 3027 of *Lecture Notes in Computer Science*, Berlin: Springer-Verlag, pp. 223–238. Available at http://www.cs.stanford.edu/~xb/eurocrypt04b/. (page 115.)

[27] Boneh, D., and Boyen, X. Secure Identity Based Encryption Without Random Oracles. In *Advances in Cryptology - Crypto 2004* (2004), vol. 3152 of *Lecture Notes in Computer Science*, Berlin: Springer-Verlag, pp. 443–459. Available at http://www.cs.stanford.edu/~xb/crypto04b/. (page 115.)

[28] Boneh, D., and Boyen, X. Short Signatures Without Random Oracles. In *Advances in Cryptology - Eurocrypt 2004* (2004), vol. 3027 of *Lecture Notes in Computer Science*, Berlin: Springer-Verlag, pp. 56–73. Available at http://www.cs.stanford.edu/~xb/eurocrypt04a/. (pages 98, 111 and 178.)

[29] Boneh, D., Boyen, X., and Goh, E.-J. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In *Advances in Cryptology - Eurocrypt 2005* (2005),

vol. 3494 of *Lecture Notes in Computer Science*, Berlin: Springer-Verlag, pp. 440–456. Available at `http://www.cs.stanford.edu/~xb/eurocrypt05a/`. (page 115.)

[30] Boneh, D., Crescenzo, G. D., Ostrovsky, R., and Persiano, G. Public Key Encryption with Keyword Search. In *Advances in Cryptography - Eurocrypt '04* (2004), vol. 3027 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 506–522. (pages 115, 129, 132, 133 and 141.)

[31] Boneh, D., and Franklin, M. Identity-Based Encryption from the Weil Pairing. *SIAM Journal of Computing 32*, 3 (2003), 586–615. (pages iv, 44, 114, 120, 122, 127, 137 and 162.)

[32] Boneh, D., Gentry, C., Lynn, B., and Shacham, H. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *Advances in Cryptology – Eurocrypt 2003* (2003), vol. 2656 of *Lecture Notes in Computer Science*, Springer, pp. 416 – 432. (page 98.)

[33] Boneh, D., Lynn, B., and Shacham, H. Short Signatures from the Weil pairing. In *Advances in Cryptology – Asiacrypt'2001* (2002), vol. 2248 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 514–532. (pages 98, 102 and 103.)

[34] Boneh, D., and Waters, B. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys, 2005. (page 115.)

[35] Boyen, X. Multipurpose Identity-Based Signcryption: A Swiss Army knife for Identity-Based Cryptography. In *Advances in Cryptology – Crypto 2003* (2003), vol. 2729 of *Lecture Notes in Computer Science*, Berlin: Springer-Verlag, pp. 383–399. Available at `http://www.cs.stanford.edu/~xb/crypto03/`. (pages 164, 165, 166, 167, 175, 176 and 177.)

[36] Canetti, R., Goldreich, O., and Halevi, S. The Random Oracle Methodology, Revisited. In *Symposium on Theory of Computing – STOC 98* (1998), ACM. (page 98.)

[37] CANETTI, R., HALEVI, S., AND KATZ, J. A forward-secure public-key encryption scheme. In *Advances in Cryptology - Eurocrypt 2003* (2003), vol. 2656 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 255–271. (page 115.)

[38] CARMICHAEL, R. D. On Composite Numbers which Satisfy the Fermat Congruence. In *American Mathematical Society Monthly* (1912), vol. 19, AMS, pp. 22–27. (page 19.)

[39] CASTEJON-AMENEDO, J., AND MCCUE, R. Extracting Randomness from External Interrupts. In *International Conference on Communication, Network, and Information security* (2003), ACTA Press. Available online at http://www.cs.iastate.edu/~simov/erfei.pdf. (page 15.)

[40] CERTICOM CORP. ECC FAQ. Available online at http://www.certicom.com/index.php?action=ecc,ecc_faq. (page 90.)

[41] CHA, J. C., AND CHEON, J. H. An Identity-Based Signature from Gap Diffie-Hellman Groups. In *Practice and Theory in Public Key Cryptography - PKC 2003* (Miami, USA, 2003), vol. 2567 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 18–30. (pages 98, 107, 108 and 118.)

[42] CHAN, A., AND BLAKE, I. Conditionally Verifiable Signatures. Cryptology ePrint Archive, Report 2005/149, 2005. http://eprint.iacr.org/2005/149. (page 98.)

[43] CHEN, L., AND CHENG, Z. Security Proof of Sakai-Kasahara's Identity-Based Encryption Scheme. Cryptology ePrint Archive, Report 2005/226, 2005. http://eprint.iacr.org/2005/226. (page 127.)

[44] CHEN, L., AND KUDLA, C. Identity Based Authenticated Key Agreement from Pairings. Cryptology ePrint Archive, Report 2002/184, 2002. http://eprint.iacr.org/2002/184. (pages 145, 147, 148, 154, 155 and 156.)

[45] Chen, L., and Malone-Lee, J. Improved Identity-Based Signcryption. In *PKC'2005* (2003), vol. 3386 of *Lecture Notes in Computer Science*, Springer, pp. 362–379. (pages 165, 175 and 176.)

[46] Chen, L., and Malone-Lee, J. Improved Identity-Based Signcryption. Cryptology ePrint Archive, Report 2004/114, 2004. http://eprint.iacr.org/2003/114. (pages 118 and 168.)

[47] Chen, Z. Security Analysis on Nalla-Reddy's ID-Based Tripartite Authenticated Key Agreement Protocols. Cryptology ePrint Archive, Report 2003/103, 2003. http://eprint.iacr.org/2003/103. (page 147.)

[48] Cheng, Z., and Chen, L. On Security Proof of McCullagh-Barreto's Key Agreement Protocol and its Variants. Cryptology ePrint Archive, Report 2005/201, 2005. http://eprint.iacr.org/2005/201. (pages 156 and 220.)

[49] Cheng, Z., and Comley, R. Efficient Certificateless Public Key Encryption. Cryptology ePrint Archive, Report 2005/012, 2005. http://eprint.iacr.org/2005/012. (page 115.)

[50] Cheon, J. H., Kim, Y., and Yoon, H. A New ID-based Signature with Batch Verification. Cryptology ePrint Archive, Report 2004/131, 2004. http://eprint.iacr.org/2004/131. (page 98.)

[51] Chevallier-Mames, B., Coron, J.-S., McCullagh, N., Naccache, D., and Scott, M. Secure Delegation of Elliptic-Curve Pairing. Cryptology ePrint Archive, Report 2005/150, 2005. http://eprint.iacr.org/2005/150. (page 74.)

[52] Chevallier-Mames, B., Naccache, D., Paillier, P., and Pointcheval, D. How to Disembed a Program? In *Cryptographic Hardware and Embedded Systems – CHES 2004* (2004), vol. 3156 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 441–454. (page 72.)

[53] CHOW, S., HUI, L., AND YIU, S. Identity Based Threshold Ring Signature. In *Information Security and Cryptology - ICISC 2004* (2004), vol. 3506 of *Lecture Notes in Computer Science*, Springer, pp. 218 – 232. (page 98.)

[54] CHOW, S. S. M., YIU, S. M., HUI, L. C. K., AND CHOW, K. P. Efficient Forward and Provably Secure ID-Based Signcryption Scheme with Public Verifiability and Public Ciphertext Authenticity. In *6th International Conference on Information Security and Cryptology – ICISC'2003* (2004), vol. 2971 of *LNCS*, Springer, pp. 352–369. (page 165.)

[55] CHOW, S. S. M., YUEN, T. H., HUI, L. C. K., AND YIU, S. M. Signcryption in Hierarchical Identity Based Cryptosystem. In *20th International Conference on Information Security (SEC'2005)* (2005), IFIP TC11. (pages 115 and 165.)

[56] CHUNXIANG, X., JUNHUI, Z., AND ZHIGUANG, Q. A Note on Secure Key Issuing in ID-Based Cryptography. Cryptology ePrint Archive, Report 2005/180, 2005. http://eprint.iacr.org/2005/180. (page 124.)

[57] DIEM, C. Index Calculus in Class Groups of Plane Curves of Small Degree. Cryptology ePrint Archive, Report 2005/119, 2005. http://eprint.iacr.org/2005/119. (page 89.)

[58] DIFFIE, W., AND HELLMAN, M. New Directions in Cryptography. In *IEEE Transactions on Information Theory* (1976), vol. 22, pp. 644–654. (pages 142 and 150.)

[59] DIRECTIVE, E. U. Electronic Signature Directive, 1999. Available online at http://www.e-podpis.sk/laws/eu_ep_dir93_1999.pdf and elsewhere. (page 99.)

[60] EASTLAKE, D., CROCKER, S., AND SCHILLER, J. RFC 1750: Randomness Recommendations for Security, 1994. Available online at http://www.ietf.org/rfc/rfc1750.txt. (page 14.)

[61] ElGamal, T. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in Cryptology – Crypto '84* (1984), Springer-Verlag, p. 10Ű18. A version is available online at http://crypto.csail.mit.edu/classes/6.857/papers/elgamal.pdf. (pages 23 and 96.)

[62] Enge, A. *Elliptic Curves and their Applications to Cryptography.* Kluwer Academic Publishers, 1999. (page 26.)

[63] Floyd, R. W. Non-deterministic Algorithms. In *Journal of the ACM* (1967), vol. 14-4, Association for Computing Machinery, pp. 636 – 644. (page 88.)

[64] for Standards in Technology (NIST), N. I. Federal Information Processing Standards Publication 186. The Digital Signature Standard. Available online at http://www.itl.nist.gov/fipspubs/fip186.htm. (page 97.)

[65] free encyclopedia, W. T. Big-O Notation Definition, 2005. (pages 80 and 81.)

[66] Fujisaki, E., and Okamoto, T. Secure Integration of Asymmetirc and Symmetric Encryption Schemes. In *Advances in Cryptology – Crypto '99* (1999), vol. 1666 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 537–554. (page 126.)

[67] Galbraith, S., Harrison, K., and Soldera, D. Implementing the Tate pairing. In *Algorithm Number Theory Symposium – ANTS V* (2002), vol. 2369 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 324–337. (pages iv, 67, 174 and 209.)

[68] Galbraith, S., and Rotger, V. Easy Decision-Diffie-Hellman Groups. Cryptology ePrint Archive, Report 2004/070, 2004. http://eprint.iacr.org/2004/070. (page 174.)

[69] Gaudry, P. Index Calculus for Abelian Varieties and the Elliptic Curve Discrete Logarithm Problem. Cryptology ePrint Archive, Report 2004/073, 2004. http://eprint.iacr.org/2004/073. (page 89.)

[70] Gentry, C., and Silverberg, A. Hierarchical ID-Based Cryptography. In *Advances in Cryptology - Asiacrypt 2002* (2002), vol. 2501 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 548–566. Available online at http://eprint.iacr.org/2002/056. (page 115.)

[71] Han, Y., and Yang, X. Elliptic Curve based Signcryption and its Multi-party Schemes. Cryptology ePrint Archive, Report 2004/142, 2004. http://eprint.iacr.org/2004/142. (page 184.)

[72] Hankerson, D., Menezes, A., and Vanstone, S. *Guide to Elliptic Curve Cryptography.* Springer, 2004. (pages 26, 30, 34, 40, 42 and 45.)

[73] Herranz, J., and Saez, G. New Identity-Based Ring Signature Schemes. In *Information and Communications Security, ICICS 2004* (2004), vol. 3269 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 27–39. (page 98.)

[74] Hess, F. Efficient Identity Based Signature Schemes Based on Pairings. In *Selected Areas in Cryptography - SAC'2002* (2003), vol. 2595 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 310–324. (page 98.)

[75] Hoffstein, J., Pipher, J., and Silverman, J. H. NTRU: A Ring-Based Public Key Cryptosystem. In *Algorithmic Number Theory (ANTS III)* (1998), vol. 1423 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 267–288. (page 82.)

[76] Horwitz, J., and Lynn, B. Towards Hierarchical Identity-Based Encryption. In *Advances in Cryptology - Eurocrypt 2002* (2002), vol. 2332 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 466–481. (page 115.)

[77] Joux, A. A One round Protocol for Tripartite Diffe-Hellman. In *Algorithmic Number Theory Symposium (ANTS-00)* (2000), vol. 1838 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 385–395. (pages 114, 146 and 147.)

[78] JUN, B., AND KOCHER, P. The Intel Random Number Generator, 1999. Cryptography Research, Inc. White Paper prepared for Intel Corporation. (page 15.)

[79] LABORATORIES, R. Crypto faq. RSA website. Available online at http://www.rsasecurity.com/rsalabs/node.asp?id=2184. (page 142.)

[80] LAGUILLAUMIE, F., AND VERGNAUD, D. Multi-Designated Verifiers Signatures. In *Information and Communications Security, ICICS 2004* (2004), vol. 3269 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 495 – 507. (page 98.)

[81] LIBERT, B., AND QUISQUATER, J.-J. New Identity Based Signcryption Schemes based on Pairings. In *IEEE Information Theory Workshop* (Paris, France, 2003). (pages 118, 126, 165 and 172.)

[82] LIBERT, B., AND QUISQUATER, J.-J. The Exact Security of an Identity Based Signature and its Applications. Cryptology ePrint Archive, Report 2004/102, 2004. Available online at http://eprint.iacr.org/2004/102. (pages 105 and 106.)

[83] LIDL, R., AND NIEDERREITER(EDS.)., H. *Finite Fields.* Cambridge University Press, 1996. Available to buy online at http://www.amazon.co.uk. (page 5.)

[84] LIN, C.-Y., AND TZONG-CHENWU. An Identity-Based Ring Signature Scheme from Bilinear Pairings. Cryptology ePrint Archive, Report 2003/117, 2003. Available online at http://eprint.iacr.org/2003/117. (page 98.)

[85] LYNN, B. Authenticated Identity-Based Encryption. Available online at http://eprint.iacr.org/2002/072. (page 163.)

[86] MALONE-LEE, J. Identity-Based Signcryption. Cryptology ePrint Archive, Report 2002/098, 2002. http://eprint.iacr.org/2002/098. (pages 118, 126, 165 and 168.)

[87] MAO, W., AND HARRISON, K. Divisors, Bilinear Pairings and Pairing Enabled Cryptographic Applications. *Hewlett Packard Technical Slide Presentation* (2003). (pages 49 and 50.)

[88] McCullagh, N., and Barreto, P. Efficient and Forward-secure Identity-Based Signcryption. Cryptology ePrint Archive, Report 2004/117, 2004. http://eprint.iacr.org/2004/117. (page 126.)

[89] McCullagh, N., and Barreto, P. L. S. M. A New Two-Party Identity-Based Authenticated Key Agreement. In *Cryptographers Track - RSA Conference* (2005), vol. 3376 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 262–274. (pages 148 and 154.)

[90] Menezes, A. *Elliptic Curve Public Key Cryptosystems.* Kluwer Academic Publishers, 1993. (pages 26, 30 and 51.)

[91] Menezes, A., van Oorschot, P., and Vanstone, S. *Handbook of Applied Cryptography.* CRC Press, October 1996, (Fifth Edition August 2001). (pages 2, 5, 8, 16, 18, 20, 21, 82 and 142.)

[92] Menezes, A. J., Okamoto, P., and Vanstone, S. A. Reducing Elliptic Curve Logarithms to a Finite Field. In *Information Theory* (1993), vol. 39, IEEE Transactions, pp. 1639–1646. (pages 47, 66, 92 and 114.)

[93] Michon. Modular Arithmetic, Fermat Theorem, Carmichael Numbers - Numericana, 2000–2005. Available online at http://home.att.net/~numericana/answer/modular.htm. (page 19.)

[94] Miller, V. Use of Elliptic Curves in Cryptography. In *Advances in Cryptography – Crypto '85* (1986), H. Williams, Ed., Lecture Notes in Computer Science, Springer-Verlag, pp. 417 – 426. (page 47.)

[95] Miyaji, A., Nakabayashi, M., and Takano, S. New Explicit Conditions of Elliptic Curve Traces for FR-Reduction. *IEICE Transactions on Fundamentals E84 A, no. 5* (2001). (pages 35 and 66.)

[96] Naccache, D. Personal communication, 2005. (page 79.)

[97] NALLA, D. ID-Based Tripartite Key Agreement with Signatures. Cryptology ePrint Archive, Report 2003/144, 2003. http://eprint.iacr.org/2003/144. (page 147.)

[98] NALLA, D., AND REDDY, K. C. Signcryption Scheme for Identity-Based Cryptosystems. Cryptology ePrint Archive, Report 2003/066, 2002. http://eprint.iacr.org/2003/066. (page 165.)

[99] NALLA, D., AND REDDY, K. C. ID-Based Tripartite Authenticated Key Agreement Protocols from Pairings. Cryptology ePrint Archive, Report 2003/004, 2003. http://eprint.iacr.org/2003/004. (page 147.)

[100] NYBERG, K., AND RUEPPEL, R. A New Signature Scheme based on the DSA giving message recovery. In *The 1st ACM conference on Computer and communications security* (1993), ACM Press, pp. 58 – 61. (page 97.)

[101] PARLIAMENT), T. O. T. I. Electronic Commerce Bill (2000), ireland, 2000. Available online at http://www.gov.ie/bills28/bills/2000/1300/default.htm. (page 98.)

[102] PATERSON, K. G. ID-Based Signatures from Pairings on Elliptic Curves. *Electronics Letters 38(18)* (2002), 1025–1026. (page 98.)

[103] POINTCHEVAL, D., AND STERN, J. Security Proofs for Signature Schemes. In *Eurocrypt'96* (1996), vol. 1992 of *LNCS*, Springer, pp. 387–398. (pages 108 and 182.)

[104] POINTCHEVAL, D., AND STERN, J. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology 13*, 3 (2000), 361–396. (pages 108 and 182.)

[105] POLLARD, J. M. Monte Carlo Methods for Index Computation (mod p). In *Mathematics of Computation* (1978), vol. 32(143), American Mathematical Society, pp. 918–924. (pages 88 and 89.)

[106] RABIN, M. Probabilistic Algorithm for Testing Primality. In *Journal of Number Theory* (1980), vol. 12, Science Direct, pp. 128–138. Available online at http://www.sciencedirect.com. (page 19.)

[107] Rivest, R., Shamir, A., and Adleman, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, 1978. Previously released as an MIT 'Technical Memo' (1977), and available online at http://theory.lcs.mit.edu/~rivest/rsapaper.pdf. (pages 11, 96 and 97.)

[108] Rosing, M. *Implmenting Elliptic Curve Cryptography.* Manning, 1998. (page 26.)

[109] Sakai, R., and Kasahara, M. ID based Cryptosystems with Pairing on Elliptic Curve. In *2003 Symposium on Cryptography and Information Security - SCIS 2003* (Hamamatsu, Japan, 2003). http://eprint.iacr.org/2003/054. (pages 98, 154, 162, 165 and 171.)

[110] Sakai, R., and Kasahara, M. ID Based Cryptosystems with Pairing on Elliptic Curve. In *2003 Symposium on Cryptography and Information Security - SCIS'2003* (Hamamatsu, Japan, 2003). See also http://eprint.iacr.org/2003/054. (page 126.)

[111] Sakai, R., Ohgishi, K., and Kasahara, M. Cryptosystems based on Pairings. In *Symposium on Cryptography and Information Security (SCIS), Okinawa, Japan* (2000), vol. 17. (pages iv, 98, 104, 115, 118, 147, 148, 151 and 161.)

[112] Schnorr, C. Efficient Signature Generation by Smart Cards. *Journal of Cryptology 4* (1991), 161–174. (page 97.)

[113] Schnorr, C., and Jakobsson, M. Security of Signed ElGamal Encryption. In *Asiacrypt '00* (2000), vol. 1976 of *Lecture Notes in Computer Science*, pp. 73–89. Available online at http://www.mi.informatik.uni-frankfurt.de/research/papers/schnorr.signed_elgamal.2000.pdf. (page 163.)

[114] Scott, M. Authenticated ID-Based Key Exchange and Remote log-in with Insecure token and PIN number. Cryptology ePrint Archive, Report 2002/164, 2002. http://eprint.iacr.org/2002/164/. (pages 118, 147 and 148.)

[115] SCOTT, M. Personnal communication with noel mccullagh, 2002. (page 122.)

[116] SCOTT, M. Computing the Tate Pairing. In *Cryptographers Track - RSA Conference* (2005), vol. 3376 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 293–304. (pages vii, 66, 68, 119 and 121.)

[117] SCOTT, M. The Tate Pairing, 2005. Available online at `http://www.computing.dcu.ie/~mike/tate.html`. (page 48.)

[118] SHAMIR, A. Identity Based Cryptosystems and Signature Schemes. In *Advances in Cryptology – Crypto'84* (1984), vol. 0196 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 47–53. (pages iv and 115.)

[119] SHANKS, D. Class Number, A Theory of Factorization and Genera. In *Symposium of Pure Mathematics* (1971), vol. 20, American Mathematical Society, pp. 415–440. (page 87.)

[120] SHIM, K. Cryptanalysis of Al-Riyami-Paterson's Authenticated Three Party Key Agreement Protocols. Cryptology ePrint Archive, Report 2003/122, 2003. `http://eprint.iacr.org/2003/122`. (page 147.)

[121] SHIM, K. Cryptanalysis of ID-Based Tripartite Authenticated Key Agreement Protocols. Cryptology ePrint Archive, Report 2003/115, 2003. `http://eprint.iacr.org/2003/115`. (page 147.)

[122] SHIM, K. Efficient ID-Based Authenticated Key Agreement Protocol based on Weil Pairing. *Electronics Letters 39*, 8 (2003), 653–654. (page 147.)

[123] SHIM, K. Efficient One Round Tripartite Authenticated Key Agreement Protocol from Weil Pairing, 2003. (page 147.)

[124] SHIN, J., LEE, K., AND SHIM, K. New DSA-Verifiable Signcryption Schemes. In *Information Security and Cryptology - ICISC 2002* (2002), vol. 2587 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 35–47. (page 170.)

[125] Smart, N. *Cryptography: An Introduction.* McGraw Hill, 2003. (pages 2, 5, 7, 10, 18, 26, 30 and 34.)

[126] Smart, N., Stern, J., and Naccache, D. Projective Coordinates Leak. In *Advances in Cryptology - EuroCrypt 2004* (April 2004), Springer Verlag LNCS 3027, pp. 257–267. (page 45.)

[127] Smart, N. P. An Identity Based Authenticated Key Agreement Protocol Based on the Weil Pairing. *Electronics Letters 38* (2002), 630–632. (pages 118, 147, 152 and 161.)

[128] Solinas, J. Generalized Mersenne numbers, 1999. Available online at http://www.cacr.math.uwaterloo.ca/. (page 64.)

[129] Solinas, J. ID-based Digital Signature Algorithms. Slide Show, 2003. Available online at http://www.cacr.math.uwaterloo.ca/conferences/2003/ecc2003/solinas.pdf. (page 68.)

[130] Sui, A., Chow, S., Hui, L., Yiu, S., Chow, K., Tsang, W., Chong, C., Pun, K., and Chan, H. Separable and Anonymous Identity-Based Key Issuing. In *1st International Workshop on Security in Networks and Distributed Systems (SNDS 2005), in conjunction with 11th International Conference on Parallel and Distributed Systems (ICPADS 2005)* (2005), IEEE Computer Society. (page 124.)

[131] Sun, H.-M., and Hsieh, B.-T. Security Analysis of Shim's Authenticated Key Agreement Protocols from Pairings. Cryptology ePrint Archive, Report 2003/113, 2003. http://eprint.iacr.org/2003/113. (page 147.)

[132] Team, T. M. K. Kerberos: The Network Authentication Protocol, 2003 – 2005 (available outside U.S. and Canada). Documentation available online at http://web.mit.edu/kerberos/www/krb5-1.4/. (pages 22 and 142.)

[133] Vanstone, S., Mullin, R., and Agnew, G. Elliptic Curve Encryption Systems, 2000. (page 44.)

[134] Verheul, E. R. Evidence that XTR is more secure than Supersingular Elliptic Curve Cryptosystems. In *Advances in Cryptology - Eurocrypt 2001* (2001), vol. 2045 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 195–210. (page 60.)

[135] Wang, X., Yin, Y., and Yu, H. Collision Search Attacks on SHA-1. Available online at http://theory.csail.mit.edu/~yiqun/shanote.pdf. (page 98.)

[136] Wang, X., and Yu, H. How to break MD-5 and other hash functions. Available online at http://www.infosec.sdu.edu.cn/paper/md5-attack.pdf. (page 98.)

[137] Washington, L. *Elliptic Curves: Number Theory and Cryptography.* Chapman and Hall/CRC, 2003. (pages 26, 33, 49, 50, 54, 55, 57 and 92.)

[138] Xie, G. Cryptanalysis of Noel McCullagh and Paulo S. L. M. Barreto's two-party identity-based key agreement. Cryptology ePrint Archive, Report 2004/308, 2004. http://eprint.iacr.org/2004/308. (page 158.)

[139] Xu, J., Zhang, Z., and Feng, D. A Ring Signature Scheme Using Bilinear Pairings. In *Information Security Applications, 5th International Workshop, WISA 2004* (2000), vol. 3325 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 163–172. (page 98.)

[140] Yacobi, Y. A note on the Bilinear Diffie Hellman Assumption. Cryptology ePrint Archive, Report 2002/113, 2002. http://eprint.iacr.org/2002/113. (page 82.)

[141] Yi, X. An Identity-Based Signature Scheme from the Weil Pairing. *IEEE Communications Letters 7(2)* (2003), 76–78. (page 98.)

[142] Yuen, T. H., and Wei, V. K. Fast and Proven Secure Blind Identity-Based Signcryption from Pairings. In *CT-RSA'2005* (2003), vol. 3376 of *Lecture Notes in Computer Science*, Springer, pp. 305–322. (page 165.)

[143] Zhang, F., and Chen, X. Yet Another Short Signature without Random Oracles from Bilinear Pairings. Cryptology ePrint Archive, Report 2005/230, 2005. Available online at http://eprint.iacr.org/2005/230. (page 98.)

[144] Zhang, F., and Kim, K. ID-based Blind Signature and Ring Signature from Pairings. In *Advances in Cryptology - Asiacrypt 2002* (2002), vol. 2501 of *Lecture Notes in Computer Science*, Springer, pp. 533–547. (page 98.)

[145] Zhang, F., Safavi-Naini, R., and Susilo, W. An Efficient Signature Scheme from Bilinear Pairings and Its Applications. In *International Workshop on Practice and Theory in Public Key Cryptography – PKC'2004* (2004), Lecture Notes in Computer Science, Springer-Verlag. to appear. (page 98.)

[146] Zheng, Y. Digital Signcryption or how to achieve Cost (Signature & Encryption) << Cost(signature) + Cost(encryption). In *Advances in Cryptology – Crypto'97* (1997), vol. 1294 of *Lecture Notes in Computer Science*, Springer, pp. 165–179. (page 163.)

[147] Zheng, Y., and Imai, H. Efficient Signcryption Schemes on Elliptic Curves. *Information Processing Letters 68 (5)* (1998), 227 – 233. Available online at http://citeseer.ist.psu.edu/129994.html. (page 163.)

Appendix A

Java Random Numbers

The following code uses the sound card to generate random numbers It fills a large byte array full of CD quality sound and then picks the least significant bit of each 16 bit frame Given a parameter k it will generate a random number x in the interval $0 \leq x < 2^k$

```
/*
Java function to generate a random number
from the input of a soundcard
*/
        public BigInteger getRandBits(int lengthOfRandom) {

        TargetDataLine line;
        Thread thread;
        duration = 0;
        audioInputStream = null;

            // line-in, is the micropohone, we are recording CD quality, mono signal

        AudioFormat format = new AudioFormat(AudioFormat.Encoding.PCM_SIGNED, 44100, 16,
                             1, 2, 44100, true);

            DataLine.Info info = new DataLine.Info(TargetDataLine.class,
                format);

            if (!AudioSystem.isLineSupported(info)) {
                return new BigInteger("-1");
            }

            // get and open the target data line for capture.

            try {
                line = (TargetDataLine) AudioSystem.getLine(info);
                line.open(format, line.getBufferSize());
            } catch (LineUnavailableException ex) {
                return new BigInteger("-1");
            } catch (SecurityException ex) {
                return new BigInteger("-1");
            } catch (Exception ex) {
                return new BigInteger("-1");
            }

            // play back the captured audio data
            // ByteArrayOutputStream out = new ByteArrayOutputStream();
            int frameSizeInBytes = format.getFrameSize();
            int bufferLengthInFrames = line.getBufferSize() / 8;
            int bufferLengthInBytes = bufferLengthInFrames * frameSizeInBytes;
            byte[] data = new byte[bufferLengthInBytes];
            int numBytesRead;
            int exponent = 0;
                        BigInteger total = new BigInteger("0");
                        int bufin = -1;
                        int cycles = 0;
                        BigInteger BTwo = new BigInteger("2");

            line.start();

                        //System.out.println(bufferLengthInBytes);
                if((numBytesRead = line.read(data, 0, bufferLengthInBytes)) == -1) {
                                System.exit(0);
                }
                                byte[] nba = new byte[1];
                                // want to construct random number here
                                while(cycles < lengthOfRandom)
                                {
                                cycles = cycles + 1; //k cycles for a 2^k number
                                bufin = bufin + 2; //16bit frame so advance two blocks
                                byte nextbit = (byte) (1 & data[bufin]);//take last bit of byte
                                nba[0] = nextbit; //convert to byte array
                                total = (total.multiply(BTwo)).add(new BigInteger(nba));//d and a
                                }
                line.drain();
                line.stop();
                line.close();
                line = null;

        return total;
    }
```

Appendix B

Java Library for $k = 2$ Elliptic Curves

The following code adds two points in an elliptic curve The code is slightly more complicated than the equations given in Ch 2 as this code implements point addition and point doubling, and some slight complications involving the point at infinity $\mathcal{O}$

```
public Point add(Point exPoint) {
        /*
         *      Just check if either of the points is the point at infinity
         *      or if one is the negative of the other
         */

    if(this.isInfinity()) {
        return exPoint;
    }
    if(exPoint.isInfinity()) {
        return this;
    }
    if((this.negate()).equals(exPoint)) {
        Point retP = new Point(EC); // this point is set to infinity by default
        return retP;
    }

    BigInteger x2 = exPoint.getX();
    BigInteger y2 = exPoint.getY();
    BigInteger delta = new BigInteger("0");
    BigInteger deltan = new BigInteger("0");
    BigInteger deltad = new BigInteger("0");

    if(!x.equals(x2)) {
        deltan = (y2.subtract(y)).mod(EC.getModulus());
        deltad = (x2.subtract(x)).modInverse(EC.getModulus());
        delta = (deltan.multiply(deltad)).mod(EC.getModulus());

    }
    else if((x.equals(x2)) && (!y.equals(new BigInteger("0")))) {
        BigInteger two = new BigInteger("2");
        BigInteger three = new BigInteger("3");
        deltan = ((three.multiply(x.modPow(new BigInteger("2"),
            EC.getModulus()))).add(EC.getA())).mod(EC.getModulus());
        deltad = (two.multiply(y)).modInverse(EC.getModulus());
        delta = (deltan.multiply(deltad)).mod(EC.getModulus());
    }
    BigInteger x3 = ((delta.modPow(new BigInteger("2"),
            EC.getModulus())).subtract(x).subtract(x2)).mod(EC.getModulus());
    BigInteger y3 = (((x.subtract(x3)).multiply(delta)).subtract(y)).mod(EC.getModulus());
    Point retP = new Point(EC, x3, y3);

    return retP;
};
```

The following code is the simplest and slowest implementation of elliptic curve point scalar multiplication. It is the basic "double and add" algorithm and is included here for its simplicity. A more complicated windowing method is implemented on the accompanying CD.

```
public Point multiply(BigInteger exS) {
    BigInteger S = exS;
    Point tp = new Point(this.getEC(), x, y);
    Point tprt = new Point(this.getEC());//this is the point at infinity (running total)

    while(S.bitLength() > 0) {
        if(S.testBit(0)) {
            tprt = tprt.add(tp);     //add
        }
        tp = tp.add(tp);   //double

        S = S.shiftRight(1);   //divide s by 2
    }

    return tprt;
}
```

A Java implementation of the "Map To Point" function in Boneh and Franklin's IBE scheme. The function selectively implements the faster "Map To Point" function of McCullagh if the boolean input is set to `false`.

```
public Point(Curve exEC, String exID, String hash, boolean OrderQ) throws Exception {
    int hlen;

    if(hash.equals("SHA-256"))
    {
     hlen = 32;
    }
    else if(hash.equals("SHA-1"))
    {
       hlen = 20;
    }
    else
    {
       throw new Exception(hash + ":ALGORITM NOT SUPPORTED
            IN IDENTITY TO POINT MAPPING\nTRY \"SHA-1\" OR \"SHA-256\"");
    }

    MessageDigest md = MessageDigest.getInstance(hash);
    md.update(exID.getBytes());
    byte[] s = md.digest();

            BigInteger p=exEC.getModulus();


            BigInteger h= BigInteger.ONE;
    int i, j;

        j=0; i=1;
    while(true) {
        h = h.multiply(new BigInteger("256"));
        if (j==hlen) {
            h = h.add(new BigInteger(Integer.toString(i++)));
            j=0;
        }
        else {
            h = h.add(new BigInteger(Integer.toString(s[j++])));
        }
        if (p.compareTo(h) == -1)
            break;
    }
    h= h.mod(p);

            //System.out.println("Hash value is :" + h.toString(16));
/*
 * Now we want to form a point and use this as the X co-ord
 * P is congruent to 3 mod 4, this makes finding sqrt easy
 */

    EC = exEC;
    BigInteger ty = genY(h, EC);
    x = getTx();
    y = ty;

    //this.clone(exEC, getTx(), ty);

    if(OrderQ == true)  //does the point have to be of order q, if yes do this, if not don't
    {
    this.multiply(exEC.getCoF());
    }
    OnCurve = true;

}
```

And the function genY, which is used by "Map To Point" to find a point on the curve given only the X co-ordinate.

```
/*
        Function to find the Y co-ordinate of a point,
        given the X co-ordinate
*/

    private BigInteger genY(BigInteger exh, Curve exEC) {

            tx = exh;
            BigInteger pmod = exEC.getModulus();
            BigInteger delta = exh.modPow(new BigInteger("3"), pmod);
            delta = delta.add(exEC.getA().multiply(exh)).mod(pmod);
            delta = delta.add(exEC.getB()).mod(pmod);
            BigInteger exp = (pmod.add(BigInteger.ONE)).divide(new BigInteger("4"));
            BigInteger sqrt = delta.modPow(exp,pmod);
            BigInteger norm = sqrt.modPow(new BigInteger("2"),pmod);

            if (delta.compareTo(norm) != 0)
            {
              return genY(tx.add(BigInteger.ONE), exEC);
            }

            return sqrt;
    }
```

A Java implementation of $t(P, Q)$, the reduced Tate pairing, using Miller's algorithm. This code is relatively optimisation free to make it more readable and easy to relate to the mathematics of chapter 3. This code shows clearly the relationship between Miller's algorithm and the "Double and Add" algorithm for elliptic curve point scalar multiplication. This code takes both points from the extension field, so will be slow. It minimises polynomial division by computing the miller function as a numerator `num` and denominator `denum` as suggested by Galbraith *et al.* [67]. A more optimised version of the Tate pairing is including on the accompanying CD.

```
public ZZn2 e(ECn2 P, ECn2 Q)
{
    ECn2 LP = P.copy();
    ECn2 LQ = Q.copy();

    ZZn2 Qx = LQ.getX();
    ZZn2 Qy = LQ.getY();

    num = new ZZn2(this.P);              //these will both be set to one
    denom = new ZZn2(this.P);

    ECn2 pA = P.copy();                   //A = P
    GA = pA;

    int nb = q.bitLength();

    for(int i = nb-2;i>=0;i--)
    {
        num=num.multiply(num);
        denom = denom.multiply(denom);

        g(pA,pA,Qx,Qy);
        pA = GA;                //this will have changed because of g(.)

        if(q.testBit(i))
        {
          g(pA,P,Qx,Qy);
          pA = GA;
        }
    }

    ZZn2 res = num.divide(denom);

    if((!pA.isZero()) || (res.isZero()))
    {
        return new ZZn2(this.P);
    }

    BigInteger e = (this.P.add(BigInteger.ONE).divide(this.q));

    ZZn2 resc = res.conj();
    res = resc.divide(res);
    res = res.pow(e);

    return res;
}
```

The function g which is used in the computation of Miller's algorithm. This function must work out the gradient of a line.

```
public void g(ECn2 pA, ECn2 B, ZZn2 Qx, ZZn2 Qy) {
    ZZn2 lam = new ZZn2(P);
    ZZn2 d,u,y;
    u = pA.getX();
    y = pA.getY();

    pA = pA.add(B);
    lam = pA.getlam();
    if(lam.isZero())
    {
        return;
    }
    if(pA.isZero())
    {
        u = u.subtract(Qx);
        d = new ZZn2(P);      //this will be set to one
    }
    else
    {
        u = u.subtract(Qx);
        u = u.multiply(lam);
        y = y.subtract(Qy);
        u = u.subtract(y);
        d = pA.getX();
        d = d.subtract(Qx);
    }

    num = num.multiply(u);
    denom = denom.multiply(d);
    GA = pA;

}
```

The following code is used to multiply two field elements $\in \mathbb{F}_{q^2}$. This is the basis of pairing exponentiation. It can be used with the standard "square and multiply" algorithm for exponentiation or more complex sliding window methods.

```
public ZZn2 multiply(ZZn2 exPoint) {
    if((exPoint.getA().equals(a)) && (exPoint.getB().equals(b))) {
    /*  same point
        a = (a+b)(a-b)
        b = 2ab
     */

        BigInteger sa,ta,tb,tf,ts;

        tf = (a.add(b)).mod(p);
        ts = (a.subtract(b)).mod(p);
        ta = (tf.multiply(ts)).mod(p);
        sa = (a.add(a)).mod(p);
        tb = (b.multiply(sa)).mod(p);

        return new ZZn2(p, ta, tb);
    }
    else {
        BigInteger t,t2,t3,tb;

        t = (a.multiply(exPoint.getA())).mod(p);
        t2 = (b.multiply(exPoint.getB())).mod(p);
        t3 = exPoint.getA().add(exPoint.getB()).mod(p);
        tb = b.add(a).mod(p);
        tb = tb.multiply(t3).mod(p);
        tb = tb.subtract(t).mod(p);
        tb = tb.subtract(t2).mod(p);
        t = t.subtract(t2).mod(p);

        return new ZZn2(p, t,tb);
    }
}
```

B 1 Proof of Theorem 6 7.1

Proof Algorithm $\mathcal{B}$ takes as input $\langle P, Q, \alpha Q, \alpha^2 Q, \quad , \alpha^p Q\rangle$ and attempts to extract $e(P,Q)^{1/\alpha}$ from its interaction with $\mathcal{A}$

In a preparation phase, $\mathcal{B}$ selects an index $\ell \xleftarrow{R} \{1, \quad , q_{\mathcal{H}_W}\}$, elements $I_\ell \xleftarrow{R} \mathbb{Z}_q^*$ and $w_1, \quad w_{\ell-1}, w_{\ell+1} \quad w_{q_{\mathcal{H}_W}} \xleftarrow{R} \mathbb{Z}_q^*$ For $i = 1, \quad , \ell - 1 \ \ell + 1 \quad , q_{\mathcal{H}_W}$, it computes $I_i = I_\ell - w_i$ As in the technique of Boneh-Boyen, it sets up generators $G_2 \in \mathcal{G}_2$, $G_1 = \psi(G_2) \in \mathcal{G}_1$ and another $\mathcal{G}_2$ element $U = \alpha G_2$ such that it knows $q_{\mathcal{H}_W} - 1$ pairs $(w_i, H_i = (1/(w_i + \alpha))G_2)$ for $i \in \{1, \quad , q_{\mathcal{H}_W}\}\backslash\{\ell\}$ The public key Q_{pub} is chosen as

$$Q_{pub} = -U - I_\ell G_2 = (-\alpha - I_\ell)G_2$$

so that its (unknown) private key is implicitly set to $x = -\alpha - I_\ell \in \mathbb{Z}_q^*$ For all $i \in \{1, \quad , q_{\mathcal{H}_W}\}\backslash\{\ell\}$, we have $(I_i, -H_i) = (I_i, (1/(I_i + x))G_2)$

In addition $\mathcal{B}$ generates a random value $y \xleftarrow{R} \mathbb{Z}_q^*$, and publishes $e(P,Q)^y$ $\mathcal{B}$ then initializes a counter ν to 1 and starts the adversary $\mathcal{A}$ on input of (G_1, G_2, Q_{pub}) Throughout the game, we assume that $\mathcal{H}_W$-queries are distinct, that the target keywords W_0^*, W_1^* are submitted to $\mathcal{H}_W$ at some point and that any query involving a keyword comes after a $\mathcal{H}_W$-query on it

- $\mathcal{H}_W$-queries (let us call W_ν the input of the ν^{th} one of such queries) $\mathcal{B}$ answers I_ν and increments ν

- $\mathcal{H}_{\mu_T}$-queries on input $\gamma_j \in G_T$ $\mathcal{B}$ returns a random $B_j \xleftarrow{R} \{0,1\}^n$ and stores the pair (γ, B_j) in list L_2

- Trapdoor queries on an input of a keyword W_ν if $\nu = \ell$, then the simulator fails Otherwise, it knows that $\mathcal{H}_W(W_\nu) = I_\nu$ and returns $-H_\nu = (1/(I_\nu + x))\, G_2 \in \mathcal{G}_2$

At the challenge phase, $\mathcal{A}$ outputs two distinct keywords (W_0^*, W_1^*) for which she never obtained the trapdoors If $W_0^*, W_1^* \neq W_\ell$, $\mathcal{B}$ aborts Otherwise, we may assume wlog that $W_0^* = W_\ell$ (the case $W_1^* = W_\ell$ is treated in the same way) It picks $\xi \stackrel{R}{\leftarrow} \mathbb{Z}_q^*$ and $B^* \stackrel{R}{\leftarrow} \{0,1\}^n$ to return the challenge $S^* = [A^*, B^*]$ where $A^* = -\xi G_1 \in \mathcal{G}_1$ If we define $\rho = \xi/\alpha$ and since $x = -\alpha - I_\ell$, we can check that

$$A^* = -\xi G_1 = -\alpha\rho G_1 = (I_\ell + x)\rho G_1 = \rho I_\ell G_1 + \rho\psi(Q_{pub})$$

$\mathcal{A}$ cannot recognize that S^* is not a proper ciphertext unless she queries $\mathcal{H}_{\mu_r}$ on $e(A^*, G_2^{(y/(x+\mathcal{H}_W(W_0^*))}) = e(G_1, G_2)^{y\rho}$ nor $e(A^*, G_2^{(y/(x+\mathcal{H}_W(W_1^*))})$ Along the second stage, her view is simulated as before and her eventual output is ignored Standard arguments can show that a successful $\mathcal{A}$ is very likely to query $\mathcal{H}_{\mu_r}$ on either $e(A^*, G_2^{(y/(x+\mathcal{H}_W(W_0^*))}) = e(G_1, G_2)^{y\rho}$ or $e(A^*, G_2^{(y/(x+\mathcal{H}_W(W_1^*))})$ if the simulation is indistinguishable from a real attack environment

Let $\mathsf{AskH_2}$ denote this event In a real attack, we have

$$\Pr[\mathcal{A} \text{ wins}] \leq \Pr[\mathcal{A} \text{ wins} | \neg\mathsf{AskH_2}]\Pr[\neg\mathsf{AskH_2}] + \Pr[\mathsf{AskH_2}]$$

Clearly, $\Pr[\mathcal{A} \text{ wins} | \neg\mathsf{AskH_2}] = 1/2$ and $\Pr[\mathcal{A} \text{ wins}] \leq 1/2 + (1/2)\Pr[\mathsf{AskH_2}]$ On the other hand, we have

$$\Pr[\mathcal{A} \text{ wins}] \geq \Pr[\mathcal{A} \text{ wins} | \neg\mathsf{AskH_2}](1 - \Pr[\mathsf{AskH_2}]) = \frac{1}{2} - \frac{1}{2}\Pr[\mathsf{AskH_2}]$$

It comes that $\epsilon \leq |\Pr[\mathcal{A} \text{ wins}] - 1/2| \leq \frac{1}{2}\Pr[\mathsf{AskH_2}]$ and thus $\Pr[\mathsf{AskH_2}] \geq 2\epsilon$ This shows that, provided the simulation is consistent, $\mathcal{A}$ issues a $\mathcal{H}_{\mu_r}$-query on either $e(A^*, G_2^{(y/(x+\mathcal{H}_W(W_0^*))})$ or $e(A^*, G_2^{(y/(x+\mathcal{H}_W(W_1^*))})$ at some point of the game with probability at least ϵ With probability ϵ, a $\mathcal{H}_{\mu_r}$-query involving $e(A^*, G_2^{(y/(x+\mathcal{H}_W(W_0))}) = e(G_1, G_2)^{y\rho}$ will be issued To produce a result, $\mathcal{B}$ fetches a random record from the lists L_2 With probability $1/q_{\mathcal{H}_{\mu_r}}$, the chosen record contains the right element $r = e(G_1, G_2)^{y\rho} =$

$e(P,Q)^{yf(\alpha)^2\xi/\alpha}$, where $f(z) = \sum_{i=0}^{p-1} c_i z^i$ is the polynomial for which $G_2 = f(\alpha)Q$ The p-BDHIP solution can be extracted by noting that, if $\gamma^* = e(P,Q)^{1/\alpha}$, then

$$e(G_1,G_2)^{1/\alpha} = \gamma^{*(c_0^2)} e\Big(\sum_{i=0}^{p-2} c_{i+1}(\alpha^i P), c_0 Q\Big) e\Big(G_1, \sum_{j=0}^{p-2} c_{j+1}(\alpha^j) Q\Big)$$

In an analysis of $\mathcal{B}$'s advantage, we note that it only fails in providing a consistent simulation because one of the following independent events

E_1 $W_0^*, W_1^* \neq W_\ell$

E_2 $\mathcal{B}$ aborts when answering a trapdoor query

We clearly have $\Pr[\neg E_1] = (q_{\mathcal{H}_W} - 1)/\binom{q_{\mathcal{H}_W}}{2} = 2/q_{\mathcal{H}_W}$ and we know that $\neg E_1$ implies $\neg E_2$ We thus find $\Pr[\neg E_1 \wedge \neg E_2] = 2/q_{\mathcal{H}_W}$ It follows that $\mathcal{B}$ outputs the correct result with probability $2\epsilon/(q_{\mathcal{H}_W} q_{\mathcal{H}_{\mu_r}})$ □

Appendix C

Timings for Signatures with Pre-Computation

Table C 1 Efficiency comparison

	Verify				
signature scheme	exp	mul	pairings	storage	time (ms)
SOK			2	$n\mu_r$	344ms
Paterson_1	2		1	$n\mu_r$	182ms
Paterson_2	1		1	$n\mu_r$	177ms
ChaCheon		1	2		438ms
Hess	1		2	$n\mu_r$	177ms
$\text{SK}_{(ElGamal)}$		2	2		532ms
$\text{SK}_{(Schnorr)}$		1	2	$n\mu_r$	266ms
BLMQ (Ours)	1		1		**177ms**

Appendix D

Security proof for Smart's Key Agreement Protocol

Theorem D 0 1 *Smart's key agreement protocol is a secure AK protocol, assuming that E is does not make any reveal queries and that the hash functions used are random oracles*

Proof **Condition 1** holds as follows Both oracles accept holding the same session key as a direct result of the commutativity of exponentiation of members of the group $\mathcal{G}$ The session key is distributed uniformly at random by the fact that both oracles generate truly random $x \in_R \mathbb{Z}$ Therefore the product of these elements will also be random Since the exponent is random, and $g = e(P, P)$ is a generator of the group $\mathcal{G}$, and $\mathcal{H}_k$ is a random oracle, the session key will be uniformly distributed over $\{0, 1\}^k$

Condition 2 holds by the fact that if they have matching conversations then the communication was generated entirely by the two oracles Therefore, by the bilinearity of the pairing and the commutativity of exponentiation they accept and hold the same session key

Condition 3 holds as follows Consider by contradiction that $Advantage^E(\kappa)$ is non-negligible Then we can construct from E an algorithm $\mathcal{F}$ that solves the BDHP with non-negligible advantage $\mathcal{F}$ is given as input the output of the BDH generator $\mathcal{B}$ $\mathcal{F}$'s task is to solve the BDHP, namely, given P, aP, bP and cP, compute $v = g(P, P)^{abc}$

All queries by the adversary E now pass through $\mathcal{F}$ The following queries are allowed to be made by E

$\mathcal{F}$ starts the simulation by setting the value P and bP to be the KGC's generator point and master public key respectively These values, along with $\mathcal{G}$, are provided to the adversary E $\mathcal{F}$ also keeps two, initially empty, lists for keeping track of random oracle queries by E The first list, $\mathcal{H}_{ID}$, stores tuples of the form (ID_i, r_i), where $r_i \in_R \mathbb{Z}_p^*$ This will be explained later The second list, $\mathcal{H}_k$, stores tuples of the form $(\mu_r, \{0,1\}^k)$

Create For the j-th oracle $\mathcal{F}$ answers aP, otherwise $\mathcal{F}$ checks to see if ID_i already exists on H_{ID} If it does $\mathcal{F}$ retrieves the corresponding value r_{ID} and creates the public and private keys as $r_{ID}P$ and $r_{ID}bP$ respectively If H_{ID} does not contain ID then $\mathcal{F}$ chooses $r_{ID} \in_R \mathbb{Z}_p^*$ and (ID, r_{ID}) is added to H_{ID} $\mathcal{F}$ creates a public key as $ID = r_{ID}P$, and computes the private key as $r_{ID}bP$ However, for the j-th oracle $\mathcal{F}$ answers aP Since $\mathcal{F}$ does not know a, it cannot calculate abP, the correct private key for this oracle

$\mathcal{H}_k$ E is allowed, at any time, to access the $\mathcal{H}_k$ oracle on any input in the input domain (elements of μ_r) $\mathcal{H}_k$ is modelled as a random oracle of the type $\mathcal{H}_k$ $\mu_r \rightarrow \{0,1\}^k$, and so the query will return a value in $\{0,1\}^k$

Corrupt $\mathcal{F}$ answers Corrupt queries in the usual way, revealing the private key of the oracle being queried However, $\mathcal{F}$ does not know the private key for oracle j If E asks a Corrupt query on oracles j, $\mathcal{F}$ aborts and returns the $\perp$ symbol

Send $\mathcal{F}$ answers all send queries m the usual way, except if E asks Send $\prod_{i,j}^{n}$, for any n, $\mathcal{F}$ generates a random $s_n \in \mathbb{Z}_r^*$ and answers $s_n cP$ Remember that $\mathcal{F}$ does not know the value c This is part of the BDH problem that $\mathcal{F}$ hopes to solve with E's help

Reveal E is not allowed to make reveal queries

Test At some point E will ask a single Test query of some oracle, which we assume is some oracle $\prod_{i,j}^{n}$[1], if it is not, $\mathcal{F}$ aborts and returns the $\perp$ symbol Since it is picked it

[1] An oracle i, having had a conversation with j

must have Accepted, and not be Corrupted. Assuming that it received some value δP prior to accepting, it must be holding a session key of the form $\mathcal{H}_{\mu_r}(e(abP, s_ncP)\ e(r_ibP, \delta P))$ which is j's private key paired with the value it received, times i's private key paired with the value it received. However, $\mathcal{F}$ cannot compute this key and hence cannot simulate the query, so it simply outputs a random element of $\{0,1\}^k$.

If $\mathcal{F}$ does not abort and E does not detect $\mathcal{F}$'s inconsistency in answering the Test query then its advantage in predicting the correct session key still is $Advantage^E(\kappa)$. For this to be non-negligible, E must have queried $e(abP, s_ncP)\ e(r_ibP, \delta P)$ to the oracle $\mathcal{H}_{\mu_r}$, given s_ncP as input from $\mathcal{F}$, and δP, a value purportedly from j, with some non-negligible advantage κ'.

If, at the end of E's attack, E does not detect any inconsistencies in $\mathcal{F}$'s responses, and $\mathcal{F}$ does not abort, then $\mathcal{F}$ picks E's l^{th} query to the $\mathcal{H}_k$ oracle. $\mathcal{F}$ guess's this to be $k = e(abP, s_ncP)\ e(r_ibP, \delta P)$ for some s_n. It can calculate $e(abP, s_ncP)$ since it knows $\gamma = e(r_ibP, \delta P)$. For clarity $(k/\gamma)^{s_n^{-1}} = g^{abc}$ - this is j's private key paired with the value it received (actually $\mathcal{F}$ in this case). Hence, $\mathcal{F}$ has non-negligible advantage in solving the BDH problem.

We assume that there is some timeout τ_s on the length of a run of the protocol including the time spent in the $*$ state. We also assume that some time τ_c is allocated to allow the construction of oracles in the Create query, and time τ_o allocated for each Corrupt query. We assume that γ oracles are needed, and that s send queries are needed, and o corrupt queries are needed. $\mathcal{F}$ will abort if E does not pick, for its test query, oracle i in conversation with oracle j - there are n of these, with s messages in total. It will also abort if the Corrupt query is asked for oracles i or j. It will also fail if it does pick the correct $\mathcal{H}_k$ random oracle query. The expected time needed to solve the BDHP is

$$\frac{(\gamma\tau_c)(s\tau_s)(o\tau_o)2n\kappa'}{csl}$$

□

Appendix E

Security Proof for the McCullagh-Barreto Key Agreement

Proof The conditions 1 and 2 directly follow from the protocol specification The protocol satisfies the condition 3 if the Reveal query is disallowed

Suppose that there is an adversary A against the protocol with non-negligible probability Let q_1 and q_2 be the number of the distinct queries to $\mathcal{H}_W$ and $\mathcal{H}_{\mu_r}$ respectively (note that $\mathcal{H}_W$ could be queried directly by an $\mathcal{H}_W$-query or indirectly by a Corrupt query or a Send query) With the help of A, we can construct an algorithm B to solve a k-EBCAA1 problem with non-negligible probability

B simulates the Setup algorithm to generate the system params $(\mathcal{G}, \mu_r e, k, P, sP, \mathcal{H}_W, \mathcal{H}_{\mu_r})$ (i e , using s as the master key which it does not know) $\mathcal{H}_W$ and $\mathcal{H}_{\mu_r}$ are two random oracles controlled by B Suppose, in the game, there are T_1 oracles created by the engaged parties and A Here, we slightly abuse the notation $\prod_{ij}^{s}$ as the s-th oracle among all the oracles initiated by all the parties or the adversary, instead of the s-th instance of i This change does not affect the soundness of the model because s originally is just used to uniquely identify an instance of party i B randomly chooses

$u \in_R \{1, \ldots, T\}$ and $I \in_R \{1, \ldots, q\}$ and interacts with A in the following way

- $\mathcal{H}_W$-queries (ID_i) B maintains a list of tuples (ID_j, h_j, d_j) as explained below We refer to this list as $\mathcal{H}_W$-list The list is initially empty When A queries the oracle $\mathcal{H}_W$ at a point ID_i, B responds as follows

  1 If ID_i already appears on the $\mathcal{H}_W$-list in a tuple (ID_i, h_i, d_i), then B responds with $\mathcal{H}_W(ID_i) = h_i$

  2 Otherwise, if the query is on the I-th distinct ID, then B stores $(ID_I, h_0, \perp)$ into the tuple list and responds with $\mathcal{H}_W(ID_I) = h_0$

  3 Otherwise, B selects a random integer $h_i(i > 0)$ from the k-EBCAA1 instance which has not been chosen by B and stores $(ID_i, h_i, (h_i + s)^{-1}P)$ into the tuple list B responds with $\mathcal{H}_W(ID_i) = h_i$

- $\mathcal{H}_{\mu_r}$-queries (X_i) At any time A can issue queries to the random oracle $\mathcal{H}_{\mu_r}$ To respond to these queries B maintains a list of tuples called $\mathcal{H}_{\mu_r}$-list Each entry in the list is a tuple of the form (X_i, H_i) indexed by X_i To respond to a query on X_i, B does the following operations

  1 If on the list there is a tuple indexed by X_i, then B responds with H_i

  2 Otherwise, B randomly chooses a string $H_i \in \{0,1\}^n$ and inserts a new tuple (X_i, H_i) to the list It responds to A with H_i

- Corrupt(ID_i) B looks through list $\mathcal{H}_W$-list If ID_i is not on the list, B queries $\mathcal{H}_W(ID_i)$ B checks the value of d_i if $d_i \neq \perp$, then B responds with d_i, otherwise, B aborts the game

- Send$(\prod_{j,i}^t, M)$ B first looks through the list $\mathcal{H}_W$-list If ID_i is not on the list, B queries $\mathcal{H}_W(ID_i)$ After that, B checks the value of t If $t \neq u$, B responds to the query by correctly following the protocol If $t = u$, B further checks the value of d_i, and then responds the query differently as below depending on this value

1 If $d_i \neq \perp$, B aborts the game We note that only one party's private key is represented as $\perp$ in the whole simulation

2 Otherwise, B responds with yP obtained from the k-EBCAA1 instance

Note that $\prod_{j,i}^{t}$ can be the initiator (if $M = \lambda$) or the responder (if $M \neq \lambda$)

- Test($\prod_{j,i}^{t}$) If $t \neq u$, B aborts the game Otherwise, B randomly chooses a number $\gamma \in \{0,1\}$ and gives it to A as the response When A responds, B randomly chooses a tuple from $H-2$-list with value X_l B responds to the k-EBCAA1 challenger with the value of $X_l = e(d_j, M)$ where M is the incoming message to oracle $\prod_{j,i}^{t}$

  Note that if the game did not abort, the adversary cannot find the inconsistency between the simulation and the real world The agreed secret in oracle $\prod_{j,i}^{t}$ should be $K = e(d_j, M) \ \ e(P,P)^r$ where $r(h_0P + sP) = yP$ (recall that party i's public key is $h_0P + sP$ and the private key is unknown to B and represented by $\perp$), i e $r = y(h_0 + s)$ and $K = e(d_j, M) \ \ e(yP, (h_0 + s)^{-1}P)$

□

We do not repeat the full expected running time analysis here, the interested reader is advised to read [48]