# A Java Library for Identity Based Cryptography

**Sakorafas Vasileios**

Master of Science in Network Computing



TECHNOLOGICAL EDUCATIONAL INSTITUTE OF LARISSA

January 2011

TECHNOLOGICAL EDUCATIONAL INSTITUTE OF LARISSA

# A Java Library for Identity Based Cryptography

**Sakorafas Vasileios**

Master of Science in Network Computing

**Thesis Summary**

This Thesis provides a library for Identity Based Cryptography (IBC) that is able to be used for key generation, encryption and decryption in Identity Based systems. At first, necessary background of Cryptography and a theoretic analysis of IBC are provided. Then, the programming library for Java is analyzed regarding the implementation, tests and evaluation. The implemented Identity Based Encryption (IBE) scheme is the Sakai-Kasahara IBE. The outcome of the Thesis could be used for further academic research on IBC and applied on desktop computers, but also smart phones and other mobile computing devices.

**Keywords:** Identity Based Cryptography, Sakai – Kasahara, Java

# Table of Contents

# List of Figures

# List of Tables

# 1. Introduction

Cryptography is an ancient subject. The term comes from the two greek words kryptos (κρυπτός) and graphin (γράφειν) which mean secret and writing respectively. Kahn's encyclopedic work [20] discusses the evolution of the subject from ancient times till the time of the publication of the book in 1967. As we can see in [45] *Cryptography* is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication. It is not the only means of providing information security, but rather one set of techniques.

Cryptographic techniques are typically divided into two generic types: *symmetric-key* and *public-key*. In symmetric-key cryptography sender and the receiver share the same *secret key*, which is small in size but must be distributed in an efficient (and secret) way. In public-key cryptography two different but mathematically related keys are used: the *public key* and the *private key*. Here there is a need for larger keys, but the public key can be distributed through unsecured channels.

*Identity Based Cryptography* (IBC) is a special category of public-key cryptography, where users do not have explicit public keys. Instead, public identification information of a person or an entity can serve as its public-key. Regarding authenticity, user-specific public data is not explicitly verified, as is necessary for user public keys in certificate-based systems. The inherent redundancy of user public data in Identity Based systems with the use of authentic public system data implicitly protects against forgery.

In this Thesis, Identity Based Cryptography and related issues are going to be examined. There will be a programming approach, in the form of a Java library, towards a scheme that will provide Identity Based key-generation, encryption and decryption.

## 1.1. Motivation

Identity Based Cryptography is a rising topic the last few years. The mostly discussed and implemented Identity Based Encryption (IBE) scheme is the Boneh-Franklin IBE [4] and the only available comparisons with other schemes is strictly theoretic. So, there are some rising questions in the current state:

i. Is there another efficient IBE scheme that could be implemented?
ii. What would its performance be?
iii. Which advantages and disadvantages does it have once implemented?
iv. Why would anyone use this newly implemented scheme?

## 1.2. Aims of the Thesis

In public key cryptography, random parameters are used for encryption and decryption. IBC derives its keys from publically known attributes of the recipient. The analysis of current IBC applications leads to the conclusion that IBC generally outperforms public key cryptography, especially in terms of resource friendliness. However, IBC is not widely known and used. One reason for that is the absence of tools that could be globally used to provide IBC services.

The goal of this Thesis is to provide such a tool. At first, relative literature is going to be studied and discussed. The actual aim is to create a library for Identity-Based Cryptography that will be able to be used for key generation in Identity-Based systems. Another aim is to implement an IBE scheme both from the symmetric and the asymmetric pairing.

Java is a suitable choice as a programming language because it is open source, widely spread (computers, mobile phones, etc) and contains libraries for elliptic curve cryptography and big integer computations. It can also be used for the implementation of web services and for providing a program interface.

The overall scheme is going to be tested and evaluated according to several criteria, like time needed for the keys to be generated. It is also possible to check the execution time, depending on the key-size. The goal is to provide a scheme that is effective, providing the necessary security with the least possible resources (time, etc).

## 1.3. Research Methodology

### 1.3.1. Literature Review

The literature related to this Thesis includes general cryptography topics [20, 45], Identity Based Cryptography (general, but also in depth) and mathematic schemes, like elliptic curves [41, 50] and bilinear pairings [1, 12, 17]. Most of the journals/books regarding these topics are available online (Springerlink, ACM, IEEE) and can be accessed through the computing facilities of TEI of Larisa and the VPN account that is provided. Google Scholar and Microsoft Academic Search are also tools that can be used for finding suitable literature. There is also a library in the facilities of TEI, where useful resources can be found.

### 1.3.2. Analysis and Investigation

This thesis can be divided in two parts: First, a theoretic analysis of Cryptography in general, Identity Based Cryptography and of related issues in particular and the necessary mathematic issues. Second, a programming approach towards a scheme that will provide Identity Based key-generation, encryption and decryption.

The theoretic analysis is necessary in order for the appropriate knowledge and IBC background to be provided. Appropriate algorithms and key-sizes must be selected for the implementation part. Existing libraries have to be checked and evaluated according to their restrictions and the outcomes that they offer. There are various schemes that need to be brought together (like encryption, decryption, elliptic curves and programming libraries) and combined under a cryptography scheme implemented in Java.

The various tests that are going to be implemented will also provide some extra knowledge on how this IBC library will work, the computational cost that will be implied and various restrictions that may be encountered.

In order to achieve the goals of this Thesis, there are various topics that need to be studied in detail, which include:

- the Java programming language and its libraries that are of possible use,
- the IBC architecture and proposed algorithms for key-generation, encryption and decryption
- elliptic curves and bilinear pairings

### 1.3.3. Prototyping

The system developed is the Sakai-Kasahara IBE scheme. It is implemented as a Java library. It offers the ability to produce a Master public and private key, a user's private key and the encrypt and decrypt operations. The scheme is implemented both from the symmetric and the asymmetric pairing.

### 1.3.4. System Evaluation

The developed system is tested and evaluated according to several criteria, like time needed for the keys to be generated. The execution time, depending on the key-size, is tested. There are also tests regarding execution time for different sizes of the

messages to be encrypted-decrypted. Last, for all the above evaluation points, there is a comparison between the SK-IBE scheme and BF-IBE scheme.

## 1.4. Novel Features of the Thesis

Identity Based Cryptography has received much attention in the recent years. Existing algorithms can be used to generate a cryptographic scheme from the beginning. There have been some implementations of IBE schemes, but most of them concern the Boneh-Franklin IBE and are written in C language. This Thesis provides an implementation of the Sakai-Kasahara IBE scheme [6] in Java. As far as we are concerned there has been no such implementation yet. Providing such a programming library will simplify the implementation of IBC and at the same time be a motivation for it to be used and studied more widely.

The choice of an open source and widely spread programming language (Java) is possible to ease the implementation of cryptographic schemes in various computing devices. The additional evaluation and tests that are going to be provided are possible to contribute to further understanding practical implementations of IBC.

## 1.5. Outline of the Thesis

The structure of this Thesis is as follows: In Chapter 2 the necessary background is provided. It firstly covers the general area of cryptography (symmetric, asymmetric, digital signatures). The mathematical background, elliptic curves and bilinear maps are then studied. There is an overview of the Tate pairing, Miller's algorithm and the criteria for the choice of the appropriate curves for IBE. Identity Based Cryptography is then discussed and includes the characteristics of an IBE scheme, Key Management Systems, IBC applications, Identity Based Signature schemes and a general aspect of IBC Security and Key-sizes. Information about two main IBE schemes (Boneh-Franklin IBE and Sakai-Kasahara IBE) and a theoretic comparison between them, regarding security and performance is also provided. Regarding In Chapter 3 the Design and Analysis of the Developed System is provided. It firstly includes the description of the Sakai-Kasahara algorithms that are implemented and the Java tools and libraries that are used. The Development Strategy and the Structure of the implemented library are then analyzed, including variations from the original scheme, description of the implemented classes and the class diagrams. The overall scheme is then tested for different key and plaintext sizes and results are commented on and compared to those of Boneh-Franklin IBE scheme. Finally, in Chapter 4 an overall evaluation, conclusions and future work and are discussed.

# 2. Background

## 2.1. Background of cryptography

In [45] general aspects of cryptography are studied. Regarding cryptography, there are four main goals:

a. *Confidentiality:* a service used to keep the content of information from all but those authorized to have it. There are numerous approaches to providing confidentiality, ranging from physical protection to mathematical algorithms which render data unintelligible.
b. *Data integrity:* a service which addresses the unauthorized alteration of data. To assure data integrity, one must have the ability to detect data manipulation by unauthorized parties. Data manipulation includes insertion, deletion, substitution etc.
c. *Authentication:* a service related to identification. It applies to both entities and information itself. Two parties entering into a communication should identify each other. Information delivered over a channel should be authenticated as to origin, date of origin, data content, time sent etc. So, this aspect of cryptography can be subdivided into two major classes: *entity authentication* and *data origin authentication*. Data origin authentication implicitly provides data integrity (for if a message is modified, the source has changed).
d. *Non-repudiation:* a service which prevents an entity from denying previous commitments or actions. When disputes arise due to an entity denying that certain actions were taken, a means to resolve the situation is necessary. A procedure involving a trusted third party is needed to resolve the dispute.

These four goals must be adequately addressed both in theory and practice. Cryptography is about the prevention and detection of cheating and other malicious activities.

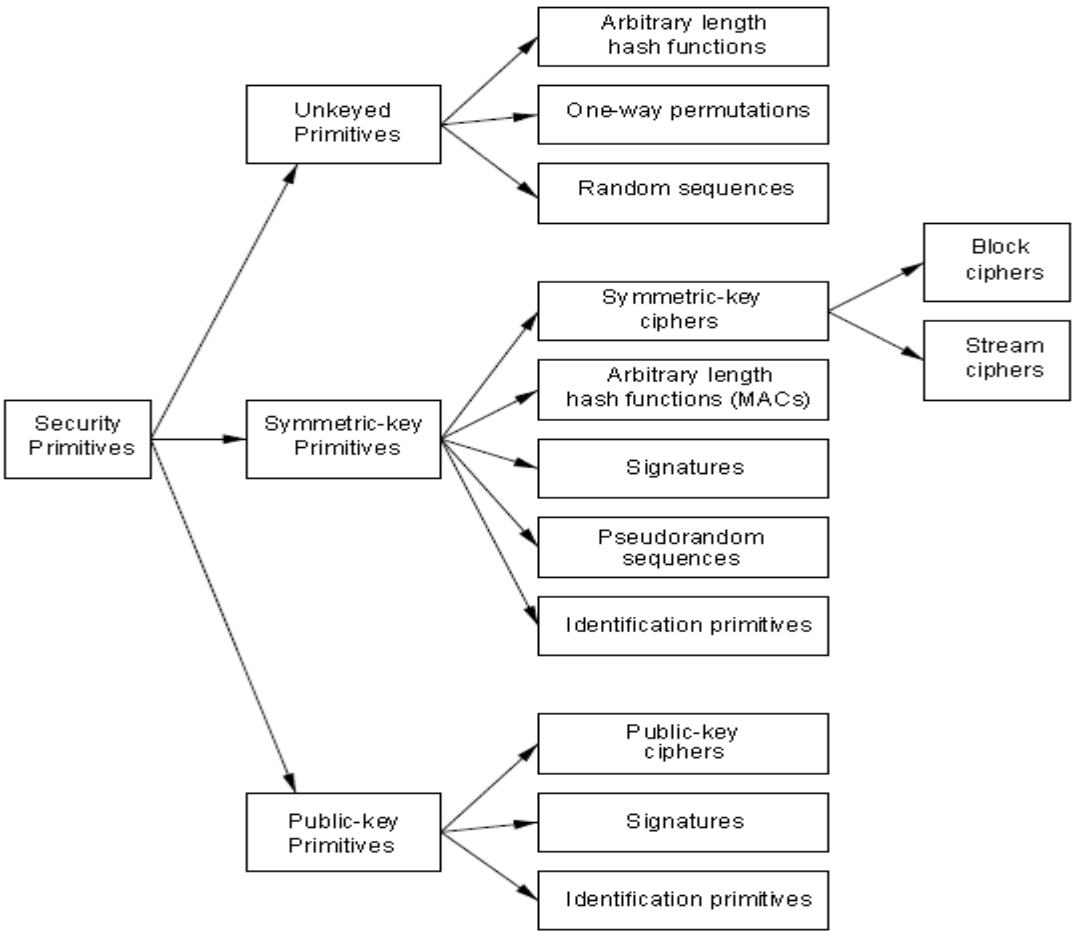


Figure 2.1: A schematic listing of primitives and how they relate

Figure 2.1 provides a schematic listing of the primitives considered and how they relate. These primitives should be evaluated according to various criteria such as:

a. *Level of security:* This is usually difficult to quantify. Often it is given in terms of the number of operations required to defeat the intended objective. Typically the level of security is defined by an upper bound on the amount of work necessary to defeat the objective.
b. *Functionality:* Primitives will need to be combined to meet various information security objectives. Which primitives are most effective for a given objective will be determined by the basic properties of the primitives.
c. *Methods of operation:* Primitives, when applied in various ways and with various inputs, will typically exhibit different characteristics. One primitive could provide very different functionality depending on its mode of operation or usage.
d. *Performance:* This refers to the efficiency of a primitive in a particular mode of operation (e.g. an encryption algorithm may be rated by the number of bits per second which it can encrypt).
e. *Ease of implementation:* This refers to the difficulty of realizing the primitive in a practical instantiation. This might include the complexity of implementing the primitive in either a software or hardware environment.

As mentioned above, cryptographic techniques are typically divided into two generic types: *symmetric-key* and *public-key*.

### 2.1.1. Symmetric-key cryptography

Symmetric-key cryptography is a method in which the sender and the receiver share the same secret key.

Let's consider an encryption scheme consisting of the sets of encryption and decryption transformations $\{E_e : \varepsilon \in K\}$ and $\{D_d : d \in K\}$, respectively, where K is the key space. The encryption scheme is said to be *symmetric-key* if for each associated encryption/ decryption key pair (ε, d), it is computationally "easy" to determine d knowing only e, and to determine e from d. Since e = d in most practical symmetric-key encryption schemes, the term symmetric-key becomes appropriate. Other terms used in the literature are *single-key*, *one-key*, *private-key*, and *conventional* encryption.

A two-party communication using symmetric-key encryption can be described below in figure 2.2:

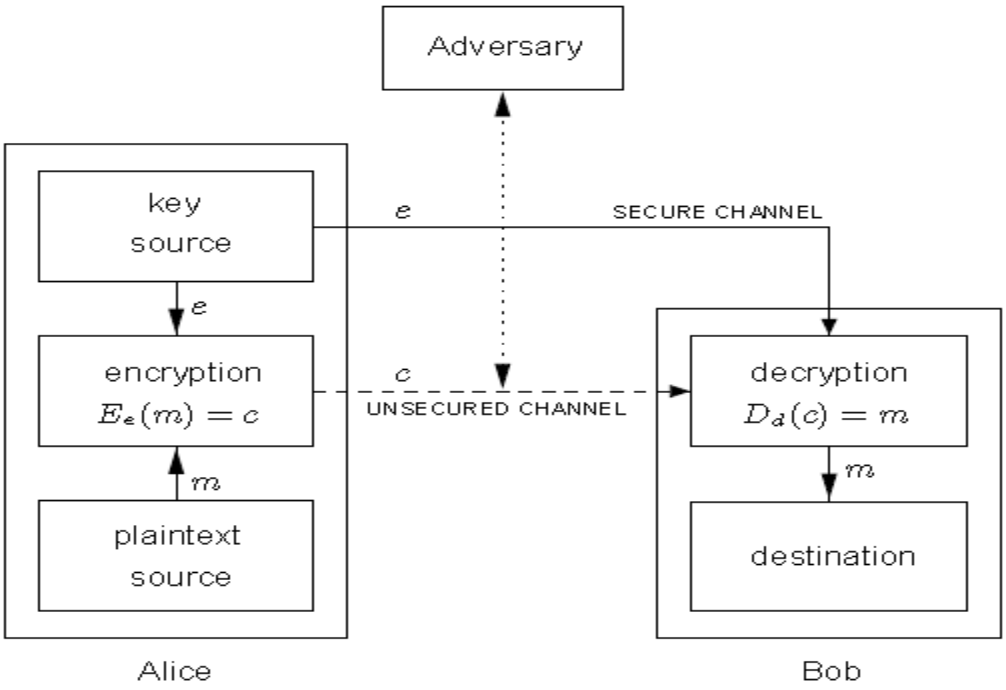


**Figure 2.2: Two-party communication using encryption, with a secure channel for key exchange. The decryption key d can be efficiently computed from the encryption key e.**

A major issue with symmetric-key systems is to find an efficient method to agree upon and exchange keys securely. This problem is referred to as the *key distribution problem.*

### 2.1.2. Public-key cryptography

In Public-key (or asymmetric) cryptography two different but mathematically related keys are used: the *public key* and the *private key*.

Let's consider an encryption scheme consisting of the sets of encryption and decryption transformations $\{E_e : \varepsilon \in K\}$ and $\{D_d : d \in K\}$, respectively. The encryption method is said to be a *public-key encryption scheme* if for each associated

encryption/decryption pair (ε, d), one key ε (the *public key*) is made publicly available, while the other d (the *private key*) is kept secret. For the scheme to be *secure*, it must be computationally infeasible to compute d from ε.

Let $\{E_e : \varepsilon \in K\}$ be a set of encryption transformations, and let $\{D_d : d \in K\}$be the set of corresponding decryption transformations, where K is the key space. Consider any pair of associated encryption/decryption transformations ($E_e$ , $D_d$) and suppose that each pair has the property that knowing $E_e$ it is computationally infeasible, given a random ciphertext $c \in C$, to find the message $m \in M$ such that $E_e(m) = c$. This property implies that given ε it is infeasible to determine the corresponding decryption key d. (Of course ε and d are simply means to describe the encryption and decryption functions, respectively.) $E_e$ is being viewed here as a trapdoor one-way function with d being the trapdoor information necessary to compute the inverse function and hence allow decryption. This is unlike symmetric-key ciphers where ε and d are essentially the same. Under these assumptions, consider the two-party communication between Alice and Bob illustrated in Figure below. Bob selects the key pair (ε, d). Bob sends the encryption key ε (called the *public key*) to Alice over any channel but keeps the decryption key d (called the *private key*) secure and secret. Alice may subsequently send a message *m* to Bob by applying the encryption transformation determined by Bob's public key to get $c = E_e(m)$. Bob decrypts the ciphertext c by applying the inverse transformation $D_d$ uniquely determined by d.

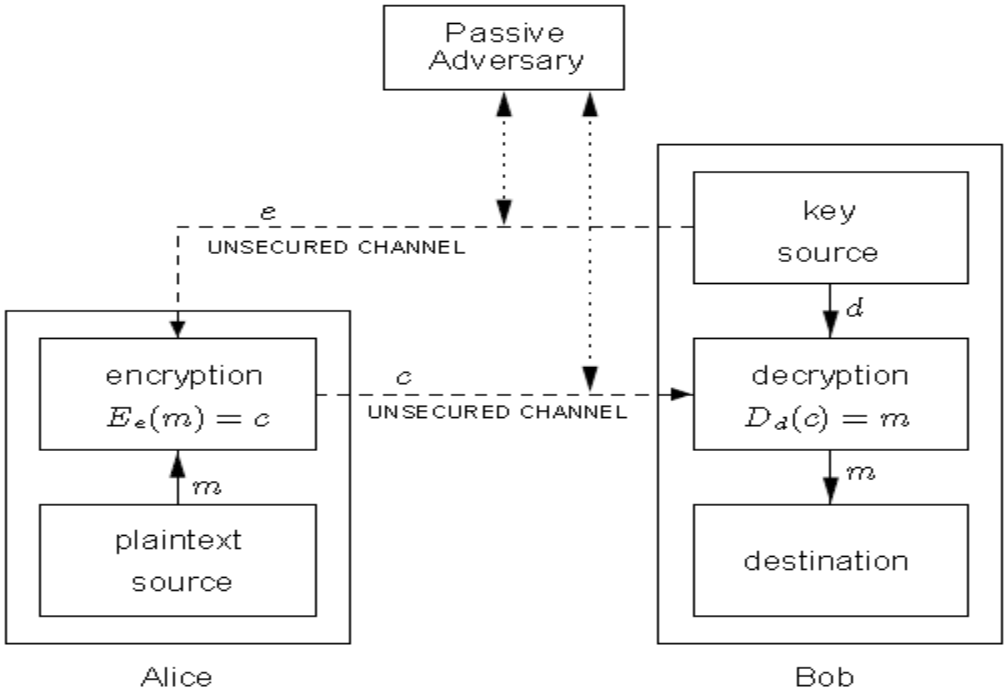


Figure 2.3: Encryption using Public-key techniques

Here the encryption key is transmitted to Alice over an unsecured channel. This unsecured channel may be the same channel on which the ciphertext is being transmitted. Since the encryption key e need not be kept secret, it may be made public. Any entity can subsequently send encrypted messages to Bob which only Bob can decrypt.

### 2.1.3. Digital Signatures

A cryptographic primitive which is fundamental in authentication, authorization, and non-repudiation is the *digital signature*. The purpose of a digital signature is to provide a means for an entity to bind its identity to a piece of information. The process of *signing* entails transforming the message and some secret information held by the entity into a tag called a *signature*.

**Nomenclature and set-up**

- $M$ is the set of messages which can be signed.
- $S$ is a set of elements called *signatures*, possibly binary strings of a fixed length.
- $S_A$ is a transformation from the message set $M$ to the signature set $S$, and is called a *signing transformation* for entity A. The transformation $S_A$ is kept secret by A, and will be used to create signatures for messages from $M$.
- $V_A$ is a transformation from the set $M \times S$ to the set $\{true, false\}$. $V_A$ is called a *verification transformation* for A's signatures, is publicly known, and is used by other entities to verify signatures created by A.

The transformations $S_A$ and $V_A$ provide a *digital signature scheme* for A.

**Signing procedure**

Entity A (the *signer*) creates a signature for a message $m \in M$ by doing the following:

1. Compute $s = S_A(m)$.
2. Transmit the pair $(m, s)$. s is called the *signature* for message m.

**Verification procedure**

To verify that a signature s on a message m was created by A, an entity B (the *verifier*) performs the following steps:

1. Obtain the verification function $V_A$ of A.
2. Compute $u = V_A(m, s)$.

Accept the signature as having been created by A if $u = true$, and reject the signature if $u = false$.

## 2.2. Mathematical Background

### 2.2.1. Elliptic Curves and Bilinear Maps

In [43] elliptic curves and bilinear pairings are studied in general. Elliptic Curve Cryptography (ECC) has become an active area of research since the seminal works of Koblitz and Miller [21], [29]. Pairing based cryptography builds on the foundations laid by ECC and since the work of Boneh and Franklin [4], pairing based cryptography has become one of the most active areas of research in cryptography.

These are rich and non-trivial areas of specialization and we provide a minimal introduction to these topics to enable us to get a flavor of how IBE schemes are constructed.

*Elliptic Curves*: Let q be a large prime and m an integer with $m \geq 1$. Let $F_{q^m}$ be the finite field with $q^m$ elements. Here, q denotes the characteristic of the field and m the extension degree. The multiplicative group of $F_{q^m}$ is denoted by $F^*_{q^m}$.

Then, the elliptic curve E over $F_{q^m}$ is denoted by E/$F_{q^m}$ and is defined to be the set of elements $\{x, y\} \in F_{q^m} \times F_{q^m}$ satisfying an equation of the form:

$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_5$ , where $a_i \in F_{q^m}$ for i = 1,2,3,4,5.

A point $P = \{x, y\} \in F_{q^m} \times F_{q^m}$ is said to be on the curve if it satisfies the above equation. $E\{F_{q^m}\}$ represents the set of points on the curve and together with a point at infinity denoted by $\infty$, forms an additive Abelian group.

*Bilinear Pairings*: Suppose that $E\{F_{q^m}\}$ has a cyclic subgroup G of prime order p. We define the embedding degree or security multiplier to be the least integer $k \geq 0$ such that $p|q^{km} - 1$ and $p \wedge q^l -$ for all $0 \leq l \leq k$. We let $G_T$ denote the cyclic subgroup of $F_{q^m}$ of prime order p. Then, an admissible bilinear pairing is a function e which maps a pair of elliptic curve points in G to an element in $G_T$,

$e : G \times G \rightarrow G_T$, having the following properties:

- Bilinearity: $\forall g \in G$ and $a, b \in Z^*_p$, $e(g^a, g^b) = e(g, g)^{ab}$
- Non-degenerate: $e(g_1, g_2) \neq 1$ for some $g_1, g_2 \in G$
- Efficiently Computable: There must be an efficient algorithm that computes the map e for any pair of inputs.

We note that although G is a subgroup of $E\{F_{q^m}\}$, itself an additive group, we have used multiplicative notation to denote the group operation in G. In the literature, both additive and multiplicative notation have been used to denote the group operation in G. Admissible pairings can be derived from the modified Weil pairing or the Tate pairing [4], [13]. Some recent surveys on the applications of pairings in cryptography can be found in [31] and [32]. The underlying mathematics of these known pairings is rather involved and most cryptographic papers that deal with IBC abstract away the underlying details and treat the pairing as a black box.

Elliptic curves can provide a suitable solution for key generation in IBC. For the necessary security to be provided, elliptic curve keys need to be much smaller than other type of keys. More analytically we can see the table below (Table 2.2) [18]:

| Date | Minimum of Strength | Symmetric Algorithms | Asymmetric | Discrete Logarithm | | Elliptic Curve |
|---|---|---|---|---|---|---|
| | | | | Key | Group | |
| **2007-2010** | 80 | 2TDEA | 1024 | 160 | 1024 | 160 |
| **2011-2030** | 112 | 3TDEA | 2048 | 224 | 2048 | 224 |
| **>2030** | 128 | AES-128 | 3072 | 256 | 3072 | 256 |
| **>>2030** | 192 | AES-192 | 7680 | 384 | 7680 | 384 |
| **>>>2030** | 256 | AES-256 | 15360 | 512 | 15360 | 512 |

Table 1.2: Key-length recommendations

### 2.2.2. The Tate Pairing

The Tate pairing was introduced by Frey and Ruck [14].

Definition: Consider the elliptic curve $E(F_p)[n]$. Suppose that $\#E(F_p) = hn$ where n is a prime such that gcd(n, q) = 1. Let k be a positive integer such that $n \mid p^k - 1$ with k minimal. If this is satisfied then k is called the embedding degree. Let $\mu(n) = \{a \in F_p^k \mid a^n = 1\}$ be the $n^{th}$ roots of unity.

Then according to Washington [50], the Tate pairing t can be defined as:

$t : E(Fp)[n] \times E(F_p^k) / nE(F_p^k) \rightarrow F_p^k / (F_p^k)^n$

and the modified Tate pairing t' is:

$t' : E(F_p)[n] \times E(F_p^k) / nE(F_p^k) \rightarrow \mu(n)$

Assume point $P \in E(F_p)[n]$ and point $Q \in E(F_p^k)$ with P != 0, then Tate pairing can be denoted as e(P, Q). The Tate pairing for cryptography has the following properties, as we saw for the general bilinear pairings above:

i. Non-degeneracy: This means for any given P, there always exists a Q such that e(P, Q) != 1.
ii. Bilinearity: This means for any given integers a, b and any points P and Q, $e(aP, bQ) = e(P, Q)^{ab}$.
iii. Compatibility: Let n = hn'. If $P \in E(F_p)[n]$ and $Q \in E(F_p^k)[n']$, then $t_{n'}(hP, Q) = t_n(P, Q)^h$.

### 2.2.3. Miller's Algorithm

Computing the Tate pairing is a costly process. When the pairings were used first, the best known algorithm was exponential in the size of the input. In 1986, Miller found that pairings can be achieved through divisor theory [28]. This algorithm focuses on finding the principle divisor of P with some specific line functions. The algorithm is a computationally efficient approach for Tate pairing. In the recent years, variant versions of Miller's algorithm were developed for the optimization and efficiency of Tate pairing computation [1, 17, 12, 9]. Despite all these optimizations, the time cost on pairing computation is still the most significant bottleneck of pairing based cryptography [24].

### 2.2.4. The choice of curves: Efficiency vs. Security

For the implementation of the Tate pairing, a randomly chosen curve would normally have a large embedding degree (the size of k). A large k provides resistance against the MOV [27] reduction, but also implies low speed and high computational cost. A smaller k would have benefits regarding speed and computational cost. However, it is important to have an efficient trade-off between performance and security.

Such a case is the case of *supersingular* curves. An elliptic curve E over $F_p^k$ is supersingular if $E(F_p)$ has no points of order p [41]. In [22], the efficiency and security of elliptic curves for pairing based cryptography is analyzed. The main advantage of curves with small embedding degree is the flexibility one has in choosing the two most important parameters of the system, thus the field size p and the prime order n of the basepoint $P \in E(F_p)$. One can easily get n and p both to have optimal bit-lengths and at the same time to be Solinas primes [42] (that is, the sum or difference of a small number of powers of 2). However, the use of a Solinas prime could possibly enable an attacker to use a special form of the number field sieve. Increased field sizes could be a necessary offset for the efficiency advantage provided by the use of such a prime.

Regarding the MOV reduction, we can see in [26] that any elliptic curve with a low embedding degree is susceptible to an MOV reduction, in which it is possible to reduce the problem of calculating a discrete logarithm in an elliptic curve group to calculating the discrete logarithm in a finite field. This can be done as follows:

Let $G_1$ be an elliptic curve group, $G_T$ be a multiplicative group of a finite field, and $e : G_1 \times G_1 \rightarrow G_T$ a pairing. Suppose that we have $P \in G_1$ and want to calculate the discrete logarithm of aP. If e(P, P) = g then $e(P, aP) = e(P, P)^a = g^a$, so by calculating the discrete logarithm of $g^a \in G^T$ we find the value of a. If $G_1$ is an elliptic curve group with an order of n bits, for example, calculating a discrete logarithm in $G_1$ by Pollard's rho algorithm requires $O(\sqrt{n})$ time, while calculating a discrete logarithm in $G_T$ using the index calculus algorithm requires $O(\exp((64/9)^{1/3} (\log n)^{1/3} (\log \log n)^{2/3}))$ time, which may be much less than the time to calculate a discrete logarithm in $G_1$ .

To get 80 bits of strength with an ordinary elliptic curve, a subgroup G of $E(F_p)$ with an order of 160 bits is adequate. This is based on the running time of Pollard's rho algorithm, which is roughly the same for a 160-bit group order, which is also roughly the same as the running time for the index calculus algorithm for a 1,024-bit finite field order. If we have that $E/F_p$ is supersingular with an embedding degree of k = 2, for example, then we can also calculate a discrete logarithm in G by calculating a discrete logarithm in $F_{p^2}^*$ by using the index calculus algorithm. In typical applications, the size of p is roughly the same size as $\#E(F_p)$, being no more than one or two bits larger, so we might have a 162-bit q in this case. With such a p we could use the MOV reduction to calculate discrete logarithms in G by calculating discrete logarithms in a finite field with an order of only $2 \times 162 = 324$ bits, a calculation that is much easier than calculating a discrete logarithm in a finite field with an order of 1,024 bits. It is, however, possible to attain the same levels of bit security with supersingular curves as with ordinary curves by using larger group orders. Increasing the size of this p to be 512 bits, for example, will increase $p^2$ to approximately 1,024 bits, making calculating discrete logarithms in $F_{p^2}^*$ as difficult as attacking an 80-bit symmetric key.

Note that there is nothing about supersingular curves aside from their low embedding degree that allows the MOV reduction to be carried out. Even an ordinary curve with a low embedding degree is vulnerable to the MOV reduction. Because the calculation of pairings requires a curve with low embedding degree to make the pairing calculation feasible, all such curves need to have their parameters chosen so that they are secure even if an MOV reduction is possible.

In [22] we can see a recommendation for the sizes of the group and the finite field, for curves with certain embedding degrees, in order to attain certain security levels:

| **Security (bits)** | **80** | **128** | **192** | **256** |
|---|---|---|---|---|
| **Bitlength of $p^k$ (finite field order)** | 1024 | 3072 | 8192 | 15360 |
| **Group-order for k = 1** | 1024 | 3072 | 8192 | 15360 |
| **Group-order for k = 2 (ss)** | 512 | 1536 | 4096 | 7680 |
| **Group-order for k = 2 (nss)** | 512 | 1536 | 4096 | 7680 |
| **Group-order for k = 6** | 171 | 512 | 1365 | 2560 |
| **Group-order for k = 12** | | 256 | 683 | 1280 |
| **Group-order for k = 24** | | | | 640 |

Table 2.2: Parameter sizes for certain security levels
(ss= supersingular, nss= non-supersingular)

## 2.3. Identity Based Cryptography

Identity Based Cryptography was first proposed by Shamir [40] in 1984. Identity Based systems resemble ordinary public-key systems, involving a private transformation and a public transformation, but users do not have explicit public keys

as before. Instead, the public key is replaced by (or constructed from) a user's publicly available identity information (e.g. name and network or street address). Any publicly available information which uniquely identifies a user and can be undeniably associated with the user, may serve as the identity information.

An Identity Based cryptographic system is an asymmetric system wherein an entity's public identification information (unique name) plays the role of its public key, and is used as input by a trusted authority TA (along with TA 's private key) to compute the entity's corresponding private key.

After computing it, TA transfers the entity's private key to the entity over a secure (authentic and private) channel. This private key is computed from not only the entity's identity information, but must also be a function of some privileged information known only to TA (TA's private key). This is necessary to prevent forgery and impersonation – it is essential that only TA is able to create valid private keys corresponding to given identification information. Corresponding publicly available system data must be incorporated in the cryptographic transformations of the Identity Based system, analogous to the certification authority's public key in Certificate Based systems.

**Authenticity in Identity Based systems**

Identity Based systems differ from Public Key systems in that the authenticity of user-specific public data is not (and need not be) explicitly verified, as is necessary for user public keys in certificate-based systems. The inherent redundancy of user public data in Identity Based systems, together with the use of authentic public system data, implicitly protects against forgery: if incorrect user public data is used, the cryptographic transformations simply fail. More specifically: signature verification fails, entity authentication fails, public-key encryption results in undecipherable text, and key-agreement results in parties establishing different keys, respectively, for (properly constructed) Identity Based signature, authentication, encryption, and key establishment mechanisms. The motivation behind Identity Based systems has been to create a cryptographic system modeling an ideal mail system wherein knowledge of a person's name alone suffices to allow mail to be sent which that person alone can read, and to allow verification of signatures that person alone could have produced. In such an ideal cryptographic system:

- Users need exchange neither symmetric keys nor public keys
- Public directories (files of public keys or certificates) need not be kept
- The services of a trusted authority are needed solely during a set-up phase (during which users acquire authentic public system parameters, to be maintained).

As seen in [43] the IBC approach eliminates certificates and the associated processing and management overheads from PKC. This simple idea whereby the recipient can in effect choose and appropriately manipulate the public key of the intended recipient prior to sending a message has a number of subtle advantages, enabling many interesting features. For example:

- The sender can provide a preset expiration date for the message by including either the current date or expiration date in the identifier. Then, the receiver is

only able to obtain the corresponding private key from the TA during the specified time period.

- User credentials can be managed more easily, by including the clearance level in the public key. Then, a recipient is only able to read the encrypted message if he possesses the appropriate clearance.
- Decryption keys can be delegated so that only those with particular responsibilities can read particular messages.

In this paradigm of cryptography, users' identifier information can be used as public key for encryption or signature verification. As a result, Identity Based Cryptography significantly reduces the system complexity and the cost for establishing and managing the Public Key Infrastructure. Although Shamir constructed an Identity Based Signature (IBS) scheme using the existing RSA function, he was unable to construct an Identity-Based Encryption (IBE) scheme, which became a long-lasting open problem. In 2001, this problem was solved by Boneh and Franklin [4] and Cocks [8].

In [43] it is said that while IBC potentially removes the problem of trust in the public keys, it introduces trust in the TA, which by virtue of issuing private keys to users, using its knowledge of the system-wide master secret, is now automatically a key-escrow agency. While, this is unacceptable to many users and in many application scenarios, this is exactly the property desirable in closed military, government and corporate infrastructures. These organizations want to ensure that communications between their employees are secure, so as to protect state, trade secrets etc. However, they also want to ensure that these encrypted communications can be read if the need arises, for example when an employee changes jobs. As the TA uses the system-wide master secret to compute private keys for users in the system, it can effectively recompute a private key for any arbitrary identity string at any point without ever archiving private keys. This greatly simplifies key management as the TA now only needs to store and protect the secrecy of its base secret.

An IBE scheme is defined in terms of four algorithms:

- Setup: On input $1^k$, outputs a master public key mpk which includes system parameters params, and a master secret key msk. We assume that params contains descriptions of the message and ciphertext spaces, MsgSp and CtSp. This algorithm is randomized.
- KeyDer: A key derivation algorithm that on input mpk, msk and identifier id, returns a private key $usk_{id}$. This algorithm may or may not be randomized.
- Enc: An encryption algorithm that on input mpk, identifier id and message m $\in$ MsgSp, returns a ciphertext c $\in$ CtSp. This algorithm is usually randomized. We will write c = Enc(mpk, id, m) in general. When we wish to emphasize that randomness r (drawn from some space RSp) is used when performing an encryption, we will write c = Enc (mpk, id, m: r).
- Dec: A decryption algorithm that on input mpk, a private key $usk_{id}$ and a ciphertext c $\in$ CtSp, returns either a message m $\in$ MsgSp or a failure symbol $\perp$.

It is assumed that identities are bit strings of arbitrary length, i.e. id $\in \{0,1\}^*$. However, concrete schemes may require identities to be drawn from some restricted

sets. In such situations, hashing of bit strings onto appropriate sets can be used to allow the use of arbitrary bit strings as identities. These algorithms must satisfy the standard consistency requirement that decryption undoes encryption: $\forall m \in MsgSp$, $\forall id \in \{0, 1\}^*$, $\forall usk_{id} = KeyDer(mpk, msk, id)$, if $c = Enc(mpk, id, m)$ then $Dec(mpk, uskid, c) = m$.

In [46] we can see the IBE key management system, applications and signature schemes.

### 2.3.1. Key Management System

The Voltage Corporation's white paper on IBE [49] outlines the requirements for key management in an enterprise environment:

1. Delivery of Encryption Keys for internal recipients, customers and partners.
2. Authenticate users and deliver decryption keys to users and groups specified by the data sender.
3. Jointly manage keys with partners where each partner only needs to manage keys for his own users.
4. Deliver keys to trusted infrastructure components e.g. technical and business processes such as content scanning, auditing, or anti-virus.
5. Recover Keys e.g. in scenarios where a user leaves the organization or machines lose disk storage.
6. Scale for Growth so that large transaction volumes can be managed and load-balanced and geographic deployments are possible.

These requirements are not met by the most common key management systems – symmetric key management and public key infrastructure (PKI) key management:

In a *Symmetric Key Management* system the sender tells the key manager who is receiving the data to be encrypted and an encryption key is set. The receiver of the encrypted data then authenticates that the data is coming from a valid sender via the key manager who in turn sends the decryption key so as to enable the data to be decrypted. The same key is used to encrypt and decrypt the data. This means that such a key management system is fast from a performance perspective. However, in many cases, Symmetric Key Management systems have high storage costs as they require a database to store the generated keys. Additionally, the key manager must always be available as it plays a role in every encryption and decryption operation.

*PKI Key Management system*: In a PKI system, a public and private key are created simultaneously by a certification authority (CA). The private key is given only to the requesting party and the public key is made available as part of a digital certificate in a directory that all parties can access. The private key is never shared and cannot be accessed via the Internet. Thus, the sender accesses the public key from the central directory and uses it to encrypt the data. The receiver then authenticates that the sender is a valid one from the CA and then decrypts the data with their private key.

One advantage of PKI systems over Symmetric Key ones is that there is no requirement for a key server to be contacted for each message sent. However, key

recovery is difficult as the recipients generate the private keys themselves. In addition, a sender must locate a public key for every recipient and authenticate its validity. This is not always possible as the directory may not be able to supply public keys for all recipients.

Limitations with both key management systems are apparent when we consider the issue of mobility. In the case of Symmetric Key Management systems, a very large key management database would be required when mobile commerce transactions are factored in. We would also need to consider the fact that the transactions would exponentially increase the number of operations the key server needs to handle. In the case of PKI Key Management systems the complexity of its operation would seem to make its practical implementation for Mobile Commerce systems difficult especially given that the issue of key recovery would be exacerbated with mobile recipients.

Figure 2.4 illustrates the operation of an Identity Based Encryption (IBE) system. The encryption key is derived mathematically from the receiver's identity. Thus, when the sender specifies the identity of the receiver(s) an encryption key is derived. The data is then encrypted and sent to the receiver who authenticates the data with a key server. Once authenticated, the key server sends the decryption key to the receiver and the data can be decrypted. With IBE the sender does not need to contact the key server at all while the receiver only needs to contact the key server once to authenticate and receive the decryption key. There is no need for a key database as the server can construct the receiver's decryption key mathematically. Encrypting information is also straightforward as the sender can dictate which key server can be used to protect data. The location of the key server can be in the sender's or receiver's organization or indeed can be managed by a third party.

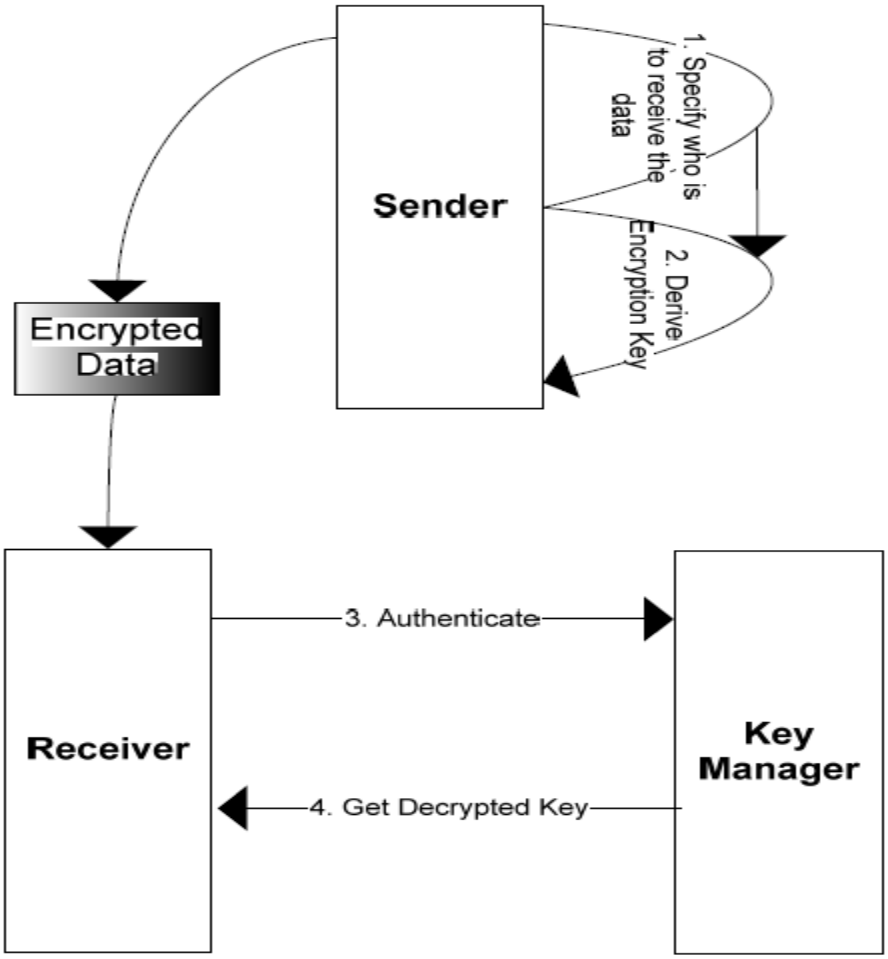


Figure 2.4: The operation of an IBE scheme

Voltage Corporation's White Paper on IBE [49] outlines how IBE meets the requirements for key management in an enterprise environment.

1. As the keys are derived mathematically from the recipient's identity, keys are always available for recipients.
2. Existing authentication resources such as directories or web authentication can be reused.
3. Partners can manage keys jointly as IBE facilities the selection of a local key server, a partner's key server or a service to protect the data.
4. The server can regenerate keys for different infrastructure components as needed.
5. As all keys are generated from a base secret stored at the key server any key can be securely regenerated and recovered as long as this base secret can be retrieved.
6. Since we don't need a database or a per–transaction connection to the key server additional applications and transactions are easy to add to the system.

The last point is important in relation to mobile computing applications given the exponential growth that can occur when a mobile commerce capability is introduced into an enterprise. The comparative simplicity of the system vis-à-vis symmetric or PKI systems is another benefit for mobile computing systems as querying a key server from a mobile device could potentially slow a transaction down significantly.

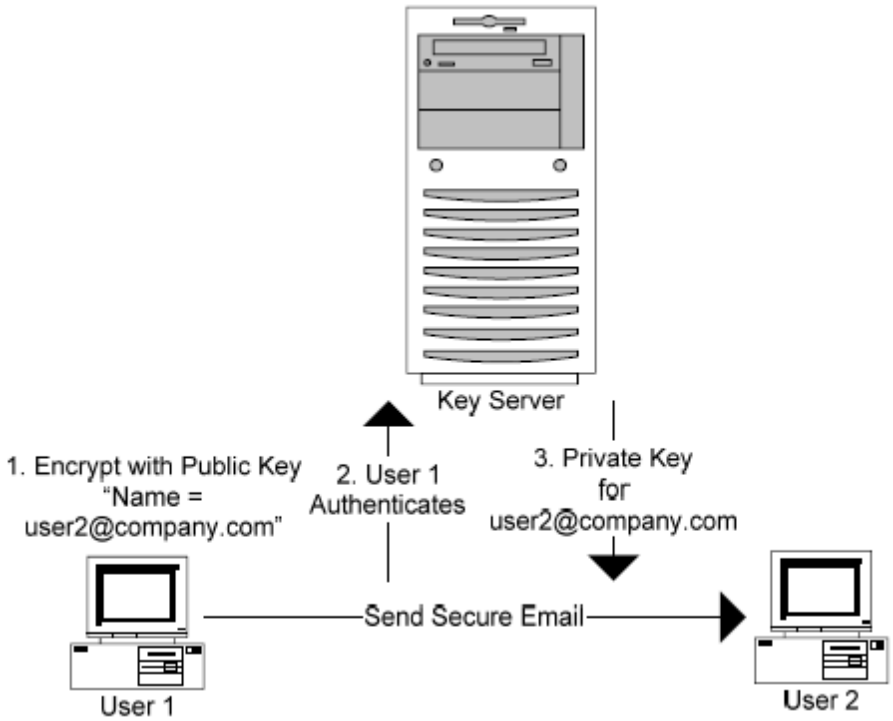


Figure 2.5: Sending an Email using IBE

Figure 2.5 illustrates how a secure email is sent using IBE. Assuming we have a sender User 1 who sends a secure email to a recipient User 2, the latter's email address being user2@company.com, the following steps take place:

1. User 1 encrypts the email using User 2's email address (user2@company.com) as the public key.
2. When User 2 receives the message he/she contacts the key server. The key server contacts a directory or other external authentication source to authenticate User 2's identity.
3. After authenticating User 2, the key server then returns his/her private key, with which User 2 can decrypt the message. This private key can be used to decrypt all future messages received by User 2.

Private keys only need to be generated once, upon initial receipt of an encrypted message. All subsequent communications corresponding to the same public key can be decrypted using the same private key, even if the user is offline. Also, because the public key is generated using only User 2's email address, User 2 does not need to have downloaded any software before User 1 can send him a secure message.

### 2.3.2. IBC Applications

Gagné [16] outlines several applications for IBE. These include:

- The previously discussed forward-secure encryption.
- The revocation of public keys whereby the current date can be included in the construction of the public key, thus providing a preset expiration date.

- The management of user credentials where the inclusion of a clearance level in the public key means that a receiver will only be able to decrypt the message if he/she has the appropriate clearance level.
- Delegations of decryption keys whereby management can give subordinates private keys corresponding to their responsibilities so that subordinates can only decrypt messages which fall within their responsibilities.

Voltage Corporation's White Paper on Email Security [47] outlines how IBE provides better performance than its symmetric and asymmetric key management counterparts. With the former, the need for a central server to manage each transaction means that the server gets busier the more email users are added to the system and there is no offline capability. There is a similar lack of offline capability with asymmetric key management systems. Moreover, the performance of asymmetric key management systems is affected by the difficulties that can be encountered in locating certificates and the administrative problems in validating these certificates. By contrast, messages can be encrypted and decrypted using IBE even when offline. Ad-hoc communication is also possible as no pre-enrolment of users is required. Penn and Sage [36] expand on these advantages to explore how IBE is easier to integrate into other products and how better key usage and management is facilitated.

IBE has other applications other than secure email. Voltage Corporation's White Paper on Secure Messaging for Financial Services [48] explains how financial services institutions can use IBE to exchange sensitive information without a plug-in or software download. Oliveira, Aranha, Morais, Daguano, Lopez and Dahab in [33] argue how IBE would seem to be the only practical means of providing security for Wireless Sensor Networks (WSNs). In the research carried out, an implementation of the Tate pairing, dubbed TinyTate, is introduced and the use of IBE to solve the key distribution problem in WSNs is proposed.

### 2.3.3. Identity Based Signature Schemes

The principles of Identity Based Signature Schemes (IBS) are the same as that of IBE. The private key is derived from the recipient's identity and the receiver obtains the key for verifying the signature from a key server. The operation is similar to other digital signature schemes such as Figure 2.6.

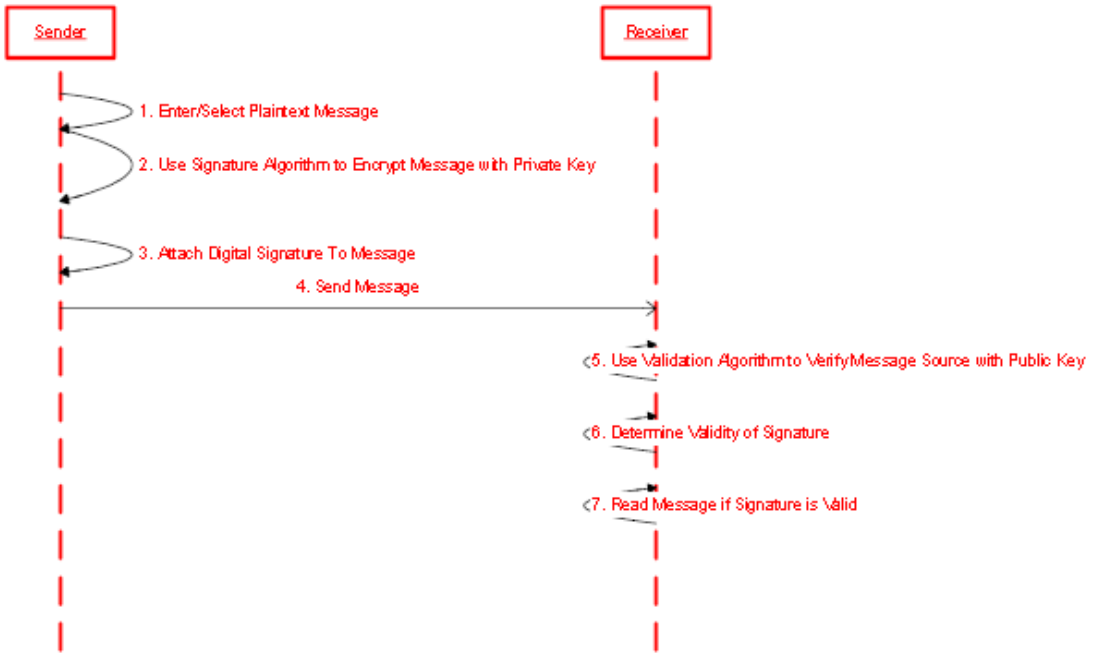


Figure 2.6: A Digital Signature Scheme

The operation of a typical digital signature scheme is illustrated here. The sender enters or selects a plaintext message a signature algorithm encrypts the message using the private key. The digital signature is attached to the message which then can be sent. On receipt of the message a validation algorithm is used to verify the signature. If the signature is valid the message can then be read.

IBE Schemes have existed for many years but often have lacked proofs as to their security. However, as Libert and Quisquater [23] point out this is gradually improving. And Bellare, Namprempre and Neven [2] provide proofs in the area of IBS. There are also several implementations of IBS. For example, Cha Choon and Hee Cheon [7] outline an IBS Scheme constructed using Gap Diffie-Hellman Groups.

### 2.3.4. IBC Security and Key-sizes

In [43] the subject of key-sizes for IBE is also discussed. Estimates of the required asymmetric key-sizes corresponding to symmetric key-sizes are extremely important information from a practitioner's point of view. Equivalent key-sizes for schemes based on the factoring assumption, the discrete-logarithm setting and the Elliptic Curve setting are published and updated regularly [30], [11]. As yet no significant equivalent key-size estimates are available for the pairing based setting. One of the reasons for this is that NIST/ECrypt traditionally focuses on schemes that are both mature and widely deployed and cryptography based on pairings is a relatively new area of research. It is also possible that the underlying complexity of the mathematics of pairings is a reason why significant studies have not as yet taken place in the wider community. The ECryptII Yearly Report on Algorithms and Key-sizes for 2009-1010 [11] specifically mentions about the non inclusion of equivalent key-sizes in the

pairing based setting that, "No such scheme is however included due to lack of deployment and/or maturity. In particular, recommendations for complete sets of parameters, giving certain 'symmetric key-size equivalence' is still somewhat difficult to make with sufficient assurance/confidence".

## 2.4. IBE Schemes

Shamir [40] formulated the concept of Identity Based Cryptography. However, constructing a practical IBE scheme remained an open problem for years. Boneh and Franklin [4], Cocks [8] and Sakai et al. [39] presented three different IBE solutions in 2001. The Cocks IBE scheme is based on the quadratic residuosity. Boneh-Franklin and Sakai et al. based their schemes on bilinear pairings on elliptic curves [41] and the security is based on the Bilinear Diffie-Hellman (BDH) assumption [4].

The Boneh-Franklin IBE scheme (BF-IBE) was the first to have a proof of security and has received the most attention due to this fact. In 2003, Sakai and Kasahara [38] presented a new IBE scheme, with a new key extraction algorithm. This scheme had a potential to improve performance. In 2005, Chen and Cheng [6] employed a simple version of the Sakai-Kasahara IBE scheme from [38] and the Fujisaki-Okamoto transformation [15] to present an efficient IBE scheme (SK-IBE). They also provided the security proof of SK-IBE.

SK-IBE is the scheme that is implemented in this Thesis. Information about the BF-IBE scheme is also provided, so as to have an efficient comparison of the two schemes.

### 2.4.1. Boneh-Franklin IBE

Boneh-Franklin IBE was the first practical and secure IBE scheme. As we can see in [26], BF-IBE belongs to the full-domain hash family IBE schemes. Their basic characteristic is that an identity is mapped to a point on an elliptic curve that is used for the encryption and the decryption algorithms. This kind of mapping requires a modular exponentiation, which is a fairly expensive operation. The calculation of a pairing is also required for the encryption and decryption algorithms, which accounts for a great amount of the needed computations.

The security of BF-IBE is based on the Bilinear Diffie-Hellman problem (BDHP). The original Boneh- Franklin paper [4] used the random oracle model to prove that an adversary able to decrypt a message that has been encrypted with Boneh-Franklin IBE can use his decryption algorithm to solve the BDHP, so if one believes that the BDHP is sufficiently difficult to solve, then Boneh-Franklin IBE must also be sufficiently

difficult to decrypt. The Boneh-Franklin scheme is resistant to chosen-plaintext attacks and adaptive chosen-identity attacks. With the use of the Fujisaki-Okamoto transformation it is also resistant to chosen-ciphertext attacks and adaptive chosen-identity attacks.

### 2.4.2. Sakai-Kasahara-IBE

As we can see in [26], SK-IBE belongs to the family of exponent inversion schemes. Here, a string representing an identity is hashed to an integer that is used in encryption and decryption operations. Such schemes are faster than full-domain hash schemes because modular exponentiation is avoided. The Sakai Kasahara IBE scheme presented here is actually quite different than that, originally described by Sakai and Kasahara. It has nevertheless kept this name because the key generation is motivated by their work.

### 2.4.3. Security of SK-IBE

In [4] and [3], we can see the definitions of the Bilinear Diffie Hellman problem (BDHP) and the q-Bilinear Diffie Hellman Inverse problem (q-BDHIP):

**BDHP**: Given group elements ($P_1$, $P_2$, $P_2^x$, $P_2^y$, $P_2^z$) for x, y, z $\in_R$ $Z_q$, compute $\hat{e}(P_1, P_2)^{xyz}$.

**q-BDHIP**: Given group elements ($P_1$, $P_2$, $P_2^x$, $P_2^{x2}$, …, $P_2^{xq}$) for x $\in_R$ $Z_q$, compute $\hat{e}(P_1, P_2)^{1/x}$.

As we can see in [26], an adversary observing a message that is encrypted with the Sakai-Kasahara IBE has access to P, $P_1 = \alpha P$, $P_3 = \gamma P$, and $g = \hat{e}(P, P)^{\alpha\beta}$ from the public parameters of the scheme. He also observes sP and $(sq_{ID})P_1sP_3 = (\alpha sq_{ID} + \gamma s)P = s(\alpha q_{ID} + \gamma)P$ from the ciphertext. From these values he wants to recover $sg = \hat{e}(P, P)^{\alpha\beta s}$. He can accomplish this in at least two ways. First, he can calculate s from sP by calculating a discrete logarithm sP in $G_1$, and then calculating sg with this result. He can also calculate β as the discrete logarithm of $g = (\hat{e}(P, P)^{\alpha})^{\beta}$ in GT and then calculate $sg = \hat{e}(\alpha P, sP)^{\beta} = \hat{e}(P, P)^{\alpha\beta s}$ with this value. So an adversary who can calculate discrete logarithms in either $G_1$ or $G_T$ can decrypt messages that are encrypted with the Sakai-Kasahara algorithm. The q powers that are assumed in the q-BDHIP are not directly available to an adversary who intercepts an encrypted message, but are required in the proof of selective identity security, with the value of q indicating how many other private keys an attacker has access to.

So, if we believe that the *q*-BDHIP is sufficiently difficult to solve then Sakai-Kasahara IBE must also be sufficiently difficult to decrypt. The Sakai-Kasahara scheme is resistant to chosen plaintext attacks and adaptive chosen-identity attacks, chosen-ciphertext attacks and adaptive chosen-identity attacks. [6] provides the security proof of SK-IBE.

### 2.4.4. Comparison between SK-IBE and BF-IBE

SK-IBE and BF-IBE have some common features. They also have some important differences. The security of BF-IBE is based on the BDH problem [4]. As shown in [6], BDH and 1-BDHI are polynomial time equivalent. It is obvious that the q-BDHI problem (when q > 1) is easier that the 1-BDHI problem, and therefore, is easier than the BDH problem as well. This certainly shows the disadvantage of current reduction for SK-IBE as compared with one for BF- IBE [4, 21].

However, the advantage of SK-IBE is that it has better performance than BF- IBE, particularly in encryption and private-key extraction. A comparison of their performances is available in Table 1. By taking a closer look between SK-IBE and BF-IBE, we can see that SK-IBE is faster than BF-IBE in two aspects. First, in the Encrypt algorithm of SK-IBE, no pairing computation is required because $\hat{e}(P_1, P_2)$ can be pre-computed. Second, in operation of mapping an identity to an element in $G_1$ or $G_2$, the *maptopoint* algorithm used by BF-IBE is not required. Instead of that, SK-IBE makes use of an ordinary hash-function, mapping the identity to an integer.

| Scheme | Pairings | | Multiplications | | Exponentiations | | Hashes | |
|---|---|---|---|---|---|---|---|---|
| | Encrypt | Decrypt | Encrypt | Decrypt | Encrypt | Decrypt | Encrypt | Decrypt |
| SK-IBE | 0 | 1 | 2[*1] | 1 | 1 | 0 | 4 | 3 |
| BF-IBE | 1 | 1 | 1 | 1 | 1 | 0 | 4[*2] | 3 |

**Table 2.3: Performance comparison between SK-IBE and BF-IBE**

*1 An extra multiplication required than BF-IBE is used to map an identifier to an element in $G_1$.
*2 BF-IBE requires the maptopoint operation to map an identifier to an element in $G_1$ (or $G_2$) which is slower than the hash function used in SK-IBE which maps an identifier to an element in $Z_q^*$.

# 3. Design and Analysis of Developed System

The developed system is the SK-IBE scheme [6]. The scheme is implemented in Java with two different kinds of bilinear pairings: the asymmetric pairing $\hat{e} : G_1 \times G_2 \rightarrow G_T$ and the symmetric one $\hat{e} : G_1 \times G_1 \rightarrow G_T$. Java was selected as a programming language because it is open source, widely spread (computers, mobile phones, etc) and contains libraries for elliptic curve cryptography, bilinear pairings and big integer computations. It can also be used for the implementation of web services and for providing a program interface.

## 3.1. SK-IBE Scheme

The SK-IBE is specified by four polynomial time algorithms:

**Setup**

In this step, the master public key (public parameters) and the master private key are created. Given a security parameter, the parameter generator follows the steps:

i. Generate three cyclic groups $G_1$, $G_2$ and $G_T$ of prime order $q$, an isomorphism $\psi$ from $G_2$ to $G_1$, and a bilinear pairing map $\hat{e} : G_1 \times G_2 \rightarrow G_T$. To do this we pick an elliptic curve $E/F_q$ with embedding degree $k$, and a prime $p$ such that $p \mid \#E(F_q)$ (where $E(F_q)$ is an elliptic curve group). The security parameter will define the size of the groups $G_1$, $G_2$ and $G_T$. Pick a random generator $P_2 \in G_2^*$ and set $P_1 = \psi(P_2)$.

ii. Compute the pairing $g = \hat{e}(P_1 , P_2)$.

iii. Pick a random $s \in Z_q^*$ and compute $P_{pub} = sP_1$.

iv. Pick four cryptographic hash functions $H_1 : \{0, 1\}^* \rightarrow Z_q^*$, $H_2 : G_T \rightarrow \{0, 1\}^n$ , $H_3 : \{0, 1\}^n \times \{0, 1\}^n \rightarrow Z_q^*$ and $H_4 : \{0, 1\}^n \rightarrow \{0, 1\}^n$ for some integer $n > 0$.

The message space is $M = \{0, 1\}^n$. The ciphertext space is $\mathbb{C} = G_1^* \times \{0, 1\}^n \times \{0, 1\}^n$. The master public key is $M_{pk} = (q, G_1, G_2, G_T, \psi, \hat{e}, n, P_1, P_2, P_{pub}, H_1, H_2, H_3, H_4, g)$, and the master secret key is $M_{sk} = s$.

**Extract**

Given an identifer string $ID_A \in \{0, 1\}^*$ of entity A, $M_{pk}$ and $M_{sk}$, the algorithm returns the entity's private key: $d_A = \frac{1}{s+H1(ID A)} P_2$.

**Encrypt**

Given a plaintext $m \in M$, $ID_A$ and $M_{pk}$, the following steps are performed:

i. Compute $q_{ID} = H_1(ID_A)$

ii. Select random $\sigma \in \{0, 1\}^n$
iii. Compute $r = H_3(\sigma, m)$
iv. Compute $Q_A = q_{ID}P_1 + P_{pub}$
v. Compute $U = rQ_A$
vi. Compute $V = \sigma \oplus H_2(g^r)$
vii. Compute $W = m \oplus H_4(\sigma)$
viii. Set the ciphertext to $C = (U, V, W)$

**Decrypt**

Given a ciphertext $C = (U, V, W) \in \mathbb{C}$, $ID_A$, $d_A$ and $M_{pk}$, perform the following steps:

i. Compute $q_{ID} = H_1(ID_A)$
ii. Compute $g' = \hat{e}(U, d_A)$
iii. Compute $\sigma' = V \oplus H_2(g')$
iv. Compute $m' = W \oplus H_4(\sigma')$
v. Compute $r' = H_3(\sigma', m')$
vi. Compute $U' = r'(q_{ID}P_1 + P_{pub})$
vii. If $U \neq U'$, output $\perp$, else return $m'$ as the plaintext

In the above scheme an asymmetric pairing is used, that is $\hat{e} : G_1 \times G_2 \rightarrow G_T$. It is also possible to use a symmetric pairing $\hat{e}' : G_1 \times G_1 \rightarrow G_T$. In this case $G_1 = G_2$ and $P_1, P_2 \in G_1$.

## 3.2. Tools and Libraries

The Integrated Development Environment used is Netbeans (version 6.9.1) and the Java Development Kit version is 1.6. Except from the JDK, Jpair [10] is used as an external library for the computation of bilinear pairings. It was the only available library, with documentation, found for pairing computations in Java. The implemented library for this Thesis is actually incorporated into Jpair.

### 3.2.1. Jpair

Jpair is a Java implementation of bilinear pairings. It has no dependencies on external libraries. An implementation of the Boneh-Franklin IBE scheme is also included, which should be useful for an efficient comparison between BF-IBE and SK-IBE. According to the author, it can be used on the Android platform, without changing any of the code.

Currently Jpair intentionally supports only curves over prime fields (characteristics $p > 3$ and $p = 3 \bmod 4$) and with embedding degree of 2. The Weierstrass equation of the curve is $y^2 = x^3 + ax + b$. More specifically, Jpair supports the supersingular curve

$y^2 = x^3 + x$ over the field Fp for some prime p = 3 mod 4. This is called Type A in the PBC [25] terminology. The advantage of this curve is that the number of the points on the curve is exactly p+1 and so the parameters of a random pairing can be easily generated.

Jpair also supports non-supersingular curves in the form of $y^2 = x^3 - 3x + b$ over the field Fp for some prime p = 3 mod 4. In this case a = -3, so the performance of this type of curves can be boosted using the more efficient algorithms proposed by Chatterjee et al. [5].

The Jpair classes that are used are the following:

**Package "uk.ac.ic.doc.jpair.api"**

This package contains the classes that provide an interface for the classes included in the "uk.ac.ic.doc.jpair.pairing" package.

Class *Field*: The interface for an abstract finite field.

Class *FieldElement*: The interface for a field element.

Class *Pairing*: The interface for a definition of an abstract pairing $\hat{e} : G_1 \times G_2 \rightarrow G_T$.

**Package "uk.ac.ic.doc.jpair.pairing"**

This package contains the classes that are needed for the Tate pairing computation and operations over elliptic curves and bilinear pairings.

Class *BigInt*: A wrapper class of the java.math.BigInteger class. It implements the FieldElement interface.

Class *ByteArrayUtil*: It performs operations with Byte Arrays.

Class *Complex*: This is an immutable class that defines complex numbers over a finite field, i.e. the real and imaginary parts of the complex number are taken over the finite field.

Class *ComplexField*: Here, the extension field $F_p^2$ is represented as a field of complex numbers over $F_p$.

Class *EllipticCurve*: This class defines elliptic curves over GF(p). The Weierstrass equation of the curves is: $Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$ or $Y^2 = X^3 + aX + b$.

Class *Fp*: This class defines finite fields of characteristic p, where p is a prime number. All the arithmetic operations are done modulo p.

Class *JacobPoint*: It returns or sets the coordinates of an elliptic curve point in the Jacobean system.

Class *PairingFactory*: Here, a Tate pairing is returned using the supersingular curve $y^2 = x^3 + x$ over a random field, given the security parameters (group size, field size, source of randomness). Group size is the bit length of the group $G_1$ and field size is the bit length of p which defines GF(p).

Class *Point*: It returns or sets the Affine coordinates of an elliptic curve point.

Class *TatePairing*: The Tate Pairing is implemented here. This implementation uses the pairing friendly curve $Y^2 = X^3 + aX + b$ defined over GF(p), where $p = 3 \bmod 4$. $G_1$ is taken as an order-q subgroup of the group formed by all the points on the curve. The curve has an embedding degree of 2. It has a corresponding twisted curve $Y^2 = X^3 + aX - b$. Points from the twisted curve are used in the computation as elements in $G_2$ to avoid operations in the extension field. The algorithm is taken from [35]. The parameters for computing the Tate Pairing are: the elliptic curve on which $G_1$ is defined, the group-order (order of $G_1$) and a co-Factor (a big integer such that co-Factor*group-order = #E, where #E is the number of points on the curve).

### 3.2.2. Classes from Internal Packages

The following classes are used in the developed scheme. The information was taken from [44].

*java.io.Serializable:* java.io provides for system input and output through data streams, serialization and the file system. Serializability of a class is enabled by the class implementing the java.io.Serializable interface.

*java.math.BigInteger:* Provides immutable arbitrary-precision integers and operations.

*java.util.Random:* An instance of this class is used to generate a stream of pseudorandom numbers.

*java.io.UnsupportedEncodingException:* Used for an exception when the Character Encoding is not supported.

*java.security.KeyPair:* This class is a simple holder for a key pair (a public key and a private key).

*java.security.PrivateKey:* A private key.

*java.security.PublicKey:* A public key.

*java.security.MessageDigest:* This MessageDigest class provides applications the functionality of a message digest algorithm, such as MD5 or SHA.

## 3.3. Development Strategy and Structure

As stated above, an implementation of the Boneh-Franklin IBE scheme is included in Jpair. The SK-IBE scheme algorithms were implemented in such a way as to ease the understanding of the similarities and differences between the two schemes. The implementation, under this consideration, also enables the efficient comparison of these schemes.

### 3.3.1. Structure

**The symmetric Tate Pairing**

In order to achieve the construction of the symmetric pairing two new classes and a method to an existing class (PairingFactory) were added. They were created by introducing slight variations to Jpair's Pairing and TatePairing classes and the ssTate method.

Class *SymPairing*: The interface for a definition of an abstract pairing $\hat{e} : G_1 \times G_1 \rightarrow G_T$.

Class *SymmetricTatePairing*: The symmetric Tate Pairing is implemented here. This implementation uses the pairing friendly curve $Y^2 = X^3 + aX + b$ defined over GF(p), where $p = 3 \bmod 4$. $G_1$ is taken as an order-q subgroup of the group formed by all the points on the curve. The curve has an embedding degree of 2. The parameters for computing the symmetric Tate Pairing are: the elliptic curve on which $G_1$ is defined, the group-order (order of $G_1$) and a co-Factor (a big integer such that co-Factor*group-order = #E, where #E is the number of points on the curve).

Method *symTate*: This method is added in the PairingFactory class. Given the security parameters (group-size, field-size, rnd), it returns a symmetric Tate pairing, constructed using the supersingular curve $y^2 = x^3 + x$ over a random field. Group-size is the bit-length of group $G_1$. Field-size is the bit-length of p which defines GF(p). Rnd is the source of randomness.

**SK-IBE**

Two packages were created in order to implement the SK-IBE scheme: uk.ac.ic.doc.jpair.sk.ibe and uk.ac.ic.doc.jpair.sk.ibe.key. They contain the following classes:

Class *SKCipher*: This class implements the Sakai-Kasahara Identity-Base Encryption (IBE) scheme. The scheme consists of five algorithms, in contrast to the original scheme that consists of four algorithms. The additional algorithm is "DecryptionSetup" which actually breaks the decryption step in two parts. The

thought behind this is that part of the decryption algorithm calculates values which remain steady for a given Master Key-pair. So it will only be needed when the Master Key-pair is changed. This way, there is a significant reduction in the time needed for the decryption.

i. Setup: generates global system parameters (a master public key and a master private key). Note that the bilinear map must be generated beforehand. This is done in order to keep the setup step close to Jpair's setup for the BF-IBE.
ii. Extract: uses the master key to generate the user's private key corresponding to an arbitrary string ID.
iii. Encrypt: encrypts messages using the public key ID. The messages that are encrypted are byte arrays. This means that any kind of information can be encrypted after it has been transformed into a byte array. The produced ciphertext is C = (U, V, W). As we can see in the description of the algorithm, $W = m \oplus H_4(\sigma)$. So, W is the byte array that contains the encrypted message and has the same size as the original message m. This happens because σ is hashed into the same bit-length as m.
iv. DecryptionSetup: This is an intermediate algorithm used to make the decryption faster. It returns the point $Ux = q_{ID}P_1 + P_{pub}$ (to be checked if equal with the point U of SKCText), Ux remains the same for a single Master Keypair.
v. Decrypt: decrypts messages using the corresponding private key. The difference from the original SK-IBE scheme is in step vi of the algorithm, where instead of $U' = r'(q_{ID}P_1 + P_{pub})$, we have $U' = r'Ux$.

Class *SKCText*: This class represents the ciphertext C = (U, V, W).

Class *Util*: This class is not public. It contains the necessary hash functions and the xor operation for two byte arrays. This is not a new class. It is the same as the Util class of the BF-IBE scheme of Jpair, without the hash-to-point operations. In the original SK-IBE scheme four hash functions are mentioned: $H_1 : \{0, 1\}^* \rightarrow Z_q^*$, $H_2 : G_T \rightarrow \{0, 1\}^n$, $H_3 : \{0, 1\}^n \times \{0, 1\}^n \rightarrow Z_q^*$ and $H_4 : \{0, 1\}^n \rightarrow \{0, 1\}^n$ for some integer n > 0. We can see that only two are actually needed: $H_1$ and $H_4$. H2 can be replaced by H4 if we convert the pairing $\in G_T$ to a byte array. H3 can be replaced with H1 by using the System.arraycopy function of Java to produce a single byte array ($\{0, 1\}^n || \{0, 1\}^n$). The hashing algorithm is SHA-512.

Class *SKMasterPublicKey*: This class represents the master public key.

Class *SKMasterPrivateKey*: This class represents the master private key.

Class *SKUserPrivateKey*: This class represents the user's private key.

**SK-IBE from the symmetric pairing**

Two packages were created in order to implement the SK-IBE scheme from the symmetric pairing: uk.ac.ic.doc.jpair.symsk.ibe and uk.ac.ic.doc.jpair.symsk.ibe.key. They contain the following classes, which are equivalent to those of SK-IBE and perform exactly the same operations, with the symmetric pairing instead of the asymmetric one:

Class *SymSKCipher*

Class *SymSKCText*

Class *Util*

Class *SymSKMasterPublicKey*

Class *SymSKMasterPrivateKey*

Class *SymSKUserPrivateKey*

### 3.3.2. Class Diagrams

Below we present the class diagrams of the implemented scheme. The classes in the diagrams are described above in 3.2.1 and 3.3.1. The exported UML diagrams were created by using a freeware UML generator specifically designed for Java that is called JUG (Java UML Generator). This program was downloaded from SourceForge.net [37] and the results for all the classes that were used are the following:

**Package uk.ac.ic.doc.jpair.api**

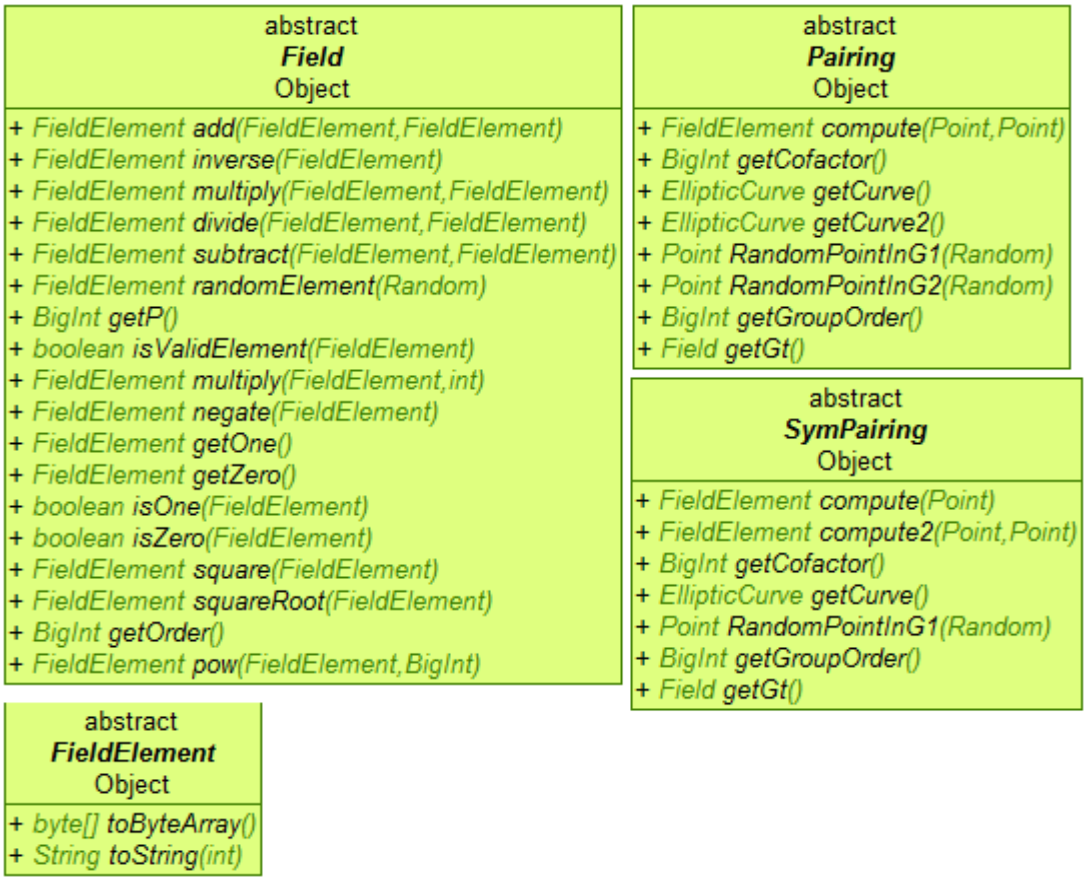


Figure 3.1: UML diagrams of Field, FieldElement, Pairing and SymPairing classes

## Package uk.ac.ic.doc.jpair.pairing


Figure 3.2: UML diagrams of BigInt and Complex classes

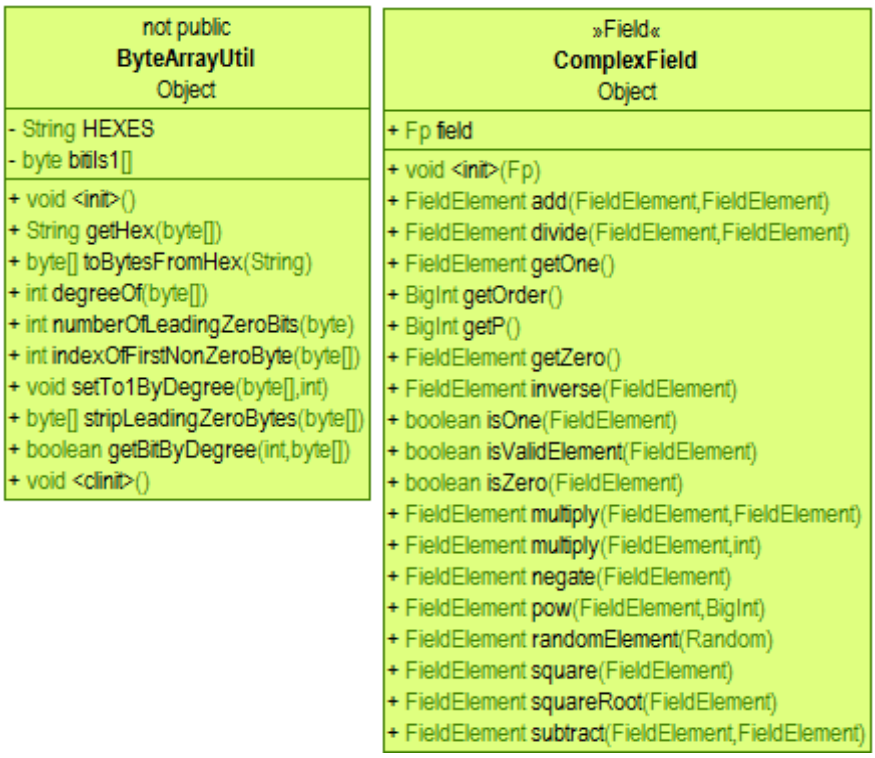


**Figure 3.3: UML diagrams of ByteArrayUtil and ComplexField classes**

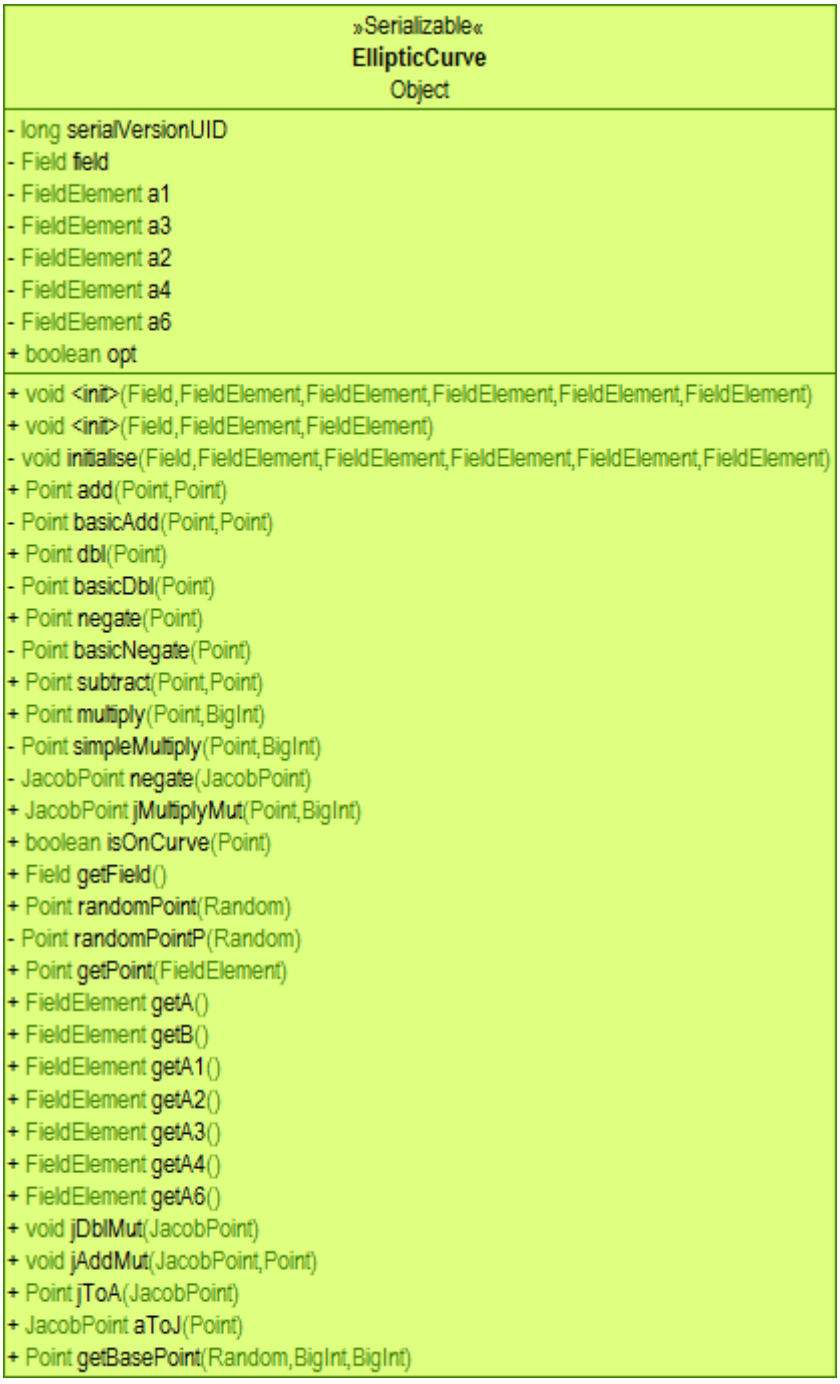


**Figure 3.4: UML diagram of EllipticCurve class**

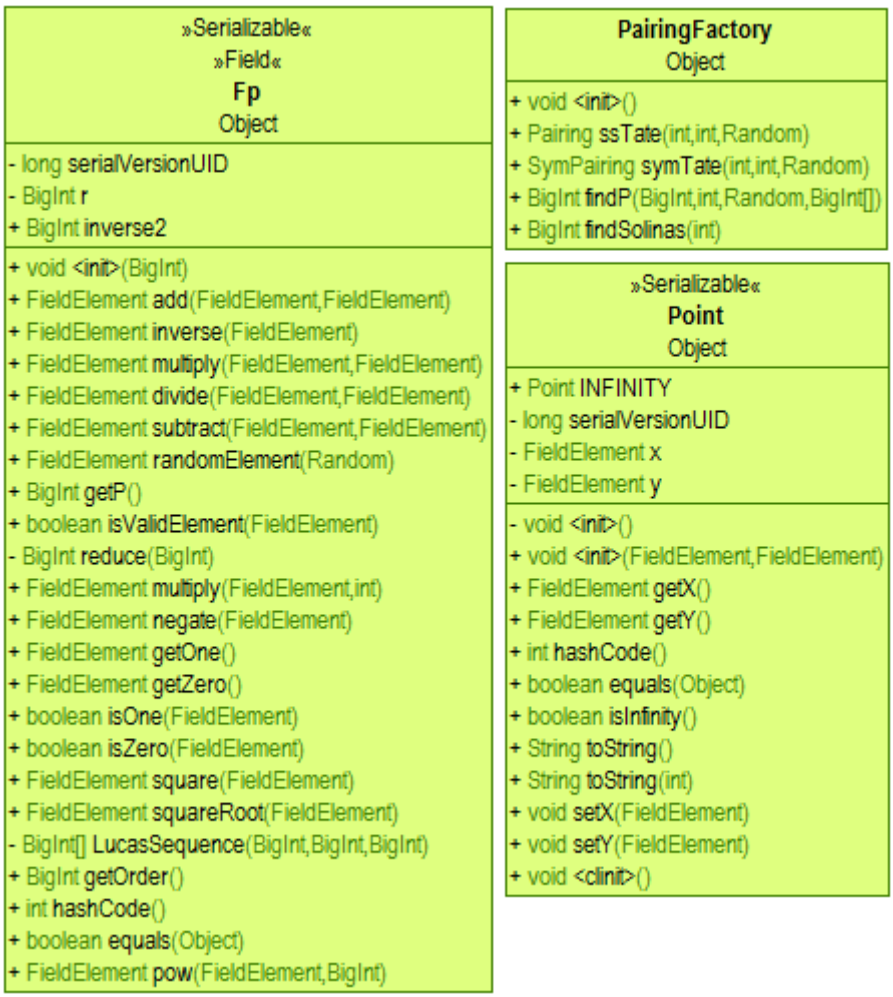


Figure 3.5: UML diagrams of Fp, PairingFactory and Point classes


Figure 3.6: UML diagrams of JacobPoint, SymmetricTatePairing and TatePairing classes

## Package uk.ac.ic.doc.jpair.sk.ibe

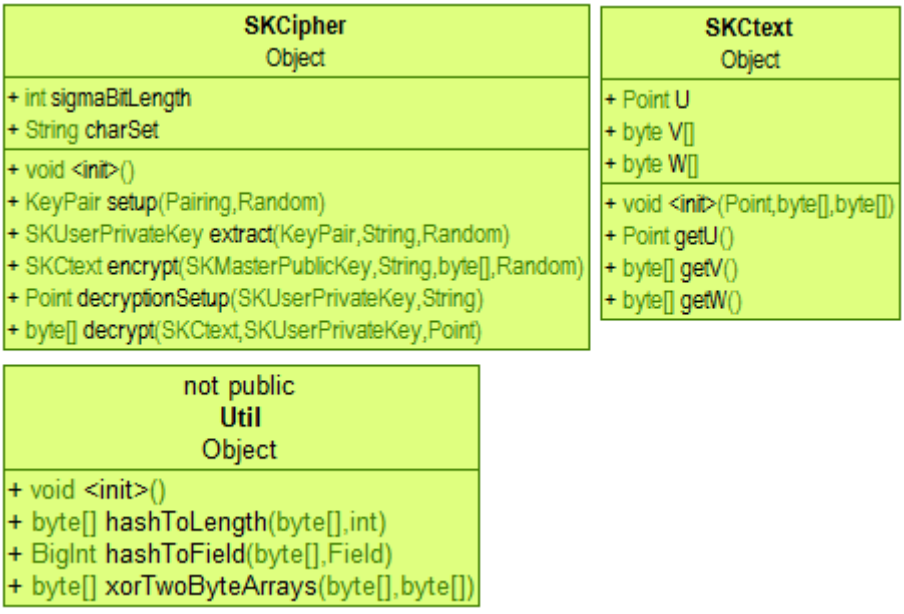


Figure 3.7: UML diagrams of SKCipher, SKCtext and Util classes

## Package uk.ac.ic.doc.jpair.sk.ibe.key

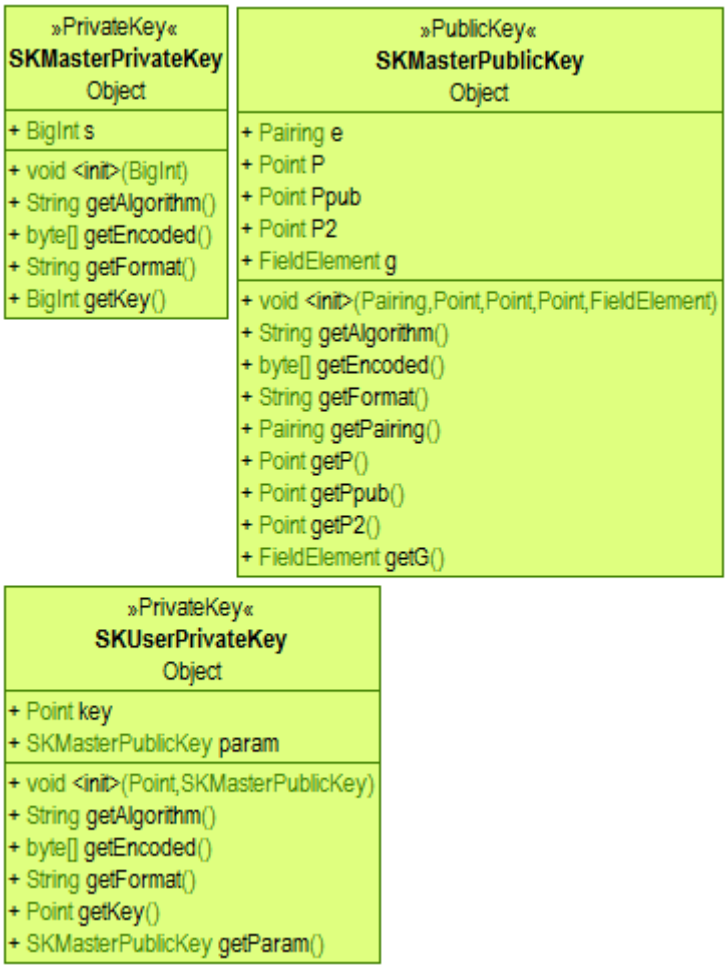


Figure 3.8: UML diagrams of SKMasterPrivateKey, SKMasterPublicKey and SKUserPrivateKey

## Package uk.ac.ic.doc.jpair.symsk.ibe

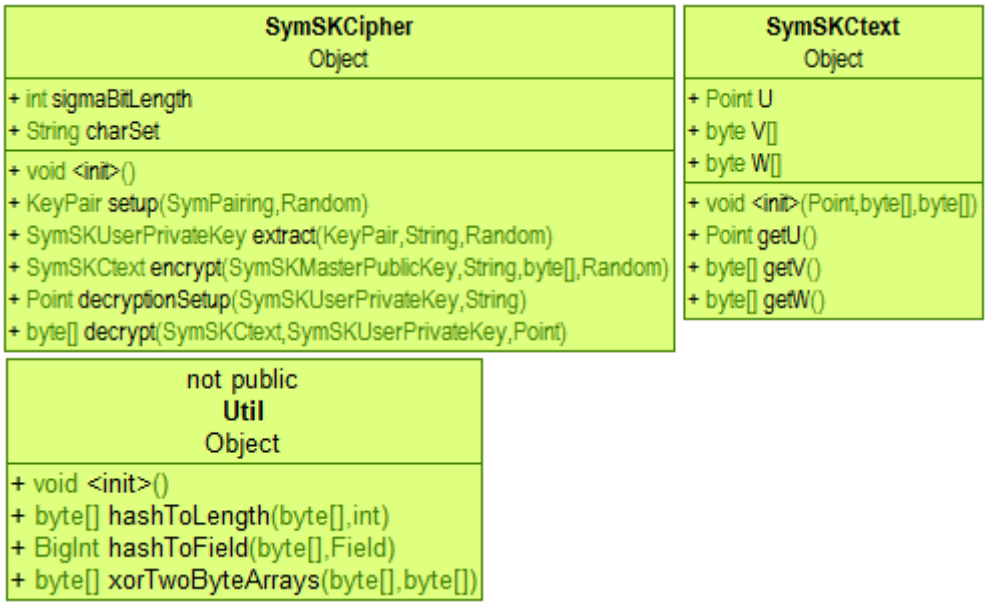


Figure 3.9: UML diagrams of SymSKCipher, SymSKCtext and Util classes

**Package uk.ac.ic.doc.jpair.symsk.ibe.key**

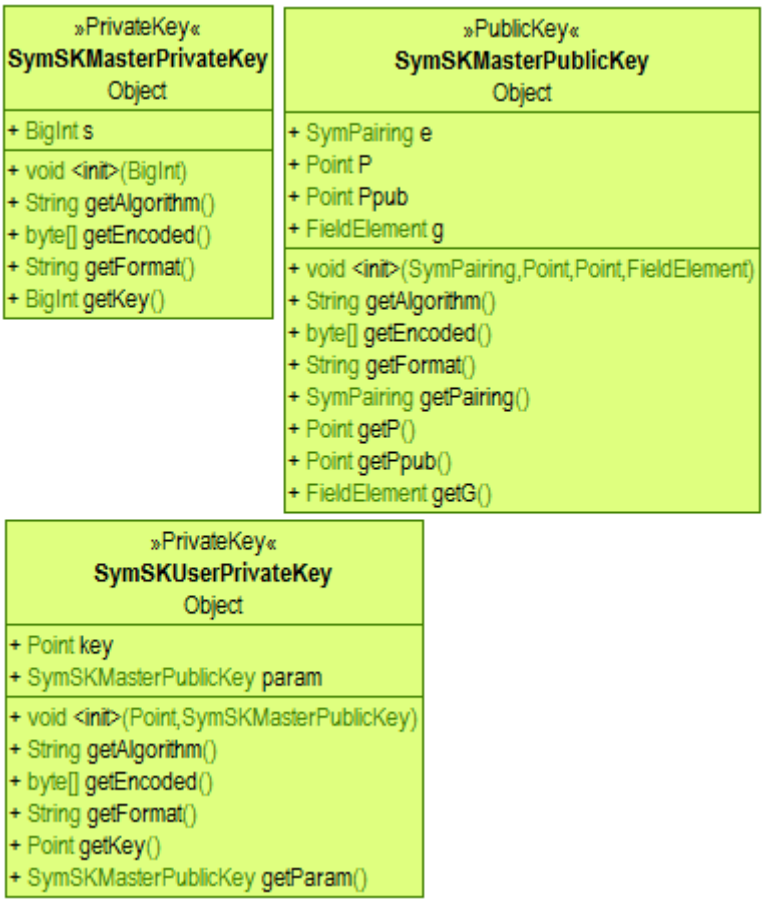


Figure 3.10: UML diagrams of SymSKMasterPrivateKey, SymSKMasterPublicKey and SymSKUserPrivateKey

## 3.4. Testing and Results

The implemented schemes were tested in order to get comparisons regarding the implementation time. The schemes compared are: BF-IBE (Jpair), SK-IBE, symSK-IBE (SK-IBE from the symmetric pairing). The equivalence of the IBE key-sizes to symmetric cryptography key-sizes are taken from [22] and [34], [19].

**Testing Objectives**

Execute the above IBE schemes for different group-order, finite-field and plaintext sizes.

Compare the execution times for the setup, extract, encrypt and decrypt algorithms.

**Environment**

*CPU*: Intel® Core™ 2 Duo CPU P7350 @ 2.00 GHz 2.00 GHz

*RAM*: 4,00 GB

*Operating System*: Windows 7 Professional 64-bit, Service Pack 1

*Java*: NetBeans IDE 6.9.1, JDK 1.6

*Other*: The computer was disconnected from the network. No other applications were running.

### 3.4.1. Results

The tables and graphs of the results are presented here. Note that in the setup phase, the calculation of the bilinear map is not included. The messages to be encrypted-decrypted are random big integers. The values of the time measurements are the average of 100 executions. The implementation time is in milliseconds.

| **Case 1** | group order 1024 bit - field size 2048 bit - message 83886080 bit (10 MB) | | |
|---|---|---|---|
| | **BF** | **SK** | **symSK** |
| **Setup** | 750 | 2778 | 2651 |
| **Extract** | 1492 | 646 | 643 |
| **Encrypt** | 3670 | 3246 | 3230 |
| **Decryption-Setup** | 0 | 1318 | 1315 |
| **Decrypt** | 2628 | 2621 | 2607 |

Table 3.1: Case 1, Security equivalent to 112 bit symmetric-key cryptography

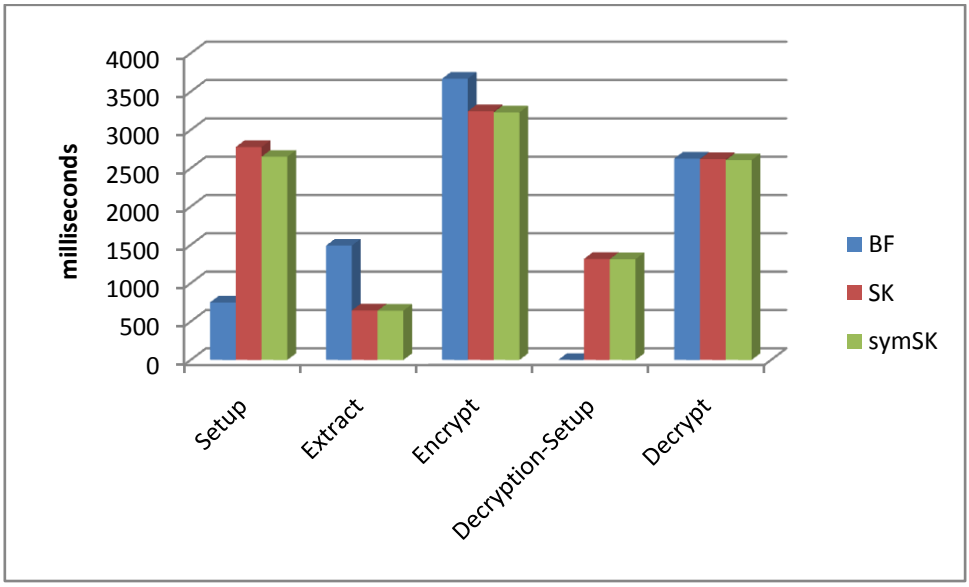


Figure 3.11: Depiction of Case 1

**Case 1:** Regarding the selection of the parameters, the group order is 1024 bits, the field size is 2048 bits and the message is 10 MB. For setup, the SK is 270,4% and the symSK is 253,5% slower than BF. For extract, the SK is 56,7% and the symSK is

56,9% faster than BF. For encrypt, the SK is 11,6% and the symSK is 12% faster than BF. For decrypt, the time needed is almost the same for all three schemes.

| **Case 2** | group order 1536 bit - field size 3072 bit - message 83886080 bit (10 MB) | | |
|---|---|---|---|
| | **BF** | **SK** | **symSK** |
| **Setup** | 2092 | 8152 | 7595 |
| **Extract** | 4011 | 1870 | 1835 |
| **Encrypt** | 9486 | 8532 | 8388 |
| **Decryption-Setup** | 0 | 3776 | 3707 |
| **Decrypt** | 6715 | 6750 | 6650 |

Table 3.2: Case 2, Security equivalent to 128 bit symmetric-key cryptography

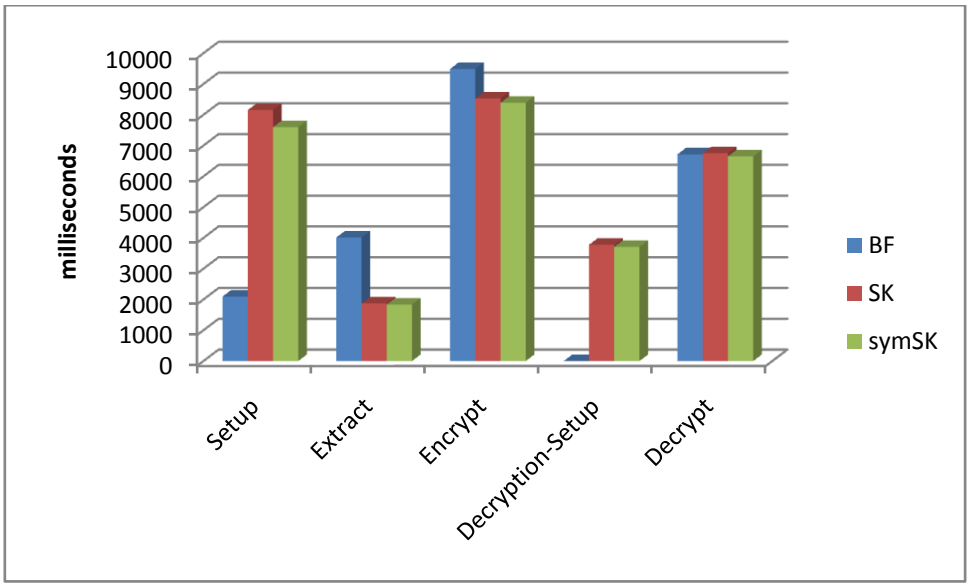


Figure 3.12: Depiction of Case 2

**Case 2:** The security provided in this case is equivalent to 128-bit symmetric-key cryptography. Regarding the selection of the parameters, the group order is 1536 bits, the field size is 3072 bits and the message is 10 MB. For setup, the SK is 289,7% and the symSK is 263% slower than BF. For extract, the SK is 53,4% and the symSK is 54,3% faster than BF. For encrypt, the SK is 10% and the symSK is 11,6% faster than BF. For decrypt, the time needed is almost the same for all three schemes.

| **Case 3** | group order 1536 bit - field size 3072 bit - message 838860800 bit (100 MB) | | |
|---|---|---|---|
| | **BF** | **SK** | **symSK** |
| **Setup** | 2338 | 8766 | 8503 |
| **Extract** | 4262 | 2032 | 2047 |
| **Encrypt** | 13996 | 13090 | 13141 |
| **Decryption-Setup** | 0 | 4138 | 4159 |
| **Decrypt** | 11373 | 11354 | 11382 |

Table 3.3: Case 3, Security equivalent to 128 bit symmetric-key cryptography (100 MB message)

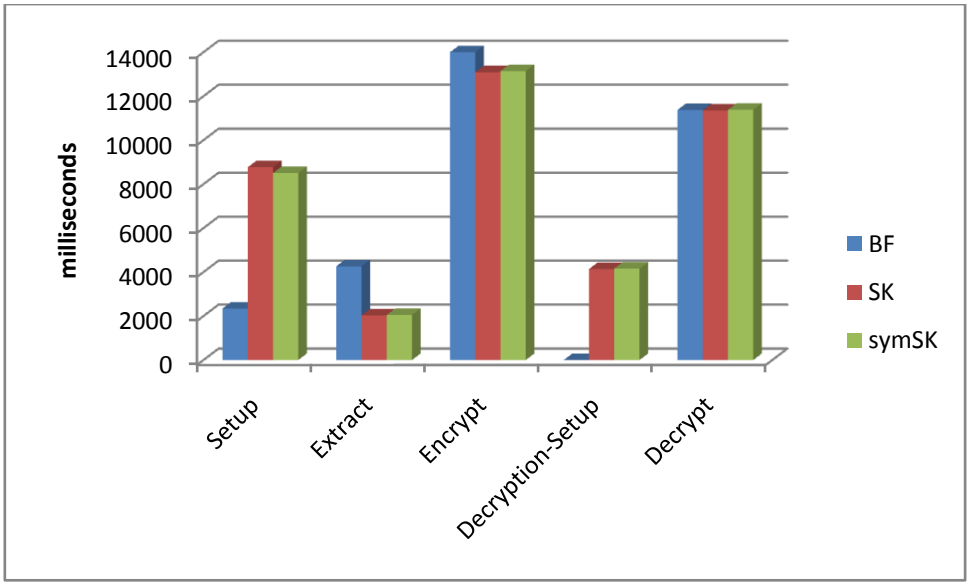


Figure 3.13: Depiction of Case 3

**Case 3:** The security provided in this case is equivalent to 128-bit symmetric-key cryptography. Regarding the selection of the parameters, the group order is 1536 bits, the field size is 3072 bits and the message is 100 MB. For setup, the SK is 274,9% and the symSK is 263,7% slower than BF. For extract, the SK is 52,3% and the symSK is 52% faster than BF. For encrypt, the SK is 6,5% and the symSK is 6,1% faster than BF. For decrypt, the time needed is almost the same for all three schemes.

| **Case 4** | SK - group order 1536 bit - field size 3072 bit | |
|---|---|---|
| | **10 MB** | **100 MB** |
| **Setup** | 8762 | 8766 |
| **Extract** | 2036 | 2032 |
| **Encrypt** | 9271 | 13090 |
| **Decryption-Setup** | 4095 | 4138 |
| **Decrypt** | 7296 | 11354 |

Table 3.4: Case 4, Security equivalent to 128 bit symmetric-key cryptography (Comparison for different message-sizes, SK scheme)

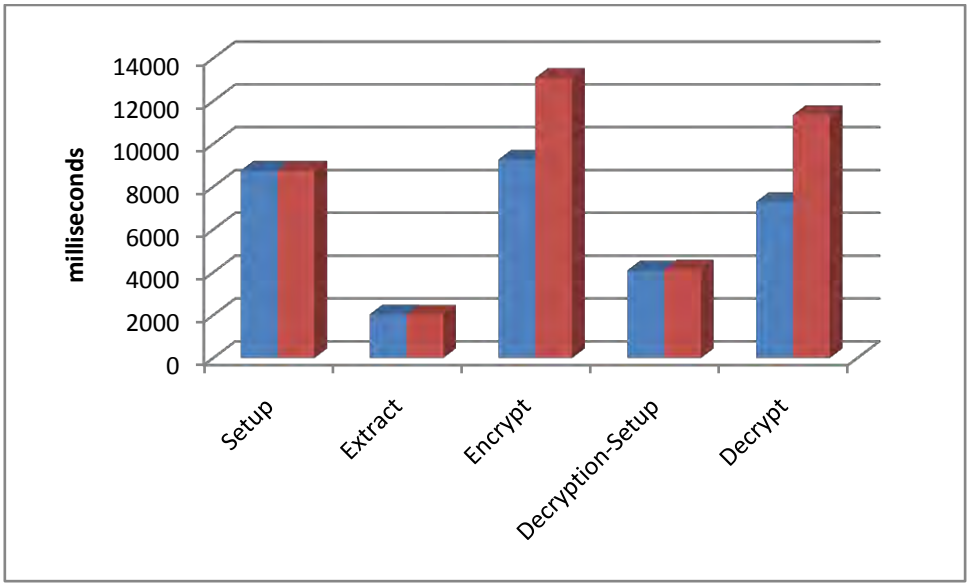


Figure 3.14: Depiction of Case 4

**Case 4:** In this case, only the SK scheme is tested. This test is about the time needed for different message-sizes. Regarding the selection of the parameters, the group order is 1536 bits, the field size is 3072 bits. The message size is 10 MB in one case and 100 MB in the other. The time needed for setup, extract and decryption-setup is almost the same, since it has nothing to do with the size of the message. The encryption of a 100 MB is 41,2% slower than that of a 10 MB message. The decryption of a 100 MB is 55,6% slower than that of a 10 MB message.

| **Case 5** | group order 3584 bit - field size 7168 bit - message 83886080 bit (10 MB) | | |
|---|---|---|---|
| | **BF** | **SK** | **symSK** |
| **Setup** | 26093 | 97851 | 91878 |
| **Extract** | 48453 | 22514 | 22423 |
| **Encrypt** | 109618 | 97866 | 97725 |
| **Decryption-Setup** | 0 | 44969 | 44918 |
| **Decrypt** | 76139 | 75948 | 75826 |

Table 3.5: Case 5, Security equivalent to 183 bit symmetric-key cryptography

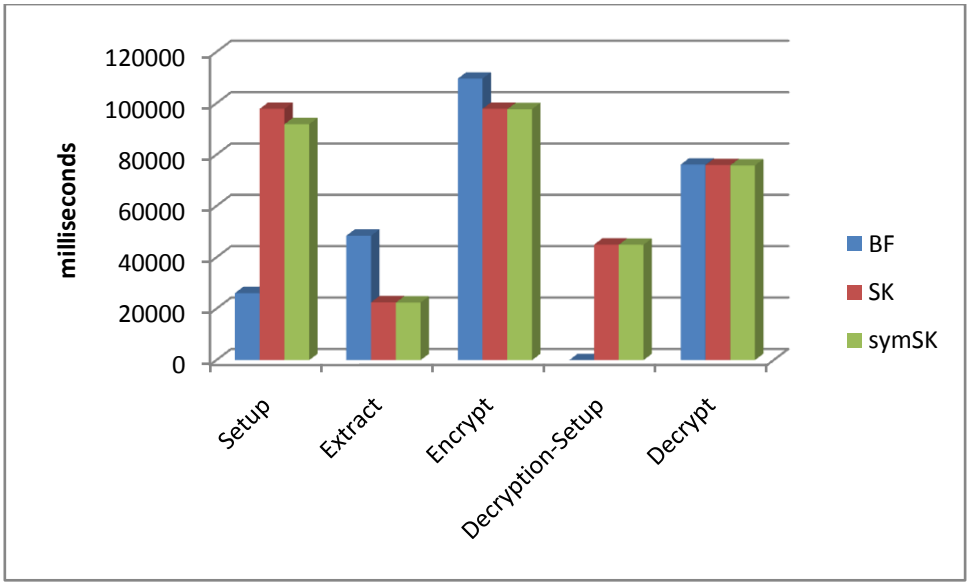


Figure 3.15: Depiction of Case 5

**Case 5:** Regarding the selection of the parameters, the group order is 3584 bits, the field size is 7168 bits and the message is 10 MB. For setup, the SK is 275% and the symSK is 252,1% slower than BF. For extract, the SK is 53,5% and the symSK is 53,7% faster than BF. For encrypt, the SK is 10,7% and the symSK is 10,8% faster than BF. For decrypt, the time needed is almost the same for all three schemes.

By examining the tables and graphs of the results, one can realize that the SK-IBE provides a slower setup, but a faster private-key extraction and encryption, while the decryption phase is at about the same levels. The slower setup is mainly due to the Jpair's functions required, given the fact that Jpair is mostly oriented for the implementation of the BF-IBE. The extraction and the encryption phases are faster because there is no need of mapping an identity onto an elliptic curve. Instead of this, the identity is hashed to produce a big integer.

In real world, a key-issuing authority would have to use the setup algorithm only a few times, or even once. On the other hand, the extract algorithm would have to be executed once for each user every time a new master public and private key would be issued. This can be translated into tens to thousands of executions. So, we can realize the importance of having low execution times for the extract algorithm. Let's consider for example that a machine, similar to that where the tests have taken place, had to issue 100 private keys for 128-bit security level. Depending on Case 2 of the tests, the BF-IBE would need 66,85 minutes, while the SK-IBE would need 31,17 minutes, that is a significant time difference. Moreover, a user that encrypts a message with SK-IBE would gain about one second in each encryption against one that encrypts with BF-IBE (8,5 sec vs. 9,5 sec).

Another interesting observation, regarding SK-IBE, is the breaking of the decryption algorithm of the original scheme into two parts: decryption-setup and decryption. Examining the results, one can see the benefit from this segregation. At 128-bit security level, a user gains 3,8 seconds at each decryption (Case 2), or else the

decryption becomes 35,87% faster. At 183-bit security level (Case 5) the gain is almost 45 seconds, or else the decryption becomes 37,19% faster. So, we can see that the introduction of the decryption-setup algorithm provides a great improvement in the implementation time, and the computational cost, of the decryption operation.

# 4. Conclusion

## 4.1. Aims of the Thesis

The goal of this Thesis was to provide a tool that could be used to provide IBC services. In order to achieve that, at first, relative literature had to be studied and discussed. The actual aim was to create a library for Identity-Based Cryptography that would be able to be used for key generation in Identity-Based systems. Another aim was to implement an IBE scheme both from the symmetric and the asymmetric pairing.

Java was a suitable choice as a programming language because it is open source, widely spread (computers, mobile phones, etc) and contains libraries for elliptic curve cryptography and big integer computations. It can also be used for the implementation of web services and for providing a program interface.

Another objective was the test and evaluation of the overall scheme according to several criteria, like time needed for the keys to be generated and the total execution time, depending on the key-size. The goal was to provide a scheme that is effective, providing the necessary security with the least possible resources (time, etc).

## 4.2. Evaluation

### 4.2.1. Literature Review

For the implementation of this Thesis there were various topics that needed to be studied, discussed and brought together. At first general cryptography topics [20, 45] were discussed. Then IBC issues were analyzed, from general notions [4,5,10], to IBE schemes [4, 8, 39, 6]. To provide a better understanding of these issues the Thesis also includes the necessary mathematic background, like elliptic curves [41, 50] and bilinear pairings [1, 17, 12]. Other important issues like Miller's algorithm [28, 29], suitable key-sizes [11, 22] and Identity Based Signature schemes [40, 23, 7] are also made clear.

### 4.2.2. Proposed System

The system developed is the Sakai-Kasahara IBE scheme. It is implemented as a Java library. It offers the ability to produce a Master public and private key, a user's

private key and the encrypt and decrypt operations. The scheme is implemented both from the symmetric and the asymmetric pairing.

The developed system is tested and evaluated according to several criteria, like time needed for the keys to be generated. The execution time, depending on the key-size, is tested. There are also tests regarding execution time for different sizes of the messages to be encrypted-decrypted. Last, for all the above evaluation points, there is a comparison between the SK-IBE scheme and BF-IBE scheme.

## 4.3. Recommendations for Future Research

This Thesis can be the basis for further research. First of all, the SK-IBE scheme implemented here is done with the use of supersingular curves. There are also other types of elliptic curves, or curves with different embedding degrees, that could be used for such an implementation. A comparison between different curves would produce useful results, regarding SK-IBE and IBE in general.

Second, the implementation is in Java. So, the scheme is possible to be transferred on the Android platform. Such a case would broaden the areas of use of the implemented IBC library and also provide information about its behavior in systems with lower hardware capabilities.

## 4.4. Conclusions

Identity Based Cryptography has received much attention in the recent years. Existing algorithms can be used to generate a cryptographic scheme from the beginning. There have been some implementations of IBE schemes, but most of them concern the Boneh-Franklin IBE and are written in C language. This Thesis has provided an implementation of the Sakai-Kasahara IBE scheme [6] in Java. Providing such a programming library is expected to simplify the implementation of IBC and at the same time be a motivation for it to be used and studied more widely.

The system produced is fully functional, and offers a faster private-key extraction and encryption than BF-IBE. The outcome of this thesis could be used for further academic usage and further studying of the development of Identity Based Cryptography. The use of Java may make the final scheme to be of use not only in desktop computers, but also in smart phones and other mobile computing devices, where the need of applying security measures is growing rapidly nowadays. The additional evaluation and tests can contribute to further understanding practical implementations of IBC.

# Bibliography


[1] Barreto P.S.L.M., Kim H. Y., Lynn B., and Scott M., Efficient algorithms for pairing-based cryptosystems. In CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology, pages 354–368, London, UK, 2002. Springer-Verlag.

[2] Bellare M., Namprempre C., Neven G., "Security Proofs for Identity-Based Identification and Signature Schemes", *Advances in Cryptography - Eurocrypt 2004*.

[3] Boneh D., Boyen X., Efficient selective-ID secure identity-based encryption without random oracles. In *Proceedings of Advances in Cryptology – Eurocrypt 2004*, LNCS 3027, pp. 223-238, Springer-Verlag, 2004.

[4] Boneh D., Franklin M., "Identity-Based Encryption from the Weil Pairing", *Proceedings of CRYPTO 2001*, LNCS 2139, pp. 213-229, Springer-Verlag, 2001.

[5] Chatterjee S., Sarkar P., Barua R., "Efficient Computation of Tate Pairing in Projective Coordinate over General Characteristic Fields". ICISC 2004: 168-181.

[6] Chen L., Cheng Z., Security proof of Sakai-Kasahara's identity-based encryption scheme. In *Proceedings of Cryptography and Coding 2005*, volume 3796 of LNCS, pages 442-459. Springer-Verlag, 2005.

[7] Choon J.C., Cheon J.H., "An Identity-Based Signature from Gap Diffie-Hellman Groups", *Desmedt, Y.G. (ed.) PKC 2003*, LNCS, vol. 2567, pp. 18–30. Springer, Heidelberg, 2002.

[8] Cocks C., An identity-based encryption scheme based on quadratic residues. In *Proceedings of Cryptography and Coding*, LNCS 2260, pp. 360-363, Springer-Verlag, 2001.

[9] Devegili A.J., Scott M., Dahab R., Implementing cryptographic pairings over barreto-naehrig curves. LNCS, 4575, pp. 197–207, 2007.

[10] Dong C., Jpair, http://jpair.sourceforge.net/

[11] ECRYPTII, "Yearly report on algorithms and keysizes", d.spa. 13 rev. 1.0, ict-2007e216676, 03/2010, 2010.

[12] Eisentrager K., Lauter K., Montgomery P.L., Fast elliptic curve arithmetic and improved weil pairing evaluation. In Topics in Cryptology ¨C CT-RSA 2003, pages 343–354. Springer-Verlag, 2003.

[13] Frey G., Muller M., Ruck H.G., "The tate pairing and the discrete logarithm applied to elliptic curve cryptosystems", *IEEE Transactions on Information Theory 1999*, 45(5), pp. 1717-9, 1999.

[14] Frey G., Rück H.G., A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. Math. Comput., 62(206), pp. 865–874, 1994.

[15] Fujisaki E., Okamoto T., Secure integration of asymmetric and symmetric encryption schemes. In *Proceedings of Advances in Cryptology - CRYPTO '99*, LNCS 1666, pp. 535-554, Springer-Verlag, 1999.

[16] Gagné M., "Identity Based Encryption: A Survey", RSA Laboratories Cryptobytes Volume 6, No.1, Spring 2003.

[17] Galbraithand S.D., Harrisonand K., Soldera D., Implementing the tate pairing. In ANTS-V: Proceedings of the 5th International Symposium on Algorithmic Number Theory, pages 324–337, London, UK, 2002. Springer-Verlag.

[18] Giry D., "Cryptographic Key Length Recommendation", BlueKrypt, 2011, http://www.keylength.com/en/4/

[19] Giry D., "Cryptographic Key Length Recommendation", BlueKrypt, 2011, http://www.keylength.com/en/7/

[20] Kahn D., *The Codebreakers*, Macmillan Publishing Company, New York, 1967.

[21] Koblitz N., "Elliptic curve cryptosystems", *Mathematics of Computation*, vol. 48, pp. 203-9, 1987.

[22] Koblitz N., Menezes A., Pairing-Based Cryptography at High Security Levels, in *Proceedings of Cryptography and Coding: 10th IMA International Conference*. Lecture Notes in Computer Science, vol. 3796 (Springer, Berlin, 2005), pp. 13–36.

[23] Libert B., Quisquater J.J., "The Exact Security of an Identity Based Signature and its Applications", *Cryptology ePrint Archive*, Report 2004/102, 2004.

[24] Lynn B., On the Implementation of Pairing-Based Cryptosystems, PhD thesis, Department of Computer Science, Stanford University, 2007.

[25] Lynn B., PBC Library, http://crypto.stanford.edu/pbc/

[26] Martin L., *Introduction to Identity Based Encryption*, Artech House Publishers, 1 edition January 2008.

[27] Menezes A., Vanstone S., Okamoto T., Reducing elliptic curve logarithms to logarithms in a finite field. In STOC '91: Proceedings of the twenty-third annual ACM symposium on Theory of computing, pages 80–89, New York, NY, USA, 1991. ACM.

[28] Miller V. S., The Weil Pairing, and its Efficient Calculation. Journal of Cryptology, 17, 2004.

[29] Miller V., "Use of elliptic curves in cryptography", In: Williams Hugh C, editor. *CRYPTO, Lecture notes in computer science*, vol. 218. Springer, pp. 417-26, 1985.

[30] NIST, "Recommendation for key management", special publication 800-57 part 1, 03/2007, 2007.

[31] Okamoto T., "Cryptography based on bilinear maps", In: Fossorier Marc PC, Imai Hideki, Lin Shu, Poli Alain, editors, *AAECC, Lecture notes in computer science*, vol. 3857, Springer, pp. 35-50, 2006.

[32] Okamoto T., "On pairing-based cryptosystems", In: Nguyen Phong Q, editor. *VIETCRYPT. Lecture notes in computer science*, vol. 4341, Springer, pp. 50-66, 2006.

[33] Oliveira L. B., Aranha D., Morais E., Daguano F., Lopez J., Dahab R., "TinyTate: Identity-Based Encryption for Sensor Networks", *Cryptology ePrint Archive*, Report 2007/020, 2007.

[34] Orman H., Purple Streak Dev., Hoffman P., Rfc 3766, Determining Strengths For Public Keys Used For Exchanging Symmetric Keys, VPN Consortium, April 2004.

[35] Park Eds. C., Chee S., "Efficient Computation of Tate Pairings in Projective Coordinates over General Characteristic Fields",Proc. 7th Int. Conference on Inf Security and Cryptology (ICISC 2004), LNCS 3506, Springer 2005, 168-181.

[36] Penn J., Sage A., "Voltage Security Tries To Put The Spark Back Into Secure Email", Forrester Research, Inc., 2004.

[37] Pschernig Elias, JUG Release 1.6 *alpha, http://sourceforge.net/projects/jug/

[38] Sakai R., Kasahara M.. ID based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054.

[39] Sakai R., Ohgishi K., Kasahara M., Cryptosystems based on pairing over elliptic curve (in Japanese). *The 2001 Symposium on Cryptography and Information Security*, Oiso, Japan, January 2001.

[40] Shamir A., Identity-based cryptosystems and signature schemes. In *Proceedings of Advances in Cryptology - Crypto '84*, LNCS 196, pp.47-53, Springer-Verlag, 1985.

[41] Silverman J. H., The arithmetic of elliptic curves, Springer GTM 106, 1986.

[42] Solinas J., Generalized Mersenne numbers, Technical Report CORR 99-39, University of Waterloo, 1999, http://www.cacr.math.uwaterloo.ca/techreports/1999/corr99-39.pdf

[43] Srinivasan S., "Identity based encryption: Progress and challenges", *Information Security Technical Report*, 15 (1), pp. 33-40, 2010.

[44] Sun Microsystems, Inc., Java 2 Platform, Standard Edition, version 1.4.2, API Specification, http://docs.oracle.com/javase/1.4.2/docs/api/overview-summary.html

[45] Van Oorschot P.C., Menezes A.J., Vanstone S.A., "Handbook of Applied Cryptography," CRC Press, Inc., 1997.

[46] Vertoda, "An Overview of Identity Based Encryption", Sykoinia Limited, 2009.

[47] Voltage Security, Inc., "Email Security – The IBE Architectural Advantage", 2006, http://www.voltage.com

[48] Voltage Security, Inc., "Secure Messaging for Financial Services: Conforming to GLBA Safeguards", 2006, http://www.voltage.com

[49] Voltage Security, Inc., "The Identity Based Encryption Advantage", http://www.voltage.com

[50] Washington L., Elliptic Curves: Number Theory and Cryptography. Chapman and Hall, CRC, 1st edition, 2003.