

Darmstadt University of Technology

Department of Informatics

# **Identity Based Public Key Infrastructures**

Bachelor Thesis  
by  
**TSVETAN PENEV**



supervised by Prof. Dr. Johannes Buchmann  
and Katja Schmidt-Samoa  
Date of submission: August 2005



# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>The Basics</b>	<b>4</b>
2.1	Diffie-Hellman Key Exchange . . . . .	4
2.2	Public key cryptography and trapdoor one-way functions . . . . .	5
2.3	RSA . . . . .	6
2.4	ElGamal . . . . .	7
2.5	Digital signatures . . . . .	8
<b>3</b>	<b>Public Key Infrastructure(PKI)</b>	<b>10</b>
3.1	What is PKI? . . . . .	10
3.2	The defects in PKI . . . . .	13
3.2.1	Authenticating the user . . . . .	14
3.2.2	Authenticating the CA . . . . .	14
3.2.3	Certificate Revocation Lists . . . . .	14
<b>4</b>	<b>Identity-based cryptography (IBC)</b>	<b>16</b>
4.1	Introduction to IBC . . . . .	16
4.2	Preliminaries . . . . .	17
4.3	Basic Concepts . . . . .	20
4.4	Identity-Based Encryption . . . . .	21
4.5	Chosen plaintext secure IBE . . . . .	25
4.6	ID-based Signatures from Pairings . . . . .	26
4.7	Conclusion . . . . .	28
<b>5</b>	<b>Combining the best of both worlds</b>	<b>29</b>
5.1	Introduction . . . . .	29
5.2	Preliminaries . . . . .	29
5.3	The CL-PKE scheme . . . . .	30
5.4	Security of the CL-PKC scheme . . . . .	31
<b>6</b>	<b>Conclusions and open problems</b>	<b>34</b>

# Chapter 1

## Introduction

The need for secret communication dates back since mankind ever existed. Most of us have played games that required teams to exchange encrypted messages, have sent messages to their beloved such that eavesdroppers can't understand them, even major war campaigns depended on a single encrypted message.

For many years, cryptographers believed that there is no way for two parties to exchange encrypted messages without exchanging some secret key before the communication begins. The classic and most widely used schemes in the past for encryption and decryption were the symmetric ones or also called private-key algorithms. If Bob and Alice wanted to take advantage of some symmetric algorithm they still had to negotiate and exchange some key before the communication begins. That key was used for encryption as well as for decryption and so it had to be kept secret. Later on however, as the number of people who wanted to communicate securely drastically increased, the need for a new kind of encryption methods was knocking on the door.

In the year 1977, a group of well known mathematicians today, Adi Shamir, Ron Rivest and Len Adleman introduced the idea of public key cryptography. They presented the RSA - cryptosystem which has been the most widely used public key cryptosystem until now. In such an asymmetric cryptosystem there are two distinguished keys. One key  $d$ , known as decryption key or also private key and another key  $e$ , known as encryption key or public key. The idea is that no one knowing the public key can make out what the private key is in some realistic time. If two parties want to communicate they first have to exchange their public keys and then encrypt messages with the other party's public key. The authenticity of the public key, however, is very important, since it guarantees that we are encrypting a message that can only be decrypted by the owner of the corresponding private key. If an intruder were able to change Bob's public key, then the attacker would be the only one who can decrypt the messages which were originally meant for Bob. For instance, in the real world the RSA public key is a pair  $(n, e)$  of natural numbers and the private key is a single natural number  $d$ .

Nowadays the importance of cryptography has grown even more with the ever growing world of Internet users. Security of data has become indispensable to today's global information society and public key cryptography, a key element in securing Internet communication. One can judge for oneself for the complexity of the matter, bearing in mind that according to latest research,

there are about  $9 * 10^8$  Internet users in 2005. The Public Key Infrastructure (PKI) relies on certificates issued by some trustcenter to users which ensure the connection between public key and identity. These certificates allow users to authenticate themselves and facilitate signing and encrypting of secure communication. However, in such a system if we want to send a message to Bob we have to check Bob's certificate, who issued it to Bob, is it still valid and only then if the information we receive is satisfactory we can send him a message. Following this same pattern before sending a message and keeping up to date information for certificates is very crucial in a PKI. Knowing that users enter and leave the system at any point in time makes the maintaining of such an infrastructure very difficult. The need for certificate issuance before secure communication can take place and management of these certificates has made this infrastructure too complex and the average user is reluctant to use it, despite all the benefits it can bring.

Probably the most common approach for data encryption nowadays is the one that uses hybrid encryption. It makes use of the fact that symmetric encryption is considerably faster than the asymmetric one and encrypts messages with a symmetric session key. Then it encrypts the session key using asymmetric cryptography which happens to be still fast because the session key is much smaller than the data file. However, the hybrid approach deals with only one problem. Certificates are still a vital component that cannot be removed nor replaced and the enormous infrastructure remains untouched. The ever growing number of Internet users who want to exchange data securely but still faster and with less preparation, has made it clear that a new approach was needed.

A solution for those needs is probably the so called Identity-Based Encryption (IBE). The revolutionary idea behind IBE is to make one's identity serve as the public key. First of all that makes Certificate Revocation Lists (CRL) obsolete because we don't need to look up the certificate of the user we are sending the message to, since we already know the encryption key we are going to use. Second it enables deployment of secure messaging between end users without the costly pre-enrolment strategy in PKIs. In the last couple of years Identity-Based Encryption has gained some significant popularity in the commercial and private sector. People are overwhelmed by the fact that they don't have to enrol for any kind of certificate and that the need to check some CRLs before initiating a secure communication is history. Even more, like in some well known commercials on the TV which promote the different functionalities of their products, I will say: That's not all. Despite the elegant way in which IBE deals with Certificate Revocation Lists and pre-enrolment for certificates, it combines very well with existing PKIs and it's considered to be of significant value.

Many companies have invested a considerable amount of money in their infrastructure, which is most probably working well. Now it would have been understandable if they recognized the advantages of IBE but still not use it, if only IBE was not compatible with the existing PKI. The bottom line is that when combining existing PKIs with IBE, companies can still make use of their investments in the PKI, which is considered best for authentication purposes, and as well offer new functionality offered by IBE. By doing so users can establish secure communication not only to users with certificates but to one's who may very well have no idea of what certificates in a PKI are used for.

# Chapter 2

## The Basics

In this chapter I am going to make a quick retrospection to explain how cryptography evolved over the years. By doing so I am going to explain some of the basic mathematical problems and notions in cryptography that are still relevant today and lie in the foundations of new trends in contemporary cryptography.

### 2.1 Diffie-Hellman Key Exchange

Two of the pioneers in public key cryptography, Whitfield Diffie and Martin Hellman, presented a public key distribution system that had a great influence in the cryptographic world ever since [1]. They recognized that speed is also sometimes a vital factor in transmitting encrypted messages and that many pairs of users won't simply wait for some secret key be transmitted over a secure physical channel. And that is only one part of the problem in symmetric encryption, because if  $n$  people want to exchange encrypted messages, each pair of users has to negotiate for a secret key beforehand. In total that makes  $\frac{n(n-1)}{2}$  keys that have to be exchanged prior to begin of secure communication, whereas in public key cryptography this is reduced to  $n$ . The idea by public key cryptography is that there are two distinguished keys, the public key and the private key, where each user has only one public key. Unlike private key cryptography there is no need for the public key to be kept secret, because it is assumed that no one can make out what the private key is just by knowing the public key. Of course, the public key has to be protected against forgery and thus kept authentic.

Diffie and Hellman had an idea at the time that relies on the difficulty of computing logarithms over a finite field  $GF(q)$ . Basically the mathematical problem looks like this:

You have a number  $a$  that is a fixed primitive element of  $GF(q)$  and

$$y = a^x \pmod q \text{ for some } 1 \leq x \leq q - 1$$

The idea is that anyone can easily compute  $y$  knowing what the secret exponent  $x$ , the base  $a$  and  $q$  is but it is very difficult to compute  $x$  from  $y$  and  $q$ . Anyone who is capable of computing discrete logarithms *mod*  $q$  is also capable of breaking the Diffie-Hellman key exchange algorithm. Here is how the algorithm works:

Let's say user A wants to communicate securely with user B but they have only a public channel at their disposal. Here is how they should proceed using the Diffie-Hellman key exchange. They negotiate for a prime number  $q$  and a generator for the cyclic group mod  $q$ . Once they have done that each one of them chooses a number  $X_i \in \{1, 2, \dots, q - 2\}$  and computes  $Y_i = a^{X_i} \bmod q$ . Then they exchange the so computed numbers  $Y_i$  but keep the exponents secret. Let's assume that user A has posted the number  $Y_1$  and user B the number  $Y_2$ . The next step is to negotiate a common key that no one except A and B know, using only the public channel and the information received from the other user. Here comes the idea from Diffie and Hellman into play. At that stage, after the exchange of the  $Y_i$ 's, user A should know  $Y_1, X_1$  and  $Y_2$ , user B on the other hand should know  $Y_2, X_2$  and  $Y_1$ . Then the common key is computed like this:

$$K_{1,2} \equiv a^{X_1 X_2} \bmod q$$

Why is that possible? Aren't the exponents kept secret? Yes, there is no problem, since  $Y_i = a^{X_i} \bmod q$  and then the common key is given by  $K_{i,j} \equiv Y_i^{X_j} \bmod q$ . That is computationally feasible for each of the users because they know their own secret exponents and the  $Y_i$  from the other user. For instance, in the situation above user A can get the common key by computing  $K_{1,2} \equiv Y_2^{X_1} \bmod q$ .

The security of this approach relies on the fact that the discrete logarithm problem is difficult to solve and anyone who is capable of computing discrete logarithms is in the position to break the system. For example, if Alice is such a person and is eavesdropping the public channel between A and B, she can gather all the information necessary to compute the common key  $K_{1,2}$ . All she has to do is calculate  $K_{i,j} \equiv Y_i^{\log_a Y_j} \bmod q$  [1].

## 2.2 Public key cryptography and trapdoor one-way functions

A *one-way function* is a function that is not invertible from a computational point of view. For any argument  $x$  in the domain of the function it should be easy to compute its image and still computationally impossible for any randomly chosen  $f(x)$  to invert the function and compute  $x$ .

Now a *trapdoor one-way function* is a function for which unlike one-way functions a simply computed inverse exists. However, it should be easily computable only for authorized personnel, which in this case means people who possess some specific trapdoor information gathered in the generation process of the trapdoor one-way function.

Diffie and Hellman recognized that business was quickly evolving and a new digital form was needed that can replace the written signature. That technique should provide anyone with the capability to check the authenticity of any digital signature and yet make it impossible for anyone else to produce a valid signature except the legitimate signer. They called that technique a *one-way authentication*. Probably the most influential idea they had was to use a public key cryptosystem to produce one-way authentication.

Let's say Alice wants to send Bob a message  $\mathcal{M}$  that Bob can later authenticate it as produced from Alice. Alice can make use of her secret deciphering key

$\mathcal{D}$  to "decrypt" the message and send it to Bob. Bob on the other side knowing the public encryption key  $\mathcal{E}$  can "encrypt" the message. If the "encryption" is successful, Bob can be sure that the message was written by Alice, since she is the only one who knows her secret key and the only one who could have produced that message with this property [1].

Diffie and Hellman described also an idea how [... a trap door cryptosystem can be used to produce a public key distribution system]. The trick is that a designer of a trapdoor one-way function is in the position to easily invert the function and make out what the secret key was. If say Alice and Bob want to set up a common private key they can proceed like this. First Alice chooses some random key and sends an encrypted message to Bob. Now Bob is the one who have produced the trapdoor and made it public but kept the trapdoor information secret, so when he receives the encrypted message from Alice, he can find out what Alice's secret key was. As a result of that, they have a common secret key [2]. Unfortunately, Diffie and Hellman couldn't give a concrete example of a trapdoor one-way function at the time.

## 2.3 RSA

About the same time when Diffie and Hellman proposed their highly influential ideas a group of well known mathematicians Ron Rivest, Adi Shamir and Len Adleman, laid the foundations of today's Public Key Infrastructures by discovering the RSA algorithm. That's how the asymmetric cryptosystems came on the scene. In an asymmetric cryptosystem there are two distinguished keys. One of them is called the public key  $e$ , which is open to the public, anyone can view the public key but no one should be able to alter it. The other key is the private key  $d$  or decryption key and it should be known only to the person who wants to receive secret messages encrypted with the corresponding public key. Such a cryptosystem relies on the fact that it is impossible to guess what the private key  $d$  is, knowing only the public key  $e$ .

The RSA algorithm resides on the fact that factoring relatively big numbers in their prime factors is very difficult and takes an enormous amount of time (sometimes speaking in years) to accomplish. However, no one knows if this particular problem will continue to be difficult enough, so that it can be implemented in cryptography. It may very well be that in the next couple of years a machine is built or an algorithm is invented that can factor numbers in their prime factors without much problem. So cryptographers have to make the underlying algorithms easily exchangeable, because if tomorrow RSA is no longer secure it will be necessary to substitute it with some other still secure algorithm. Here is a short introduction to the main calculations that take place while using the RSA algorithm:

*In the calculations below by  $\varphi$  is meant the Euler function. The function is defined as:  $\mathbb{N} \rightarrow \mathbb{N}, m \rightarrow \varphi(m)$  The number  $\varphi(m)$  gives us how many numbers  $a \in \{1, 2, \dots, m\}$  there are that satisfy  $\gcd(a, m) = 1$ .  $\gcd(a, b)$  is the greatest common divisor of the two numbers  $a$  and  $b$ .*

1. We find two random, relatively big prime numbers  $p$  and  $q$ . 2. We calculate their product  $n = pq$ . 3. Then we choose a natural number  $e$  with the following characteristics,

$$1 \leq e \leq \varphi(n) = (p-1)(q-1) \text{ and } \gcd(e, \varphi(n)) = 1$$

and then calculate a natural number  $d$  as follows,

$$1 \leq d \leq \varphi(n) \text{ and } de \equiv 1 \pmod{\varphi(n)}.$$

Because  $\text{gcd}(e, \varphi(n)) = 1$  there exists a number  $d$  satisfying the above conditions and it can be calculated quite fast with the extended Euclidian algorithm. As a consequence of that  $e$  is always an odd number. Now having done all that, the public key consists of the pair  $(n, e)$  and the private key is  $d$ . The number  $n$  is called the RSA-Module,  $e$  is the encryption exponent and  $d$  is the decryption exponent.

If you want to encrypt a message  $m$  with say the public key  $(n, e)$  of Bob then all you have to do is calculate the corresponding cipher text like this  $c = m^e \pmod{n}$  and send it to Bob. He on the other side has to decipher the message by calculating  $m = c^d = (m^e)^d \pmod{n}$ . No one else could have done that not knowing the secret exponent  $d$  [14].

## 2.4 ElGamal

Probably the second most popular algorithm today is ElGamal, which is often referred to as Diffie Hellman, simply because it lies in the core of the ElGamal algorithm. Unlike RSA, the Diffie Hellman algorithm, like discussed above, relies on another particularly difficult mathematical problem, which is calculating the discrete logarithm of numbers in some finite cyclic group  $\mathbb{G}$ . The workflow by this algorithm is as follows:

If we have Bob and Alice who want to communicate securely using the ElGamal algorithm, they have to do the following:

1. First Alice chooses a prime number  $p$  and a generator  $g \pmod{p}$ .
2. Then she chooses a random exponent  $a \in \{1, \dots, p-2\}$  and calculates  $A = g^a \pmod{p}$ .
3. The public key for Alice is  $(p, g, A)$ . The private key is the exponent  $a$ .

Now if Bob wants to send a message  $m \in \{0, 1, \dots, p-1\}$  encrypted to Alice, then he has to supply himself with the authentic public key  $(p, g, A)$  from Alice. Then Here is a sketch of how a PKI could look like:

1. He chooses a random number  $b \in \{0, 1, \dots, p-2\}$  and calculates  $B = g^b \pmod{p}$ .
2. The number  $B$  is the part of the key belonging to Bob according to Diffie Hellman.
3. Bob calculates  $c = A^b m \pmod{p}$  and the encrypted message that he sends to Alice is  $(B, c)$ .

To decrypt the message Alice should do the following:

1. She has to calculate  $x = p-1-a$  and because  $1 \leq a \leq p-2$  the inequality  $1 \leq x \leq p-2$  holds.
2. Then Alice calculates  $m = B^x c \pmod{p}$ .

The following calculation proves that the decryption algorithm does what it is supposed to do:

$$B^x c \equiv g^{b(p-1-a)} A^b m \equiv (g^{p-1})^b (g^a)^{-b} A^b m \equiv A^{-b} A^b m \equiv m \pmod{p}.$$

## 2.5 Digital signatures

Digital signatures are used to achieve the following security goals: integrity, authentication, non - repudiation and timestamping. There are two popular signature schemes, one uses RSA and the other ElGamal. Here I am going to sketch only the RSA signature scheme.

The choice of numbers discussed above by the use of RSA for encryption and decryption are the same. The difference is in their usage. Assuming that we have calculated the special numbers  $e, d, n, p, q$  we can produce a valid digital signature for any given plain text  $m \in \{0, 1, \dots, p-1\}$ . However, by this approach we can produce signatures only to numbers not bigger than the RSA-Module. If we want to sign longer texts, than we need to introduce a hash function, which will not only help us sign longer texts than the RSA-Module, but protect the signature from some attacks, like the existential forgery.

Let's neglect for now the two problems above and focus on the main idea for digital signatures. If Alice wants to sign some plaintext with say value  $m$ , she can do that by computing  $s = m^d \pmod{n}$ , where  $s$  is the signature. As one easily sees, the decryption exponent used in the first application of RSA is now used for producing digital signatures, so it would be quite stupid to use the same key pair for encryption and at the same time for producing signatures. Another interesting fact is that Diffie and Hellman also had the same idea but they couldn't give a concrete implementation that can be used for these purposes.

Now if Bob wants to verify Alice's signature he has to receive from Alice a valid signature with the plain text  $m$  attached to it. Why is that necessary? Well to verify the signature of Alice, Bob has to get Alice's public key  $e$  and calculate to see if  $m = s^e \pmod{n}$ . Thus anyone who knows Alice's public key can verify her signature but the problem with the existential forgery stays and here is where hash functions come into play.

First of all, however, I have to define the notion of existential forgery. A signature is secure against existential forgery if no one can produce a valid signature without knowing the secret signing key. If Alice's signature is not secure against existential forgery than Bob can supply himself with her verification key and then produce signatures that can be verified by anyone as authentic. All that Bob has to do is choose some number  $c$  and set in motion the verification procedure. At the end he will get as outcome, say a message  $m$ , for which if Alice had produced a signature it would have been  $c$ . So now, no one can recognize the messages signed by Alice from those forged by Bob.

To deal with existential forgery one has to use hash functions. Cryptographic hash functions are functions which map data to short bit strings of fixed length. A collision of a hash function is a pair of documents with the same hash value. Of course, a cryptographic hash function is required to be collision resistant, or in other words it should be impossible for anyone to find a collision for this function. The most popular hash function today is SHA-1, which produces hash values in

the form of bit strings of length 160 bits. There are other hash functions, the most common of which are: MD2, MD5 and RIPEMD-160. However, all these hash functions have been broken by now, even SHA-1 (Crypto 05). Luckily, for some parameters, for instance, SHA-1 224/256/384/512 is still secure. There are two very important requirements for hash functions. One is that hash values of data should be easy to calculate and the second one states that it should be practically impossible (with the means available today) to invert the function.

Now instead of just plain text that is submitted for signing one calculates the hash value  $h(m)$  for the message  $m$  that is to be signed. Then we follow the exact same scheme as above and get  $s = h(m)^d \bmod n$  as the signature of the message  $m$ . This time anyone who tries to validate Alice's signature will get  $x = s^e \bmod n$  and to authenticate the signature as valid has to verify that  $h(m) = x$ . Hash functions make signatures resistant against existential forgery, because hash functions are practically non-invertible. If Bob proceeds in the same way as before he will only get a value that is supposed to be a hash value from the verification to which, however, he cannot compute the pre-image and thus gain nothing.

Digital signatures can be produced with ElGamal as well but I am not going to go into detail here. The algorithms briefly discussed above [14] are the backbone of modern cryptography. There are a number of different applications that provide security nowadays but most of them are based on the same algorithms and ideas. There is, however, a new trend that is gaining quickly popularity and it's concerned with identity-based cryptosystems.

## Chapter 3

# Public Key Infrastructure(PKI)

### 3.1 What is PKI?

The problem with public key encryption is that public keys are just numbers which fulfil some specific requirements of the underlying encryption algorithm but convey no information whatsoever about the identity of the person in possession of the corresponding private key. If Bob manages to substitute Alice's public key for his own, he can fool other people that this is the real public key for Alice and then intercept and decrypt her messages. Even worse, Alice won't be able to decrypt those messages because she doesn't know Bob's private key. To deal with that problem digital certificates were introduced, which present proof that some particular public key belongs to user X. A digital certificate is issued by some trusted third party to a user, whose identity was verified prior to certificate issuance. Each certificate holds certain information about the issuer, the subject of the certificate, signature algorithm, the validity period and the signature of the issuer. Dealing with a large number of certificates is a considerable burden over the PKI infrastructure, which has to deal with certificate issuance, revocation, renewal and management. These certificates are built on the X.509 specification of the ISO standard.

Now knowing the basic underlying algorithms, let's see what the public key security system consists of. There are three infrastructural services that comprise the PKI.

- The ***Certification Authority (CA)*** is where users get their public keys signed, binding the authentic user public key with the user's identity.
- The ***Directory***, which is a publicly accessible database of valid certificates
- The ***Certificate Revocation List (CRL)***, which is a publicly accessible database of invalid certificates.

*Note: Some people talk about the CA as the unit responsible for key generation and certificate signing, others use it meaning the whole infrastructure which*

*comprises everything from registration to certificate delivery and services.*

A more accurate approach is to use *trust center* when speaking of the whole infrastructure that participates in each stage of the certificate production process. A trust center consists of a Registration Authority (RA), Key Authority (KA) and a Certificate Management Authority (CMA).

- The RA is the authority which checks one's credentials and ascertains one's identity. Usually that is done by presenting some eligible id document, preferably with picture.
- The KA is responsible for key pair generation, certificate issuing, CRL signing, manufacturing a transport password and a PIN- Envelope and also backup for keys.
- The CMA deals with certificate status, end user key delivery, revocation and dealing with mistakes (which depending on the policies used can cause a restart of the whole procedure or revocation of the certificate and issuance of another one.)

Such a partitioning of the system has its benefits. Sometimes the RA = KA, but for some cases RA  $\neq$  KA is more appropriate. Some of the reasons are that we can, for instance, introduce Local Registration Authorities(LRA) which can reduce the time needed for the user to register (LRAs placed at the company where the user works). Since registration costs time, introducing a number of LRAs will distribute the workload and the registration is made more convenient to the user. Another reason for building such a trust center with distinguished RA, KA and CMA is that we split the responsibilities between these authorities which have to meet different requirements. For example, the CMA must be always on-line so that anyone can check the validity of a certificate and it is responsible for end user key delivery. On the other hand the KA should be preferably off-line, so that there is no chance someone from outside can meddle in the key generation process. A sketch of how a PKI could look like is given in Figure 3.1 [18].

To illustrate how the whole system works I assume that Alice wants to communicate with Bob through a secure connection and that they are both not part of some PKI. The question is: What should they do in order to communicate securely?

To do so they have to inform some Registration Authority, be it a Local Registration Authority or some centralized RA. They have to present some eligible document, preferably with picture on it, like id-card or driving licence to prove to the registration officer that they are who they claim to be. The registration officer can be a colleague of Bob and Alice, which should considerably diminish the chance that the registration officer is fooled for someone's identity. Also if the policies permit it, Alice and Bob can enter the registration information on a web page. Then they will be provided with a one-time password which they can use later to activate the certificate. However, if Alice believes that storing her private key in the trust center for back up purposes is not appropriate, due to say some prejudices, she can generate a key pair of her own and then apply for a certificate. This time she has to present her public key to the registration officer and prove that she possesses the corresponding private key (PoP). The

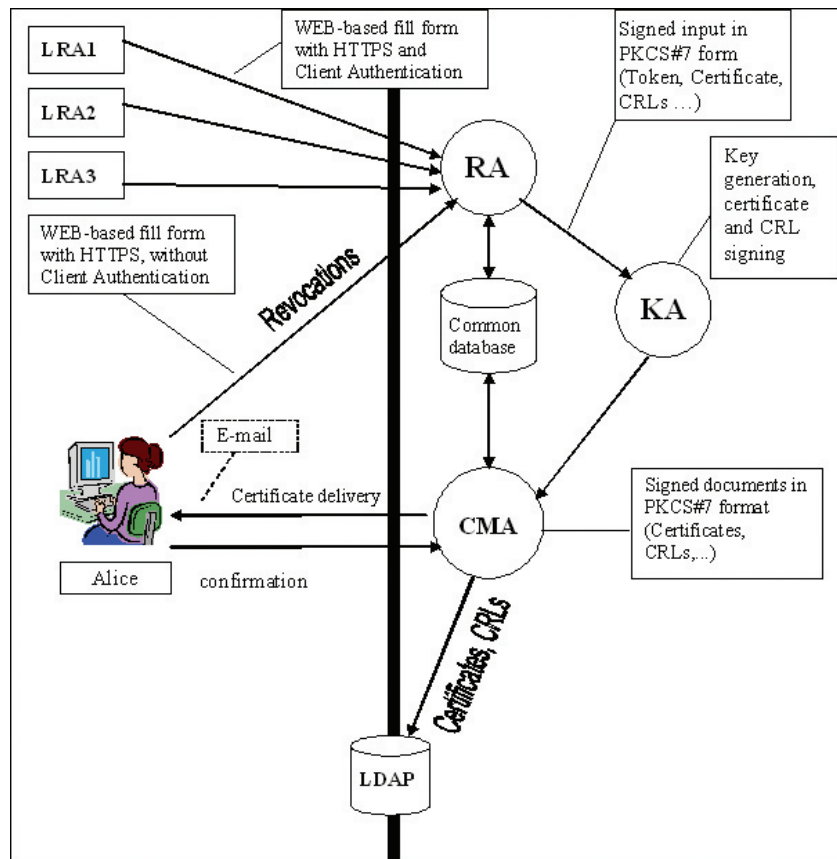


Figure 3.1: Figure

registration officer has to convince himself that no one can meddle with her public key and the identity assigned to it. The main problem by this approach, however, will be where and how to store her private key if she wants to have some reliable security. This can be a serious problem and could require a lot of diligence since there are a number of attacks that can be used to retrieve her private key if she is not careful enough. The other problem is that she is on her own if she wants to decrypt some old message, which was sent to her after she has lost her private key and before she could post a revocation. So the first choice, to let the KA generate a key pair for you and store the private key in a place with considerable security, could be the better choice but only when it comes to storing decryption private keys. Storing private keys used for signing documents is not a good idea, because the KA will have no problems in signing documents in the name of Alice. Even if Alice loses her private key, she will only not be able to produce more signatures but still anyone will be able to verify her signature. All she has to do is apply for another key pair and then sign documents with her new private signing key.

The next step is done by the registration officer, who has to send the data to the central RA, like illustrated in Fig. 3.1, if we have the model with LRA.

Otherwise, the RA signs the registration data in PKCS#7 (Public Key Cryptographic Standard #7) format and sends it to the KA. There the KA generates the key pair which must be completely random, so no ill-intentioned person can follow some pattern in the key generation process to rebuild the pair. Then after generating the key pair, the KA must also make sure that the key pair was permanently deleted from the key generator, meaning that no one should be able to rebuild the keys. Next, it issues a certificate that presents proof that the public key really belongs to the person who presented himself/herself to the RA and then stores the keys in some persistent medium. Apart from key generation, the KA is also responsible for signing certificates and revocation lists.

Since the KA is offline it cannot send or publish the certificate belonging to all users. So the KA sends the data to the next unit of the PKI, the CMA in a signed PKCS#7 form over a secure channel. Personal security environments (PSE) usually contain sensitive data like private keys or some general secret. The CMA delivers the private keys to end users in the form of personal security environments (PSEs) in PKCS#12 format and puts the corresponding certificate in a directory on a Lightweight Directory Access Protocol (LDAP) server. Anyone who wishes to prove for himself/herself the authenticity of the certificate can look up the certificate which resides in the LDAP server in the appropriate directory. After receiving the PSE, the user has to confirm that the delivery was successful.

In a wide-area network, however, these services may be deployed in a hierarchical structure of servers. For example, a CA has its own public key, which is certified from a higher CA in the hierarchy. But the root CA, which lives on the top of the hierarchy, has no one to sign its public key, so it issues a self-signed certificate. (See also Section 3.2.2.)

Having done all that, now Alice and Bob can begin secure communication by supplying with each other's certificates from the LDAP server. Then they have to extract the public key from the corresponding certificate and encrypt their messages. Sounds easy but it is wrong as well, because if we don't check the CRLs it may very well be that the user has lost the private key or something else has happened and the certificate is no longer valid.

For instance, if Alice believes that someone has broken her password and that her private key is no longer safe, she can send a revocation request to the RA, which is usually done over the internet applying Secure Socket Layer (SSL) or Transport Layer Security (TLS) over the Hyper Text Transfer Protocol (HTTP).

## 3.2 The defects in PKI

In a paper written in 1997 [15] Don Davis mentions about the "compliant defects" in the Public Key Infrastructure. He is the one to have coined the name and given a definition meaning: a rule or operation that is difficult to follow and that cannot be enforced. Users in PKI's are the ones that bear the substantial but hidden administrative burden. They are urged to rigorously validate each others' public keys and store their own private keys securely. However, end-users are most of the time unwilling or unable to manage keys diligently.

There are three major shortcomings that deteriorate the general appeal of PKI:

1. Authenticating the user (*Issuance*). How does a CA authenticate a distant user, when issuing an initial certificate?
2. Authenticating the CA (*Validation*). How does one validate the Root CA's public key?
3. Certificate Revocation Lists (*Revocation*). Timely and secure revocation is crucial for PKI. Does the average user follow the rule of checking CRLs before using a public key?

### 3.2.1 Authenticating the user

Usually the CA's administrator should meet face-to-face with the user and check his identity documents. From security point of view this is perfect but assuming that there a million users who want to participate in the PKI meeting them face-to-face is impossible. If still someone takes on the job of meeting a million people face-to-face, this would cause a chain reaction of more employees, more money and finally less appealing PKI. But that's only part of the problem because normally people are not that easy to convince to enrol for a certificate in the first place. They may recognize the advantages to some extent but the idea of having to visit some remote administration officer seems to be quite vexing.

### 3.2.2 Authenticating the CA

As mentioned above, before sending a message one has to check the authenticity of the public key that is about to be used. That means inspecting the certificate that binds the recipient's identity to the corresponding public key. The user has to authenticate it by checking the certifying signature and the signature of each public key in the chain of certifying authorities. However, one often forgets the fact that there is no automatic support offered for validating the Root CA's public key. To do so the user has to hand-check it against some authentic paper copy or using a separate security system like a smart card. It should be also clear that it is not sufficient to have the top level keys in one's browser. Even if they are signed one still has to authenticate the signature's validation key. The problem is that an attacker can replace the top-level CA keys by patching the executables, for example, and then cause the client to accept forged certificates. A probable solution can be to validate the Root CA's public key and store it in a smart card where no one can tamper with its contents. It is self-explanatory that sometimes users are reluctant to go into such detail and perform these tasks but that's the way it should be done in order to have reliable security. Another particularly bad deficiency of the system is that whenever someone neglects the validation process one deteriorates the security of the whole system.

### 3.2.3 Certificate Revocation Lists

Certificate Revocation Lists are known to be the Achilles' heel of public key cryptography. Checking only the signature belonging to a particular certificate is not enough, even if we have a hierarchical structure of CAs and we go up the hierarchy and check each certificate up to the Root CA. We still have to look up in the latest CRL, because it may very well be that the certificate was revoked for some reason. However, checking the public key certificate of the CRL server would require checking a CRL issued from a higher-level CRL server, because

a CRL server cannot testify for its own certificate's currency. At this point it becomes pretty clear that in the presence of a big hierarchy the validation process can be very time-consuming and thus frustrating.

Of course, using cached certificates would substantially decrease the duration of the validation process but a simple virus can corrupt that data and so the only chance to get up-to-date, reliable data is to download it from the CRL server. This kind of data has to be easily accessible, reliable and as fresh as possible, which itself is quite difficult to accomplish for a number of reasons.

There are other issues like private-key management, pass phrase quality and certificate information leakage that make PKI not so appealing to the general user. At that point a question arises: Is there a better way to have the same level of security and yet at a lower cost? Yes, there are some ideas for identity-based schemes, which can offer the same level of security with some adjustments. There are even better approaches like the certificateless public key scheme which comprises most of the benefits of a conventional PKI and the ideas from the identity-based cryptography.

## Chapter 4

# Identity-based cryptography (IBC)

### 4.1 Introduction to IBC

Most of the problems mentioned above were acknowledged since PKI was introduced and proper implementation was given. However, these scaling deficiencies were not that evident since the majority of users were the governments of different countries and maybe some big firms. There weren't any long chains of certificates to validate, since the number of CAs was also quite small. Despite these observations the world was quickly evolving and it was clear that a better scheme was needed.

The first scientist to propose a new revolutionary idea was Adi Shamir, one of the co-founders of the RSA algorithm. In 1984 [13], he proposed an idea based on public key cryptosystems but with entirely new way of generating key pairs. In a normal PKI the most often used cryptographic algorithm is RSA and the public/private keys are just numbers suited for the specific requirements of the algorithm but they convey no information about the identity behind these keys. Somehow one has to make sure that a particular public key belongs to say Bob. This problem was solved by introducing CA which issued certificates providing proof of the connection between a user's identity and a user's public key. Shamir came up with another idea which elegantly deals with this problem by choosing a public key not to be some random number but the user's e-mail address, street address, social security number or any other information that uniquely distinguishes this user from all others. The idea was warmly welcomed because it eliminated the need for CAs which means no certificate validation and no CRLs which tend to be the most annoying part of PKI today. This scheme enables users to communicate with each other without ever exchanging public keys, since if you want to send a secure message you have to at least know the recipients e-mail address or street address which is enough in the presented scheme. Of course, there should be some kind of trust center which will be responsible for private key generation and distribution to end users who join the system for the first time.

In his paper Shamir gave the idea and only an implementation of the identity-based signature scheme using the RSA algorithm. Identity-based encryption

(IBE) was still an open problem because if the RSA module  $n$  was to be derived somehow from the user's identity then even the trust center wouldn't be able to factor it and therefore cannot compute the decryption exponent. It was obvious at the time that with the available tools the needs of IBE could not be met.

## 4.2 Preliminaries

First I would like to present some general notions about security and security models in cryptography that will be important for the rest of the thesis.

The first one is called *computational security*. One says that a cryptosystem is computationally secure if the best known algorithm would need  $N$  iterations (where  $N$  is some considerably big number) to break the code. In other words, there is no polynomially bounded algorithm that can brake the cryptosystem. So, computational security addresses the security of the system from a computational point of view. When in Chapter 2 I talked about one-way functions I mentioned that they are non-invertible only from a computational point of view. In other words, one could have said that one-way functions are computationally secure.

The second and probably the most important topic is *provable security*. A cryptosystem is said to be provably secure if the security of the cryptosystem can be reduced to the difficulty of solving some well known mathematical problem. These security proofs, however, don't provide an absolute proof in the full meaning of the word. They just reduce the security of the cryptosystem which is relative to some other problem but that is no proof, merely a proof under certain assumptions. For instance, it is true in RSA that total break (determining the secret key from the public key) is as difficult as factoring numbers. However, no one knows wether breaking RSA implies the possibility to factor numbers. It is only true that whoever is capable of factorizing numbers is also capable of breaking RSA. Provable security can be used at different levels of cryptographic algorithm analysis. Some of the most common ones are: reducing cryptographic algorithms to mathematical primitives, enhancing security of cryptographic algorithms and reducing cryptographic protocols to underlying algorithms. A more detailed definition of some of these and some other security models can be found in [3, 4].

Another aspect that is crucial in security proofs are the assumptions that we make for the environment where the attack takes place. These assumptions are the rules of the game that an attacker is expected to follow throughout the attack. Besides that we have to consider what kind of information the adversary is acquainted with. What does the adversary know about the cryptographic algorithm? What is known about the particular implementation? And last but not least, what kind of computational power does the adversary have? In security proofs lately is most often acquainted the black box model and an adversary is considered to have placed a successful attack if he succeeds in polynomial time. It is important to note that the real world is much more complex and sometimes additional aspects should be taken into consideration. Like Stinson said [4], the validity and relevance are two different things. By validity of a proof he means the correctness of the proof under the specified environment and by relevance he means the appropriateness of what in the real world exists.

Now I would like to present a couple of Diffie-Hellman related problems which also happen to be a basic part of security proofs for identity-based schemes. Usually the security of an identity-based cryptosystem is reduced to the security of some difficult Diffie-Hellman problem.

The Diffie-Hellman related problems are met quite often when speaking about IBE, so here I am going to discuss the Weak Diffie-Hellman problem (W-DH), the Computational Diffie-Hellman problem (CDH), the Decisional Diffie-Hellman problem (DDH), the MOV reduction and the Bilinear Diffie-Hellman problem (BDH). These and a number of other Diffie-Hellman related problems, like  $k$ -Diffie-Hellman Inversion problem,  $k$ -Strong Diffie-Hellman problem and some more are reviewed in detail in [5].

The **Weak Diffie-Hellman** problem will be defined in a multiplicative or in an additive group. However, since the approach with an additive group will be most often met in this paper I assume that  $\mathbb{G}$  is an additive group of order  $q$  with generator  $P$ . So the W-DH problem is presented as follows:

1. At the beginning we are given the values  $(P, Q, sP)$  where  $P, Q \in \mathbb{G}$  and for some  $s \in \mathbb{Z}_q^*$
2. The problem to be solved is to give the value of  $sQ$  with knowledge about  $(P, Q, sP)$  only.

The **Computational Diffie-Hellman** problem can be defined in both a multiplicative or in an additive group as well. The group  $\mathbb{G}$ , the group order  $q$  and the generator are the same as given above. Here is how the CDH looks like this:

1. We are given  $(P, aP, bP)$  for  $a, b \in \mathbb{Z}_q^*$
2. Our goal is to calculate  $abP$ .

The success probability of an algorithm  $\mathcal{A}$  in solving the CDH problem is given by:

$$Succ_{\mathcal{A}, \mathbb{G}_1}^{CDH} = Prob[\mathcal{A}(P, aP, bP, abP)] = 1 : a, b \in \mathbb{Z}_q^*$$

Then we can say what the **CDH assumption** is. It states that for every polynomial time algorithm  $\mathcal{A}$ , the success probability  $Succ_{\mathcal{A}, \mathbb{G}_1}^{CDH}$  is negligible. Where a function  $f : \mathbb{R} \rightarrow \mathbb{R}$  is said to be negligible if for any  $s > 0$  the following is true:  $|f(k)| < \frac{1}{k^s}$ .

For the next problem I am going to define first what a bilinear map is, then the Decisional Diffie-Hellman problem and finally the MOV problem.

If we have two groups  $\mathbb{G}_1$  and  $\mathbb{G}_2$  both of order  $q$ ,  $\mathbb{G}_1$  with addition and  $\mathbb{G}_2$  with multiplication, where  $q$  is a large prime number and  $P$  is a generator of  $\mathbb{G}_1$ , then a bilinear map that is most often used in IBE is given by  $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ . It has to satisfy three properties in order to be an admissible bilinear map.

1. **Bilinear:** The map  $\hat{e}$  is bilinear if  $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$  for all  $P, Q \in \mathbb{G}_1$  and all  $a, b \in \mathbb{Z}$ .
2. **Non-degenerate:** The function does not map all pairs of  $\mathbb{G}_1 \times \mathbb{G}_1$  to the identity in  $\mathbb{G}_2$ . This implies that  $\hat{e}(P, P)$  is a generator of  $\mathbb{G}_2$ , because the

groups  $\mathbb{G}_1, \mathbb{G}_2$  have prime order and  $P$  is a generator of  $\mathbb{G}_1$ .

**3. Computable:** This states that there exists an efficient (polynomial time) algorithm for calculating  $\hat{e}(P, Q)$  for any  $P, Q \in \mathbb{G}_1$ .

Now we can talk about the **Bilinear Diffie-Hellman problem** (BDH) which is often used as reduction basis in IBE security proofs, like the factoring problem for RSA. In BDH we have the two groups again  $\mathbb{G}_1, \mathbb{G}_2$  of prime order  $q$ , an admissible bilinear map  $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$  and an element  $P$  which is a generator of  $\mathbb{G}_1$ . The BDH is the problem of calculating  $W = \hat{e}(P, P)^{abc} \in \mathbb{G}_2$ , given  $(P, aP, bP, cP)$  for some  $a, b, c \in \mathbb{Z}_q^*$  in  $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$ . Also an algorithm  $\mathcal{A}$  is said to have advantage  $\varepsilon$  in solving the BDH problem in  $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$  if

$$Succ_{\mathcal{A}, \mathbb{G}_1}^{BDH} = Pr[\mathcal{A}(P, aP, bP, cP) = \hat{e}(P, P)^{abc}] \geq \varepsilon$$

where the probability concerns the random choice of  $a, b, c \in \mathbb{Z}_q^*$ , the random choice of  $P \in \mathbb{G}_1^*$  and the random bits in  $\mathcal{A}$ .

Here comes the **Decisional Diffie-Hellman problem** (DDH), once more following the same definitions made above about the group  $\mathbb{G}_1$ . For input of the algorithm we have  $(P, aP, bP, cP)$  and  $(P, aP, bP, abP)$  for some random  $a, b, c \in \mathbb{Z}_q^*$  and a random generator  $P \in \mathbb{G}_1$ . The goal is to distinguish between the  $(P, aP, bP, cP)$  and  $(P, aP, bP, abP)$ . In [7] Joux and Nguyen state that the DDH in  $\mathbb{G}_1$  is easy if there exists an admissible bilinear map  $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ , because

$$c = ab \bmod q \Leftrightarrow \hat{e}(P, cP) = \hat{e}(aP, bP).$$

However, there are examples of groups  $\mathbb{G}_1, \mathbb{G}_2$  and mappings  $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ , where the CDH in  $\mathbb{G}_1$  is hard although the DDH in  $\mathbb{G}_1$  is easy.

In the next problem apart from the assumption made about  $\mathbb{G}_1$ , we are presented with the group  $\mathbb{G}_2$  as given above. There is also a bilinear map  $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$  between those groups. The **MOV** reduction is named over Menezes, Okamoto and Vanstone, who showed in [6], that the discrete logarithm problem in  $\mathbb{G}_1$  is no harder than the discrete logarithm problem in  $\mathbb{G}_2$ . The explanation can be given by a simple example where  $P, Q \in \mathbb{G}_1$  are of order  $q$ . Since  $P$  is a generator of  $\mathbb{G}_1$  then there should be a number  $\alpha$  such that  $\alpha P = Q$ . So the problem is to find  $\alpha$  when we are presented with information about  $g = \hat{e}(P, P)$  and  $h = \hat{e}(Q, P)$ . The idea is that, using bilinearity of  $\hat{e}$  we know that  $h = g^\alpha$  and by non-degeneracy of  $\hat{e}$  that  $g$  and  $h$  have order  $q$  in  $\mathbb{G}_2$ . Thus the discrete logarithm problem in  $\mathbb{G}_1$  is reduced to the discrete logarithm problem in  $\mathbb{G}_2$ . From all that we can infer that the discrete logarithm problem will be hard in  $\mathbb{G}_1$  when for specifically chosen parameters the discrete logarithm problem is hard in  $\mathbb{G}_2$ .

Having done all that, now I am going to present some basic steps in IBE and then a particular IBE scheme presented by Dan Boneh and Matthew Franklin [8].

### 4.3 Basic Concepts

In an identity-based cryptosystem the participants are the users who want to communicate and this distinguished third party called the Key Generation Center (KGC), which is responsible for calculating the private keys.

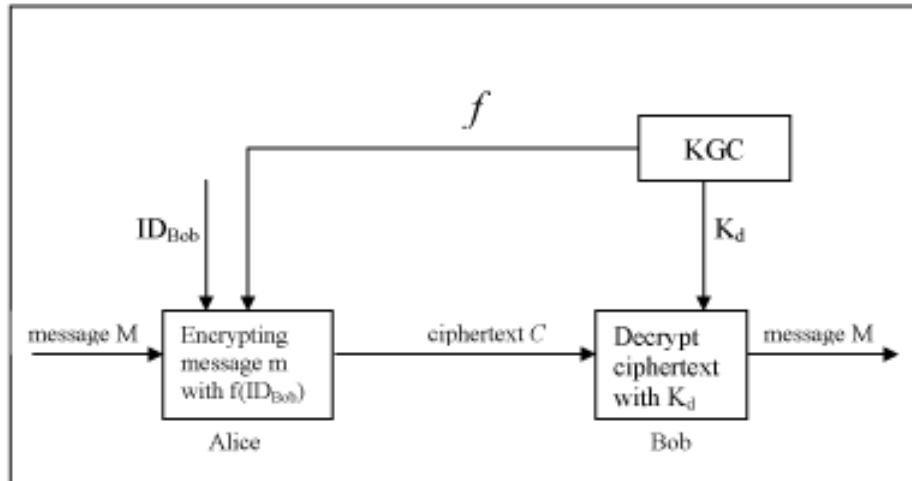
Let's say that Alice wants to make use of the new opportunities offered by IBE and send Bob an encrypted message  $m$ . Assuming that Bob's identity is uniquely determined by his e-mail address, Alice can use it to encrypt the message. There are four basic steps that describe the workflow in an identity-based cryptosystem:

**Setup:** The KGC creates the system parameters which comprise a definition of the finite message space, the finite ciphertext space, the master key, which is only known to the KGC and a public key extraction function  $f$ , which is publicly known to all the participants in the system.

**Extract:** In the extraction phase the KGC takes the public key  $f(ID)$  ( $f$  is a function that extracts from ID the public key) of the participant and using the secret master key calculates the corresponding private key for the user.

**Encrypt:** The encryption is easy. Knowing the ID of the user we want to communicate with, compute the encryption using the public key of the user. The public key is obtained easy, because we know the public key extraction function  $f$  from the KGC and the e-mail address of the receiver. This calculation should be easy and quite fast.

**Decrypt:** Decryption takes place when the other participant in the identity-based cryptosystem receives the ciphertext. In order to decrypt the received message he should authenticated himself in front of the KGC and receive a private key. Then the decryption can take place, simply by using this secret private key. Of course, that can be done in advance.



$f$  is the public key extraction function

$K_e = f(ID_{Bob})$  is the public key belonging to Bob's identity

$K_d$  is the Bob's private key given him by the KGC

*Note: The user would have still been able to receive the encrypted message even if he had not authenticated himself to the KGC. Of course, it would be impossible to decrypt the message but the point is that he can authenticate himself after he receives the encrypted message. It may very well be that Bob has no idea of what IBE is or what a KGC stands for, but he will probably be able to read some message in the e-mail that says something like "If you cannot read this message, you are probably not authorized to do so or you are not yet part of the identity based encryption scheme. If you are not familiar with IBE visit our web site <http://www>". Thus Bob can visit the web address and learn all about IBE and how to decrypt the message.*

## 4.4 Identity-Based Encryption

It was not until 2001, when Dan Boneh and Matthew Franklin proposed an identity based encryption from the Weil pairing [8], that IBE gained considerable popularity. For their scheme any bilinear map  $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$  between the two groups  $\mathbb{G}_1$  and  $\mathbb{G}_2$  can be used as long as the CDH problem is hard in  $\mathbb{G}_1$ . They used the Weil pairing as an example of such a map  $\hat{e}$ .

In order for a cryptographic scheme to be applicable it should provide a certain level of security. Boneh and Franklin proved that their scheme is secure against chosen ciphertext attack under the computational Diffie-Hellman assumption in the random oracle model.

The basic IBE system that Boneh and Franklin present is semantically secure but it does not offer chosen ciphertext security. Only their full version is also semantically secure against adaptive chosen ciphertext attack, which in the standard public key encryption model provides the strongest notion of security. The ideas, however, about semantic security and chosen ciphertext security have to be a little bit adapted to the specific set up in an identity-based system. The standard idea of chosen ciphertext security, meaning for a public key encryption scheme, is not enough for an identity-based scheme. So Boneh and Franklin provide a definition for semantic security similar to the one used in the standard model but with some additional aspects. The idea behind all this is that an attacker in an identity-based cryptosystem can be provided with the private keys of some other users of his choice and secondly the attacker is challenged on a public key of his choice and not on a random one like in the standard model. This observation makes it clear that in order an identity-based cryptosystem to be claimed secure, it must stay secure even when an attacker has some other private keys at his disposal (however, these must be different from the private key corresponding to the public key on which he is challenged on). The adversary can make only private key extraction queries and no decryption queries. So, an identity-based encryption scheme is semantically secure (IND-IN-CPA) if an adversary, who is polynomially bounded, has a non-negligible advantage against the challenger in the following game:

**Setup:** The challenger takes a security parameter  $k$ , runs the Setup algorithm and gives the adversary the resulting system parameters. It keeps, however, the master-key to itself.

**Phase 1:** When the adversary issues private key extraction queries  $ID_1, \dots, ID_m$ . The challenger responds by running algorithm Extract to generate the

private key  $d_i$  corresponding to the public key  $ID_i$ . Then he sends  $d_i$  to the adversary. Of course, these queries may be asked adaptively.

**Challenge:** In this phase the adversary outputs two texts of equal length  $M_0, M_1 \in \mathcal{M}$  and a public key  $ID$  on which he wishes to be challenged. However, the adversary must not have issued an extraction query for that  $ID$  in the previous phase. Then the challenger picks a random bit  $b \in \{0, 1\}$  and sets the ciphertext to be  $C = \text{Encrypt}(\text{parameters}; ID; M_b)$ . It sends  $C$  as the challenge to the adversary.

**Phase 2:** In this phase the adversary can issue more extraction queries  $ID_{m+1}, \dots, ID_n$ . The only constraint is that  $ID_i \neq ID$ , where the challenger responds as in Phase 1.

**Guess:** In the end the adversary outputs a guess  $b' \in \{0, 1\}$  and wins the game if  $b = b'$ . An adversary  $A$  is called an IND-ID-CPA adversary if it follows the rules of the game above. The advantage of an IND-ID-CPA adversary  $A$  against the scheme  $E$  is the following function of the security parameter  $k$ :

$$\text{Adv}_{E,A}(k) = \left| \Pr[b = b'] - \frac{1}{2} \right|.$$

The probability is taken over the random bits used by the challenger and the adversary. The definition states that an IBE scheme is semantically secure against chosen ciphertext attacks if for any polynomially bounded IND-ID-CPA adversary  $A$  the function  $\text{Adv}_{E,A}(k)$  is negligible.

Now there is only one thing left to discuss before I give the basic identity-based encryption scheme and that is random oracles. The random oracle model means that the underlying hash functions used in the scheme are assumed to be ideal random functions [17]. There have been a lot of discussions about the suitability of the random oracle model in security proofs. There is even evidence that there exist cryptographic schemes that have been proven secure in the random oracle model turn out to be insecure in the real world, when we use real hash functions. Nevertheless, the random oracle model is still used (was invented in 1993 by Bellare and Rogaway) in security proofs simply because a proof in the random oracle model gives a good approximation to security in the real world. Douglas Stinson claims in [4] that no security proof is proof in the real world, because the real world is more complex than any security model.

So, to get back to the point, the random oracle model can be looked upon as a function  $H: X \rightarrow Y$  which is chosen at random from the set of all functions that go from  $X$  to  $Y$ . A query to the random oracle at some point  $x \in X$  would cause the oracle to produce  $H(x)$  as an answer.

Finally, the basic identity-based scheme as presented from Boneh and Franklin uses a BDH parameter generator  $G$  and as input for the Setup algorithm a positive integer  $k$ . The BDH generator is a randomized algorithm that takes the security parameter  $k$ , runs in polynomial time, outputs a prime number  $q$ , the description of  $G_1, G_2$  of order  $q$  and the description of an admissible bilinear map  $\hat{e}: G_1 \times G_1 \rightarrow G_2$ . The parameter  $k$  is used to determine the size of  $q$ , meaning that  $q$  can be a  $k$ -bit number. Here are the four algorithms: Setup, Extract, Encrypt and Decrypt.

**Setup:** The key generation center takes as input a security parameter  $k \in \mathbb{Z}^+$  and proceeds as follows:

Step 1: Input  $k$  to  $\mathcal{G}$  to generate a prime  $q$ , two groups  $\mathbb{G}_1, \mathbb{G}_2$  of order  $q$ , and an admissible bilinear map  $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ . Choose a random generator  $P \in \mathbb{G}_1$ .

Step 2: Pick a random  $s \in \mathbb{Z}_q^*$  and set  $P_{pub} = sP$ .

Step 3: Choose a cryptographic hash function  $H_1: \{0, 1\}^* \rightarrow \mathbb{G}_1^*$ . Choose a cryptographic hash function  $H_2: \mathbb{G}_2 \rightarrow \{0, 1\}^n$  for some  $n$ . The hash functions  $H_1, H_2$  are considered to be random oracles in the security analysis.

The message space is  $\mathcal{M} = \{0, 1\}^n$ . The ciphertext space is  $\mathcal{C} = \mathbb{G}_1^* \times \{0, 1\}^n$ . The system parameters are  $\text{params} = (q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P, P_{pub}, H_1, H_2)$ . The master-key is  $s \in \mathbb{Z}_q^*$ .

**Extract:** For a given string  $ID \in \{0, 1\}^*$  the algorithm does: computes  $Q_{ID} = H_1(ID) \in \mathbb{G}_1$ , and sets the private key  $d_{ID}$  to be  $d_{ID} = sQ_{ID}$  where  $s$  is the master key.

**Encrypt:** To encrypt  $M \in \mathcal{M}$  under the public key  $ID$  do the following:

1. compute  $Q_{ID} = H_1(ID) \in \mathbb{G}_1$
2. choose a random  $r \in \mathbb{Z}_q^*$ , and
3. set the ciphertext to be  $C = (rP, M \oplus H_2(g_{ID}^r))$  where  $g_{ID} = \hat{e}(Q_{ID}, P_{pub}) \in \mathbb{G}_2^*$ .

**Decrypt:** Assume that  $C = (U, V) \in \mathcal{C}$  is a ciphertext encrypted using the public key  $ID$ . To decrypt  $C$  using the private key  $d_{ID} \in \mathbb{G}_1^*$  compute:

$$V \oplus H_2(\hat{e}(d_{ID}, U)) = M$$

Now we have only to prove that the decrypt algorithm actually computes the plaintext out of the ciphertext. The following calculations provide proof of that:

$$\hat{e}(d_{ID}, U) = \hat{e}(sQ_{ID}, rP) = \hat{e}(Q_{ID}, P)^{sr} = \hat{e}(Q_{ID}, P_{pub})^r = g_{ID}^r$$

If we take a closer look at the encryption algorithm we will see that  $V$  is actually the plaintext  $M$  being exclusive-ored with the hash of  $g_{ID}^r$ . According to the calculations above  $g_{ID}^r = \hat{e}(d_{ID}, U)$ , so in the decrypt phase  $V \oplus H_2(\hat{e}(d_{ID}, U))$  is actually equivalent to  $V \oplus H_2(g_{ID}^r)$  which on the other hand is equal to  $M \oplus H_2(g_{ID}^r) \oplus H_2(g_{ID}^r)$  and thus one get as a result the plaintext.

Boneh and Franklin proved that this basic scheme is semantically secure against chosen ciphertext attack in the random oracle model or in other words is in IND-IN-CPA assuming that the BDH is hard in groups generated by the Parameter generator. The theorem they gave states:

*If we assuming  $H_1, H_2$  are random oracles then the BasicIdent is semantically secure if the BDH is hard in the groups generated by the Random generator  $\mathcal{G}$ . Suppose we have an IND-IN-CPA adversary that has advantage  $\epsilon(k)$  against the given scheme and then makes  $q_E > 0$  private key extraction queries and  $q_{H_2}$  hash queries to  $H_2$ , then there exists an algorithm  $B$  that solves BDH in groups generated by  $\mathcal{G}$  with advantage at least:*

$$Adv_{G,B}(k) \geq \frac{2\varepsilon(k)}{e(1+q_E)q_{H_2}}$$

Where  $e \approx 2,71$ , the base of the natural logarithm. The running time of  $B$  is  $O(\text{time}(A))$ .

The algorithm presented above is only semantically secure against passive attacks and that is not a very strong notion of security but that was only a base version of the real algorithm. The full scheme is secure against chosen ciphertext attacks and it makes use of a transformation from Fujisaki-Okamoto.

But before going into detail, I would like to point out a few things. First of all chosen ciphertext security or IND-CCA is the standard notion of security for public key encryption schemes but in the case of an identity-based encryption a few adjustments should be made that reflect the specific environment. Like in the previous game the adversary is allowed to make extraction queries, meaning that the adversary is allowed to know any private key for identity  $ID_i$  that is different from the identity on which he is challenged on. The adversary is challenged on a key of his choice and unlike before he is allowed to make decryption queries as well. So Boneh and Franklin say that their scheme is semantically secure against an adaptive chosen ciphertext attack (IND-IN-CCA) if adversary who is polynomially bounded has a non-negligible advantage against the challenger in the following game:

**Setup:** The challenger takes a security parameter  $k$ , runs the Setup algorithm and gives the adversary the resulting system parameters. It keeps, however, the master-key to itself.

**Phase 1:** The adversary issues queries  $q_1, \dots, q_m$  where query  $q_i$  is one of:  
 - Extraction query  $(ID_i)$ . The challenger runs algorithm Extract to generate the private key  $d_i$  corresponding to the public key  $(ID_i)$  and sends  $d_i$  to the adversary.

- Decryption query  $(ID_i, C_i)$ . The challenger runs algorithm Extract to generate the private key  $d_i$  corresponding to  $ID_i$  and then runs algorithm Decrypt to decrypt the ciphertext  $C_i$  using the private key  $d_i$ . In the end it sends the resulting plaintext to the adversary.

These queries may be asked adaptively, that is, each query  $q_i$  may depend on the replies to  $q_1, \dots, q_{i-1}$ .

**Challenge:** Once the adversary decides that Phase 1 is over it outputs two equal length plaintexts  $M_0, M_1 \in \mathcal{M}$  and an identity  $ID$  on which it wishes to be challenged. The only constraint is that  $ID$  did not appear in any private key extraction query in Phase 1. The challenger picks a random bit  $b \in \{0, 1\}$  and sets  $C = \text{Encrypt}(\text{params}, ID, M_b)$ . It sends  $C$  as the challenge to the adversary.

**Phase 2:** The adversary issues more queries  $q_{m+1}, \dots, q_n$  where query  $q_i$  is one of:

- Extraction query  $(ID_i)$  where  $ID_i \neq ID$ . Challenger responds as in Phase 1.  
 - Decryption query  $(ID_i, C_i) \neq (ID, C)$ . Challenger responds as in Phase 1.  
 These queries may be asked adaptively as in Phase 1.

**Guess:** Finally, the adversary outputs a guess  $b' \in \{0, 1\}$  and wins the game if  $b = b'$ . An adversary  $A$  is called an IND-ID-CCA adversary if it follows the rules of the game above. Adversary  $A$ 's advantage in attacking the scheme  $E$  is given by the following function of the security parameter  $k$ :

$$Adv_{\mathcal{E}, \mathcal{A}}(k) = |Pr[b = b'] - \frac{1}{2}|$$

The probability is over the random bits used by the challenger and the adversary.

The definition then states that an IBE is semantically secure against an adaptive chosen ciphertext attack if for any polynomial time IND-ID-CCA adversary  $\mathcal{A}$  the function  $Adv_{\mathcal{E}, \mathcal{A}}$  is negligible.

## 4.5 Chosen plaintext secure IBE

Now the full IBE scheme presented by Boneh and Franklin uses an idea due to Fujisaki and Okamoto which transforms the BasicIdent scheme in a chosen ciphertext secure IBE scheme in the random oracle model. If  $\mathcal{E}$  is the public encryption scheme then by  $\mathcal{E}_{pk}(M; r)$  is denoted the encryption of  $M$  with the public key  $pk$  using the random bits  $r$ . Fujisaki and Okamoto gave a hybrid scheme  $\mathcal{E}_{hy}$  that looks like this:

$$\mathcal{E}_{pk}^{hy}(M) = (\mathcal{E}_{pk}(\sigma; H_3(\sigma, M)), H_4(\sigma) \oplus M)$$

where  $\sigma$  is generated at random and  $H_3, H_4$  are cryptographic hash functions. Fujisaki and Okamoto also proved that if  $\mathcal{E}$  is a one-way encryption scheme then their hybrid scheme is secure against chosen ciphertext attacks in the random oracle model. The idea is that BasicIdent is semantically secure but since semantical security implies one-wayness then Fujisaki-Okamoto's result can be applied if  $\mathcal{E}$  is semantically secure and that is just what Boneh and Franklin did in their FullIdent scheme. Here is the scheme:

**Setup:** As in the BasicIdent scheme. In addition, we pick a hash function  $H_3 : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \mathbb{Z}_q^*$ , and a hash function  $H_4 : \{0, 1\}^n \rightarrow \{0, 1\}^n$ .

**Extract:** As in the BasicIdent scheme.

**Encrypt:** To encrypt  $M \in \{0, 1\}^n$  under the public key ID do the following:

1. compute  $Q_{ID} = H_1(ID) \in \mathbb{G}_1^*$
2. choose a random  $\sigma \in \{0, 1\}^n$
3. set  $r = H_3(\sigma, M)$  and
4. set the ciphertext to be

$$C = (rP, \sigma \oplus H_2(g_{ID}^r), M \oplus H_4(\sigma)) \text{ where } g_{ID} = \hat{e}(Q_{ID}, P_{pub}) \in \mathbb{G}_2$$

**Decrypt:** Let  $C = (U, V, W)$  be a ciphertext encrypted using the public key ID. If  $U \notin \mathbb{G}_1^*$  reject the ciphertext. To decrypt  $C$  using the private key  $d_{ID} \in \mathbb{G}_1^*$  do:

1. Compute  $V \oplus_{H_2}(\hat{e}(d_{ID}, U)) = \sigma$ .
2. Compute  $W \oplus_{H_4}(\sigma) = M$ .
3. Set  $r = H_3(\sigma, M)$ . Test that  $U = rP$ . If not, reject the ciphertext.
4. Output  $M$  as the decryption of  $C$ .

Now since here  $M$  is encrypted as  $W = M \oplus H_4(\sigma)$  this can be replaced by  $W = E_{H_4(\sigma)}(M)$  where  $E$  is semantically secure symmetric encryption scheme [8, 9].

The security of the scheme is summed up in theorem which states the following:

*If the hash functions  $H_1, H_2, H_3, H_4$  are considered to be random oracles then the FullIdent is a chosen ciphertext secure IBE (IND-IN-CCA) assuming that the BDH is hard in groups generated by  $\mathcal{G}$ . More concretely formulated, an adversary  $\mathcal{A}$  has advantage  $\epsilon(k)$  against the scheme FullIdent if  $\mathcal{A}$  runs in time at most  $t(k)$ , makes at most  $q_E$  encryption and at most  $q_D$  decryption queries, and at most  $q_{H_2}, q_{H_3}, q_{H_4}$  queries to the hash functions  $H_2, H_3, H_4$  respectively, then there is a BDH algorithm  $\mathcal{B}$  for  $\mathcal{G}$  with running time  $t_1(k)$  where:*

$$\text{Adv}_{\mathcal{G}, \mathcal{B}}(k) \geq 2FO_{adv}\left(\frac{\epsilon(k)}{e(1 + q_E + q_D)}, q_{H_4}, q_{H_3}, q_D\right)/q_{H_2}$$

$$t_1(k) \leq FO_{time}(t(k), q_{H_4}, q_{H_3})$$

where the functions  $FO_{time}$  and  $FO_{adv}$  are functions defined as in the theorem by Fujisaki-Okamoto [8, 16].

Unfortunately, the proof of this theorem turned out not to be exactly correct but it still holds with a few minor changes. Nevertheless, FullIdent is the breakthrough in identity-based cryptography and it deserves our attention. Even a firm called Volatage Security provides identity-based solutions to securing communication by using the e-mail address of the user or network ID as the user's public key.

Problems with this scheme and identity-based cryptography as a whole are going to be discussed in the last chapter 6 of this paper.

## 4.6 ID-based Signatures from Pairings

In this section I am going to present a more general scheme first and then give a concrete identity-based signature (IBS) scheme due to Florian Hess [10]. Some other IBS schemes will be mentioned as well but they will not be discussed in detail here due to space limitations.

The security of the IBS scheme, like in the IBE one, is reduced to the hardness of the Diffie-Hellman problem. Key escrow is a major problem here but I am going to talk about that in the last chapter. So here is the more general IBS scheme:

The scheme consists of four general algorithms: **Setup**, **Extract**, **Sign** and **Verify**. We have three groups  $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$  of prime order  $q$ , where the first two are groups with addition as group operation and  $\mathbb{G}_3$  with multiplication. Next, we have a monomorphism  $f : \mathbb{G}_1 \rightarrow \mathbb{G}_3$  and a hash function

$H_1 : \{0, 1\}^* \times \mathbb{G}_3 \rightarrow \mathbb{Z}_q^* \times \mathbb{Z}_q^*$ . The various choices of this hash function are discussed in [10]. Then we have a hash function  $H_2 : \{0, 1\}^* \rightarrow \mathbb{G}_2^*$ . Here is what the four algorithms do:

**Setup:** The key generation center (KGC) computes two monomorphisms  $s : \mathbb{G}_2 \rightarrow \mathbb{G}_1$  and  $y : \mathbb{G}_2 \rightarrow \mathbb{G}_3$  where  $f(s(x)) = y(x)$  for all  $x \in \mathbb{G}_2$  and publishes  $y$  but keeps  $s$  secret.

**Extract:** This is again job for the KGC. It takes as input the user's identity and output the secret signing key. If the identity is given by  $ID$  then the secret key belonging to that identity is going to be  $S_{ID} = s(H_2(ID))$ . The public key of the signer is  $P_{pub} = y(H_2(ID))$ , which should be easily computable for everyone as long as he knows the identity  $ID$  of the person.

**Sign:** The signer chooses a random number  $k \in \mathbb{G}_1^*$  and then computes:

1.  $r = f(k)$
2.  $(v, w) = H_1(m, r)$
3.  $u = vS_{ID} + wk$

The signature is given by the pair  $(u, r) \in \mathbb{G}_1 \times \mathbb{G}_3^*$ .

**Verify:** When the verifier receives the signature  $(u, r)$  the verification process goes as follows:

1.  $(v, w) = H_1(m, r)$
2.  $P_{pub} = y(H_2(ID))$
3. The signature is accepted as valid if  $f(u) = (P_{pub})^v r^w$

The verification above makes sense and we can convince ourselves by computing:

$$f(u) = f(vS_{ID} + wk) = f(S_{ID})^v f(k)^w = f(s(H_2(ID)))^v f(k)^w = (P_{pub})^v r^w.$$

There are many concrete versions of the scheme defined above. Hess describes also the possible schemes and presents his proposal for concrete scheme in [10]. This scheme's security is again based on the Diffie-Hellman problem in the domain of the pairing. Now here is how his final scheme looks like:

We have the two groups  $\mathbb{G}_1, \mathbb{G}_2$  of prime order  $q$ , where  $\mathbb{G}_1$  is a group with addition as group operation and  $\mathbb{G}_2$  with multiplication. The generator of  $\mathbb{G}_1$  is  $P$  and the pairing  $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$  satisfies the conditions for an admissible bilinear map. We have also hash functions  $h_1 : \{0, 1\}^* \times \mathbb{G}_2 \rightarrow \mathbb{Z}_q^*$  and  $h_2 : \{0, 1\}^* \rightarrow \mathbb{G}_1^*$ , and the four algorithms: Setup, Extract, Sign, Verify.

**Setup:** Is performed by the trust authority, which picks a random integer  $r \in \mathbb{Z}_q^*$  and computes  $Q_{pub} = rP$ .  $Q_{pub}$  is made public and  $r$  is kept secret.

**Extract:** This algorithm is performed again by the trust authority and it takes as input a user's identity and outputs the corresponding private key to that identity. If  $ID$  is the identity of the signer then the secret key will be  $S_{ID} = rh_2(ID)$ .

**Sign:** A signer wants to sign a message  $M$ , so he takes a random element  $P_1 \in \mathbb{G}_1^*$ , then a random integer  $k \in \mathbb{Z}_q^*$  and then calculates:

1.  $s = e(P_1, P)^k$
2.  $v = h_1(M, s)$
3.  $u = vS_{ID} + kP_1$

So the signature is the pair  $(u, v) \in (\mathbb{G}_1, \mathbb{Z}_q^*)$

**Verify:** The verification of the signature  $(u, v)$  takes place in two steps:

1.  $s = e(u, P) * e(h_2(ID), -Q_{pub})^v$
2. the signature is accepted iff  $v = h_1(M, s)$

## 4.7 Conclusion

IBC makes certificates obsolete, since public keys are the users' identities, most often given by one's e-mail address. That is a big improvement, because certificates are the main burden and reason for the heavy infrastructure of today's PKI. However, like in real life it is always a game of give and take, although very appealing due to the lack of certificate, IBC suffers from a major problem called key escrow. The KGC must be delegated a considerable amount of trust because with its master key it can generate any private key and thus decrypt any message. This is the reason why IBC cannot offer true non-repudiation which is a considerable drawback. There are some ideas of introducing a couple of KGCs and using threshold technologies but like I said there is a price to pay. In this case introducing a number of KGCs would necessitate extra communication and the problem would be low-bandwidth. In the next chapter I will give a quick overview of an idea how to deal with these problems but still keep some of the benefits.

## Chapter 5

# Combining the best of both worlds

### 5.1 Introduction

The idea of certificateless public key cryptography CL-PKC was invented by a few scientists who were in the search of a cryptographic scheme which doesn't have to deal with certificates (thus have a small infrastructure) and that doesn't have the problem of inherent key escrow like in ID based cryptography. S. Al-Riyami and K. G. Paterson presented a CL-PKC scheme [11] which is closely related to the first fully practical IBE scheme presented by Boneh and Franklin [8] but still with no key escrow. It doesn't have certificates, like the name suggests, thus there is no additional cost due to certificate storage, distribution or the computational cost of certificate verification. In CL-PKC the idea to deal with key escrow is to introduce partial private keys that are generated by the KGC but the full private keys are different. This way the KGC does not get to know all the private keys, hence it cannot produce valid signatures in the name of someone else. When a user receives his partial private key from the KGC, he combines it with some secret information to get the full private key. The situation is identical when it comes to public key generation. An entity has to combine the KGC's public key parameters with the same secret information, as the one used to compute the private key, to get its public key. Everything sounds good so far but this slight change in the scheme with the introduction of partial public keys makes the derivation of the public key from the identity alone impossible. Public key information has to be provided in some other way, like some public directory or attaching the public key to the signed data (when we have a signature).

### 5.2 Preliminaries

The definitions are most of the time identical with the ones given when we talked about identity-based public key cryptography. We have again the two groups  $\mathbb{G}_1, \mathbb{G}_2$  of prime order  $q$ , where the first one is a group with respect to addition and the second with respect to multiplication.  $P$  is a generator of  $\mathbb{G}_1$

and we have an admissible bilinear map  $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$  which is typically derived from a Weil or Tate pairing.

Now the security of the scheme is related to the Bilinear Diffie-Hellman (BDH) and the Generalized Bilinear Diffie-Hellman (GBDH) problems. We already discussed the first one, so now a few word about the GBDH.

In the GBDH problem we have  $\mathbb{G}_1, \mathbb{G}_2, P$  and  $e$  as given above. Like in the BDH we are given  $(P, aP, bP, cP)$ , where  $a, b, c \in \mathbb{Z}_q^*$  are random numbers. The problem is with the presented data to find  $Q \in \mathbb{G}_1^*$  such that  $e(P, Q)^{abc} \in \mathbb{G}_2$ . The difference from the BDH is that there  $Q = P$ . Although the GBDH problem is easier than the BDH problem, there is still no algorithm that can solve the problem in polynomial time for appropriately chosen parameters of the system.

This certificateless public key cryptosystem uses a BDH parameter generator (IG), which computes outputs  $\mathbb{G}_1, \mathbb{G}_2$  and  $e$ . The security proofs of the system uses reductions to the BDH or the GBDH in groups generated by the IG.

### 5.3 The CL-PKE scheme

Al-Riyami and Paterson offer a general CL-PKE scheme [11] and then again like Boneh and Franklin [8] use the Fujisaki-Okamoto padding technique [9] to make their system secure against chosen ciphertext attack. Their scheme comprises seven distinguished algorithms: Setup, Partial-Private-Key-Extract, Set-Secret-Value, Set-Private-Key, Set-Public-Key, Encrypt and Decrypt. Here is the full algorithm:

**Setup:** The algorithm is performed by the KGC. The IG takes as input a security parameter  $k$  and outputs two groups  $\mathbb{G}_1, \mathbb{G}_2$  of prime order  $q$ , a generator  $P$  of  $\mathbb{G}_1$ , selects a random master key  $s \in \mathbb{Z}_q^*$  and computes  $P_0 = sP$ . Then it chooses hash functions  $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1^*, H_2 : \mathbb{G}_2 \rightarrow \{0, 1\}^n, H_3 : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \mathbb{Z}_q^*$  and  $H_4 : \{0, 1\}^n \rightarrow \{0, 1\}^n$ . The master key is  $s \in \mathbb{Z}_q^*$ , the message space is  $\mathcal{M} = \{0, 1\}^n$  and the ciphertext space is  $\mathcal{C} = \mathbb{G}_1 \times \{0, 1\}^{2n}$ . The system parameters are then given by  $(\mathbb{G}_1, \mathbb{G}_2, e, n, P, P_0, H_1, H_2, H_3, H_4)$ .

**Partial-Private-Key-Extract:** This algorithm is again performed by the KGC. It takes as input an identifier  $ID_A \in \{0, 1\}^*$  and then calculates  $Q_A = H_1(ID_A) \in \mathbb{G}_1^*$  and outputs the partial private key  $D_A = sQ_A \in \mathbb{G}_1^*$ . Up to this point the algorithm is identical with the algorithm presented by Boneh and Franklin [8]. The correctness of the partial private key can be checked by verifying that  $e(D_A, P) = e(Q_A, P_0)$ .

**Set-Secret-Value:** This algorithm is performed by the user. It takes as input the system parameters and the user's identifier  $ID_A$  as input. It chooses a random number  $x_A \in \mathbb{Z}_q^*$ , which is the secret value for user  $A$ .

**Set-Private-Key:** Performed on the user side. It takes as input the system parameters, the partial private key  $D_A$  for user  $A$  and the corresponding secret value calculated in the previous step. It outputs the private key  $S_A$  by computing  $S_A = x_A D_A = x_A s Q_A \in \mathbb{G}_1^*$ .

**Set-Public-Key:** Once again performed by the user. It takes as input the system parameters, user  $A$ 's secret value and calculates the public key  $P_A = (X_A, Y_A)$ , where  $X_A = x_A P$  and  $Y_A = x_A P_0 = x_A s P$ .

**Encrypt:** There are five steps that comprise this algorithm. Assume we want to encrypt a message  $M$  for user  $A$  with identifier  $ID_A \in \{0, 1\}^*$  and

public key  $P_A = (X_A, Y_A)$  then have to do the following:

1. Check  $X_A, Y_A \in \mathbb{G}_1^*$  and that the equality  $e(X_A, P_0) = e(Y_A, P)$  holds, else abort and output  $\perp$ .
2. Calculate  $Q_A = H_1(ID) \in \mathbb{G}_1^*$ .
3. Choose a random  $\sigma \in \{0, 1\}^n$ .
4. Set  $r = H_3(\sigma, M)$ .
5. Calculate and give as output  $C = (rP, \sigma \oplus H_2(e(Q_A, Y_A)^r), M \oplus H_4(\sigma))$

**Decrypt:** This algorithm consists of four steps. If we have to decrypt the ciphertext  $C = (U, V, W) \in \mathcal{C}$  using the private key  $S_A$  we have to follow the steps below:

1. Calculate  $V \oplus H_2(e(S_A, U)) = \sigma'$ .
2. Calculate  $W \oplus H_4(\sigma') = M'$ .
3. Set the value of  $r' = H_3(\sigma', M')$ , check  $U = r'P$  and if true output  $M$ , else reject  $C$  and output  $\perp$ .

## 5.4 Security of the CL-PKC scheme

The goal is to prove that the scheme complies with the standard notion of security, namely indistinguishability of encryptions against adaptive chosen ciphertext attacks (IN-CCA). In the standard public key encryption this means that the adversary operates in phases. In Phase 1 the adversary  $\mathcal{A}$  may make decryption queries of his choice. Then in the challenge phase  $\mathcal{A}$  chooses two messages  $M_0, M_1$  and gives those to the challenger. The challenger encrypts message  $M_b$  and gives the adversary the ciphertext  $C^*$ . In the second phase the adversary can make further decryption queries but cannot ask for the decryption of  $C^*$ . The attack is over with the adversary's guess  $b'$  for the bit  $b$  above.

Now in an identity-based scheme the situation is more complicated, because adversaries were allowed to make private key extraction queries of identities of it's own choice but not about the identity on which it is challenged on. Apart from that the adversary was also given the freedom to choose the identity on which to be challenged. This extension is also appropriate here, since the compromise of some user's private key should not affect the overall security of the system. To prove that the CL-PKE is secure we have to allow the adversary to make partial private key queries, private key queries or both and since this scheme doesn't offer any certificates it should also be allowed to change the value of some public keys.

To prove that CL-PKE scheme is semantically secure against a chosen ciphertext attack Al-Riyami and Paterson introduced two types of adversaries. Type one is an adversary who has no knowledge of the master key and type two is an adversary that possesses the master key. The second type of adversary is introduced in order to present a security model that deals with the problem of an eavesdropping KGC. An adversary of type one is allowed to extract partial private keys, private keys, make decryption queries and replace public keys, all to identities of it's choice. There are some restrictions, however, that an adversary  $\mathcal{A}$  of type one should recognize. Type one adversary is not allowed to request the private key for an identity to which the public key has already been changed, it cannot extract the private key for the identity on which it will be challenged on.  $\mathcal{A}$  is not allowed to change the public key for the challenge

identity before the challenge phase is over and extract the partial private key for that identity. Last, the adversary is not allowed to make decryption queries on the challenge ciphertext.

Now adversary of type two is supposed to know the master key and thus compute partial private keys of it's choice. It is also allowed to request public keys, make private key extraction queries and decryption queries for arbitrary identities. The difference is that adversary of type 2 is not allowed to replace public keys at any time. This kind of trust in the KGC does not exceed the kind of trust we put in the CA. This is identical to the trust we have in a traditional PKI that the CA is not going to forge certificates. For this type of adversary there are three restrictions: 1. It cannot change public keys, 2. It cannot extract the private key for the challenge identity at any point and 3. It cannot make decryption queries on the challenge ciphertext.

So, the CL-PKE scheme is said to be semantically secure against chosen ciphertext attacks (IND-CL-CCA) if there is no adversary  $\mathcal{A}$  of type one or two that is polynomially bounded and has a non-negligible advantage against the challenger in the following game:

**Setup:** The challenger runs the Setup algorithm with a security parameter  $k$  and gives the result to the adversary. For adversaries of type one the master key is kept secret and for an adversary of type two the master key is given to the adversary.

**Phase1:** The adversary makes use of it's rights and issues decryption queries, extraction queries about partial private keys and private keys. It can also change public keys but it should subject to the restrictions defined above for the corresponding type of adversary.

**Challenge Phase:** The adversary decides on which identity to be challenged on and then produces to plaintexts  $M_0, M_1$  of equal length. Then the challenger chooses a  $b \in \{0, 1\}$  and encrypts the plaintext  $M_b$  with the current public key of the identity on which the adversary is challenged on. If the encryption leads to an abort  $\perp$  (meaning that the adversary has changed the public key and the new public key is not in the correct form), then the adversary loses the game. Otherwise the ciphertext is delivered to  $\mathcal{A}$ .

**Phase 2:**  $\mathcal{A}$  issues requests like in Phase 1 and again follows the rules.

**Guess:** The adversary gives a guess  $b' \in \{0, 1\}$ . The adversary wins if  $b = b'$ . The advantage of the adversary in the game is given by  $Adv(\mathcal{A}) = 2(Pr[b = b'] - \frac{1}{2})$ . Al-Riyami and Paterson propose the following theorem about the security of their scheme:

*If  $H_1, H_2, H_3$  and  $H_4$  are random oracles and that there is no polynomially bounded algorithm that can solve the GBDH problem in groups generated by IG with non-negligible advantage, then the full CL-PKE scheme is IND-CL-CCA.*

The proof of the theorem can be found in the appendix of [11].

In the definitions above it is assumed that the KGC does not replace public keys and that it issues only one copy of each partial private key to the authorized recipient. This assumption, however, reduces the security level of the scheme, so some changes have to be made. Al-Riyami and Paterson offer a technique that makes each entity fix it's secret value and it's public key. Then re-defines  $Q_A = H_1(ID_A || P_A)$ , so that  $Q_A$  effectively binds the entity's identifier and public key. Now the partial private key and the private key are calculated as before but they are also bound now to the entity's choice of public key. Now,

since each entity is restricted to have only one public key it can only compute one private key. Thus, the existence of two public keys can only mean that there are two partial private keys that bind this identity to two different public keys. So whenever a KGC replaces a public key it will be easily recognized. Due to this technique the CL-PKE scheme reaches security level 3 in the hierarchical model described in [12], which is equivalent to the security level of traditional PKI.

Al-Riyami and Paterson also offer a signature scheme that nourish from the same ideas as in CL-PKE. There are again 7 algorithms that comprise the scheme with the difference that the Setup algorithm outputs a definition for the signature space  $S$  and that instead of encrypt and decrypt we have sign and verify. Details can be found in [11].

## Chapter 6

# Conclusions and open problems

There are a number of problems with today's PKI. One of them is that users need to apply for a certificate before a secret communication can commence and that is most often unacceptable in the hectic business sector. The sole certificates that make the whole system work are the main drawback, since they bring up the need of a heavy infrastructure that has to support certificate queries, certificate status queries and certificate revocation. Each certificate has to be protected from unauthorized tampering and still be available for read requests 24 hours a day. Even more important, a user must be able to check the certificate status (revoked, valid) at any time, because the public key could have been already revoked (due to compromise, affiliation change or for some other reason) and thus be no longer valid. The presence of hierarchy makes the above deficiencies even more daunting.

The picture in which PKI fits so far looks very discouraging but PKI has its good sides too. PKI in combination with smartcards is unsurpassable when it comes to authentication and strong digital signatures. There are numerous examples where PKI is used in connection with some kind of smartcard for producing digital signatures and decryption of ciphertexts. However, the problems mentioned above have urged scientists to look for a better solution.

Shamir [13] was the first to come up with the idea of identity-based schemes. He only gave a concrete scheme for an identity-based signature but also the direction of future line of research. Seventeen years later, Boneh and Franklin were the first to give a fully practical and secure IBE. The idea to use one's identity as public key makes certificates obsolete and the infrastructure becomes lightweight compared to traditional PKI. Revocation is dealt by assigning a very small validity period for the private key, like a week or a day. Astonishingly enough, this doesn't mean that the public key has to be changed every time with the private key. The KGC simply produces new private keys to the receiver after a certain authentication process takes place. With this approach it is also possible to send encrypted messages that can be only opened by the receiver in the future (once the receiver obtains the valid decryption key). The other benefits of IBE is that users can securely exchange messages without ever having to worry whether the other party has already enrolled or not.

Apart from all those good sides, IBE has bad sides too. Probably the most important and serious problem of IBE is the inherent key escrow. There are not many people that are keen on the fact that the KGC possesses a key that can produce private keys and thus decrypt any message. This is particularly a serious problem when it comes to digital signatures because such a system cannot offer true non-repudiation, since the KGC can impersonate any user and produce valid signatures. There are some propositions that introduce multiple KGCs and make use of threshold techniques to distribute the master key but this approach introduces the need for extra communication between the KGCs. To sum it up, it turns out that IBE scheme is more suitable for small groups or closed environments, where one can have a considerable trust in the KGC.

There are, however, some new ideas about combining most of the benefits of the standard PKI and those of IBE. One of them is called certificateless PKE and it is closely related to the existing pairing-based public key cryptosystems. The difference is that the KGC issues only partial private keys and thus the problem of key escrow is solved. As consequence of that the public key is no longer computable from some identity or identifier alone. Apart from that there is no need for certificates, revocation can be dealt the same way as with IBE and it can offer true non-repudiation since the private key is known to the legitimate owner.

To put things in a nutshell, standard PKI is not easy to support but it is the most widely used scheme today. Identity-based encryption is gaining quickly popularity and there is a great research activity in that area. Cryptographers are carefully analyzing the pros and cons of IBE schemes and signatures in order to solve potential problems in advance. In my opinion, in the near future IBE will become more and more popular but still stay closely related to Boneh-Franklin's scheme and pairings in general.

# Bibliography

- [1] Diffie, W. and Hellman, M.: "New Directions in Cryptography", IEEE Trans. on Information Theory, IT-22, 6, p. 644-654 (1976).
- [2] Diffie, W. and Hellman, M.: "Exhaustive Cryptanalysis of the NBS Data Encryption Standard", *Computer*, Vol. 10, p. 74-84, June 1977.
- [3] D. R. Stinson *Cryptography: Theory and Practice, Second Edition*. CRC Press, 2002.
- [4] Douglas R. Stinson: "A polemic on notions of cryptographic security", July 2004.
- [5] R. Dutta, R. Barua and P. Sarkar: "Pairing-Based Cryptographic Protocols: A survey", Cryptology ePrint Archive, Report June 2004.
- [6] A. Menezes, T. Okamoto, S. Vanstone: "Reducing elliptic curve logarithms to logarithms in a finite field", *IEEE Tran. On Info. Th.*, Vol. 39, p. 1639-1646, 1993.
- [7] A. Joux, K. Nguyen: "Seperating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups", Jan. 2001, available from eprint.iacr.org.
- [8] D. Boneh, M. Franklin: "Identity-Based Encryption from the Weil pairing", in *SIAM Journal on Computing*, Vol. 32, p. 586-615, June 2003.
- [9] E. Fujisaki and T. Okamoto: "Secure integration of asymmetric and symmetric encryption schemes", in *Advances in Cryptography - Crypto '99*, Lecture Notes in the Computer Science, Vol. 1666, Springer-Verlag, p. 537-554, 1999.
- [10] F. Hess: "Efficient identity based signature schemes based on pairings", proceedings of SAC02. LNCS 2595, p. 310-324, Springer-Verlag, 2002.
- [11] S. Al-Riyami and K. G. Peterson: "Certificateless public key cryptography", Asiacrypt 2003, LNCS Vol. 2894, p. 452-474, 2003.
- [12] M. Girault. Self-certified public keys. In D. W. Davies, editor, *Proc. EUROCRYPT 1991*, LNCS Vol. 547, p. 490-497. Springer, 1992.
- [13] A. Shamir: "Identity-based cryprosystems and signature schemes", in *Advances in Cryptology - Crypto '84*, Lecture Notes in Computer Science, Vol. 196, Springer-Verlag, 2002.

- [14] J. Buchmann: "Einführung in die Kryptographie", ISBN: 3-540-40508-9.
- [15] D. Davis: "The compliance defects in public key cryptography", in *Sixth Usenix Security Symposium Proceedings*, p. 171-178, July 1996.
- [16] E. Fujisaki and T. Okamoto: "Secure integration of asymmetric and symmetric encryption schemes", in *Advances in Cryptography - Crypto '99*, Lecture Notes in Computer Science, Vol. 1666, Springer-Verlag, p. 537-554, 1999.
- [17] M. Bellare and P. Rogaway, *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*, Proceedings of the First ACM Conference on Computer and Communications Security 1993, p. 62-73.
- [18] Exercise 8, belonging to lecture PKI, SS05, held by Prof. Johannes Buchmann in the Technical University Darmstadt