A MODIFIED IDENTITY-BASED ENCRYPTION SYSTEM FOR MESSAGING APPLICATIONS

by

AYŞE GÜL KARATOP

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabanci University
Spring 2008

A MODIFIED IDENTITY BASED ENCRYPTION SYSTEM

APPROVED BY

Assoc. Prof. Erkay Savaş (Thesis Supervisor)	....
Assist. Prof. Yücel Saygın	....
Assist. Prof. Selim Balcısoy	....
Assist. Prof. Murat Kaya	....
Prof. Dr. Aytül Erçil	....

DATE OF APPROVAL: ....

to my family

Acknowledgments

Studying Cryptology in Sabanci University was one of my greatest aims, and I would like to thank some special people who helped me effectively in this arduous period.
First of all, my thesis instructor Assoc. Prof. Erkay Savaṣ. Without his support and confidence in me, I would not be able to complete this thesis. I had a great time and experience assisting his lectures.
Secondly, it is a pleasure to thank the thesis committee, Prof. Dr. Aytül Erçil, Assist. Prof. Yücel Saygın, Assist. Prof. Selim Balcısoy and Assist. Prof. Murat Kaya for their participation and assistance.
Also I would like to thank my friends; Duygu Karaoğlan, Ulvi Kasapoğlu and İsmail Fatih Yıldırım. During this period, they have always provided motivation and support.
Finally and most importantly, I wish to thank my parents for being patient and considerate during my education.

Abstract

Identity-based encryption (IBE) systems are relatively recently proposed; yet they are highly popular for messaging applications since they offer new features such as certificateless infrastructure and anonymous communication. However, recent studies also reveal that the infrastructure needed for IBE systems may be as complicated as the conventional public key cryptosytems and not sufficient research has been conducted in relevant issues concerning the infrastructure.

Firstly, there is the issue of the existence of the Private Key Generator(PKG) as a full-trusted third party. Since PKG generates and knows users' private keys; the user privacy has not been fully achieved. This issue leads to non-repudiation problem where PKG can not only decrypt messages but also can fabricate a valid signature on behalf of any registered user. Secondly, the key-revocation leads tremendous calculations for PKG. In the case of a key-lost, finding a descriptive identity for a user may be difficult. Thus, a new master secret key is generated resulting in changing private keys of every user registered in the system.

With this thesis, a new modified IBE infrastructure is proposed to overcome the stated problems. The master key is secretly shared by two parties, Registration Authority(RA) and Private Key Generator(PKG). In addition, PKG shares the master key with every registered user. With this approach, PKG will not be able to acquire the master key provided that there will be no collusion between the parties, RA-PKG and PKG-users.

Özet

Kimlik tabanımlı şifreleme(KTŞ) sistemleri; yeni sunulmalarına karşın, sertifikasız yapılar ve anonim haberleşmelerde yeni özellikler sunduğu düşünülerek, mesajlaşma programlarında popüleritesini git gide arttırmaktadır. Buna rağmen, yeni araştırmalar gösteriyor ki; KTŞ sistemleri, açık anahtarlama kriptoloji sistemleri kadar kompleks bir yapılandırma gerektirmekte ve bu konuda araştırmalar halen süregelmektedir.
Öncelikle, KTŞ sistemlerindeki Gizli Anahtar üreticisi biriminin tam güvenilir bir durumda olması probleminin üzerinde durulması gerekiyor. Gizli Anahtar üreticisi sistemdeki tüm kullanıcıların gizli anahtarlarını oluşturmaktan sorumlu olmasından ve bu değerleri bilmesinden dolayı kullanıcı gizliliği KTŞ sistemlerinde tam olarak sağlanamamaktadır. Bu durum aynı zamanda kullanıcılar için inkar edememe problemini oluşturmakta olup, Gizli Anahtar Üreticisinin sistemde kayıtlı olan herhangi bir kullanıcı adına imza üretebilme ve kullanıcıların mesajlarını deşifre etmesine olanak sağlamaktadır. İkinci olarak, anahtar geçersiz kılma işlemleri Gizli Anahtar Üreticisi biriminde aşırı hesaplamalara ve iş yüküne neden olmaktadır. Kullanıcıların anahtar kaybetmesi durumunda, kullanıcıya atanacak açıklayıcı bir kimlik bulmak zorluklar getirmektedir. Bu sorunu ortadan kaldırmak için, yapılması gereken sistemdeki ana anahtarın tekrar üretilmesi ve dolayısıyla tüm kullanıcılar için yeniden yeni ana anahtar kullanılarak gizli anahtar oluşturulması gerekmektedir.
Sunulan bu tezle birlikte, Kimlik tabanımlı sistemlerdeki daha önceden bahsedilmiş olan sorunlara çözüm yolu bulunmuştur. Sistem ana anahtarı iki birimde, Kayıt Otoritesi(RA) ve Gizli Anahtar Üreticisi(PKG), gizli bir şekilde paylaştırılmıştır. Aynı şekilde sistem ana anahtarı Gizli Anahtar Üreticisi ile sistemde kayıtlı olan her kullanıcı arasında paylaştırılmış olup, kullanıcı gizliliği sağlanmıştır. Bu yaklaşımla; kullanıcıların Gizli Anahtar Üreticisi ve Kayıt Otoritesinin Gizli Anahtar Üreticisiyle hiç bir şekilde anlaşmaması şartıyla, Gizli Anahtar Üretici hiç bir zaman ana anahtar bilgisine ulaşamamaktadır.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Identity-based encryption (IBE) scheme is a public key cryptosystem where the public keys are unique identities in arbitrary string forms. For instance, e-mail addresses, names, pseudonyms or IP addresses can serve as a public key in IBE systems. The original concept was initially introduced by Shamir in 1984 [17] while the first practical realization of IBE system is based on pairing-based cryptography by Boneh and Franklin [1]. With the advent of pairing-based cryptography new applications of IBE cryptosystem as well as new techniques to realize it more efficiently become the major focus of the contemporary research.

Generally speaking, in IBE cryptosystems, there exists a trusted third party, so-called Private Key Generator(PKG), which is responsible for generating global parameters to be employed in the system as well as the private keys for the registered users. Users obtain their private keys from the PKG, in order to decrypt their messages intended for them. The secure delivery of private keys should be performed over secure channels, where confidentiality and authentication are provided.

IBE is principally a public key cryptosystem, where each user has a public and private key pair. To illustrate, suppose that a user, Alice, wants to send a message to Bob. She encrypts the message with Bob's unique public key, e.g. his e-mail address 'bob@sabanciuniv.edu'. Bob requests the corresponding private key from the PKG, to decrypt the message. The PKG calculates the private key, sends to Bob, and Bob consequently decrypts the message.

Since the bound between a user and its public key is based on an inherent or real-word relationship (e.g. user/name, user/e-mail address, user/assumed role etc.), the need for an infrastructure is seen by some not as comprehensive as the conventional public key infrastructure(PKI). Whereas as elaborately pointed out in [6], a fully-functional IBE system would also require a complex infrastructure in which some aspects have not been fully investigated. Firstly, there is the issue of uniqueness of public keys in IBE since real world names or identities tend to be not unique. Therefore, there should be a registration authority to keep track of used names, i.e. public keys. Secondly, the key revocation could lead to some inconvenience since one may find difficult to obtain a new descriptive name for herself such as finding a new name. One way to revoke a key without actually changing the public key requires that system parameters be changed resulting in changing of private key of every user in the system. And finally, all IBE schemes have the key escrowing property, which is considered as a weakness since the PKG knows the private key of every user. Thus, the PKG can decrypt any message intended for any user and it can fabricate a signature for any message on behalf of any user. The former results in the loss of privacy and anonymity of users in their communications while the latter leads to loss of non-repudiation property.

In this thesis, our contribution is proposing solutions to some of the shortcomings of IBE systems. Limiting our attention to messaging systems such as instant messaging and e-mail applications, we outline an infrastructure for IBE system. Our basic construction follows the idea of secret sharing of the master secret key between two semi-honest parties, namely the private key generator (PKG) and the registration authority (RA). In addition, the PKG shares the same master secret key with each user in a different way, having one share for each user registered in the system. Thus, a user and the PKG have to participate in a protocol to generate the private key for the user. A user's only interaction with the RA is during the registration phase, in which the RA not only checks the uniqueness of the identity but also assists in the protocol that generates two new shares of the master secret key for the user and the PKG. One benefit of our model is that there is no need to employ a secure channel between the PKG and users to deliver private keys since the PKG can send only its share of the private key to users.

We also propose to use ever-changing public keys by attaching date information to the natural identities of users from the perspective of communication models. While public keys changing on daily basis are convenient for instant messaging applications, weekly or monthly public keys in case of asynchronous persistent communication models such as e-mail systems seems feasible. The PKG's share of the relevant public key is sent to user automatically. The PKG does not need to be a global party in the system; there can be many local PKGs serving intranets or subdomains. For instance, the message exchange server is a candidate for a local PKG.

In our infrastructure, the users can freely adopt pseudonyms or nicknames for anonymous communication. The hardness of elliptic curve discrete logarithm problem protects the anonymity of users from the scrutiny of the PKG or any other party.

Our most basic assumption is that the RA and PKG never collude since they are semi-honest and follows the protocol steps exactly. Similarly, we also assume that a user and the PKG do not collude since we believe that there are many incentives for users not to collude with the PKG such as losing their privacy and/or anonymity if they do so.

We give a brief information about identity-based encryption systems and their mathematical background in the second section. The third part includes detailed information about our proposed infrastructure and the security of the proposed scheme is discussed in section four. In section five, the implementation of the proposed infrastructure is explained. Finally, the thesis ends up with the conclusion section.

Chapter 2

Identity-Based Encryption Systems

2.1 Background

Identity-based encryption (IBE) systems utilize elliptic curves and pairing operations as proposed in [1]. The public and private keys, as well as the ciphers are represented as elliptic curve points and the pairing function is performed during encryption and decryption processes. The underlying infrastructure of elliptic curve cryptography is built on finite fields. It is essential to point out some information about the background and underlying key terms utilized in the proposed IBE infrastructure.

2.1.1 Public Key Cryptography

Public-key Cryptography(Asymmetric Cryptography) is a form of cryptography where users are assigned a pair of keys, public and private keys. While public key information is widely distributed, the private key is kept secretly on the user side. Generally, the messages are encrypted with the recipient's public key and can only be decrypted with the corresponding private key pair belonging to the recipient.

2.1.2 Elliptic Curve Cryptography

Elliptic Curve Cryptography is an approach to Public Key Cryptography and built on elliptic curves. The security of the system is directly correlated with the properties of the Finite Field and the Elliptic Curve that are in used in the implementation of the elliptic curve cryptography. Since the system is built on elliptic curves, variables, i.e. public, private keys or messages, are represented as elliptic curve points.

Besides, in the decryption and encryption phases, elliptic curve operations are performed.

2.1.3 Secret Sharing Method

As mentioned in the Introduction section, in order to avoid drawbacks of IBE, a secret sharing method will be utilized between both RA-PKG and PKG-users. With secret sharing, a secret information is shared between parties and each share is stored secretly. While the shares stored on each party do not reveal any information about the shared secret key, the only way to constitute the key is to apply a method where all users must participate. In our infrastructure, the master key will be secretly shared by RA-PKG and PKG-Users. The algorithm of the the secret sharing method will be discussed in section three.

2.1.4 Finite Fields

A field is an algebraic structure, where operations like addition, subtraction, multiplication and division are performed. A finite field or Galois field, F_p or $GF(p)$, is a form of a field that has finitely many elements. The properties of the finite field are explained below:

- The Finite Fields have the form F_p^n, where p is the the characteristic property and n is the order of the field.
- The number of elements in the field is equal to p^n.

2.1.5 Elliptic Curves

An elliptic curve $E(F_p)$ over a finite field F_p is defined with the equation,

$$y^2 = x^3 + ax + b \text{ with } a, b \in F_p$$

The solutions to this equation are called elliptic curve points, and shown as $P = (x, y)$, where x and y are the coordinates and elements of the underlying field F_p. The points on elliptic curve along with so-called point at infinity form an additive group that we can use to define elliptic curve discrete logarithm problem. We can denote the point addition as $P+Q$, and define elliptic curve scalar multiplication of an elliptic curve point P by an integer α, as αP. The order of a point is the smallest integer, n, such that $nP = \mathcal{O}$, where $\mathcal{O}$ denotes the point at infinity, which is the identity element of the elliptic curve group. The security of elliptic curves depends on the difficulty of solving elliptic curve discrete logarithm problem (ECDLP). The ECDLP basically states that given two points Q and P from the equation, $Q = \alpha P$, it is computationally difficult to find α.

2.1.6 Bilinear Map Functions

Bilinear maps over elliptic curve points play a central role in IBE systems. A bilinear map is defined over two groups of the same prime-order q denoted by G_1 and G_2. G_1 is an additive group and is formed of a group of points on elliptic curves while G_2 is a multiplicative group. Bilinear map, therefore, is defined as $G_1 \times G_1 \rightarrow G_2$. Basically, a bilinear map, which is denoted as $\hat{e}(\cdot, \cdot)$, accepts two elements as input from G_1 and returns an element in G_2. Bilinear maps have three properties [7]:

- Bilinearity

$$\hat{e}(xP, yQ) = \hat{e}(xP, yQ) = \hat{e}(P, Q)^{xy} \quad \forall P, Q \in G_1, \forall x, y \in Z_q$$

- Non-degeneracy: The elements of G_1, except $\mathcal{O}$ are mapped to all elements in G_2.

$$\forall P \subset G_1, P \neq 0 \Rightarrow \hat{e}(P, P) \neq 1$$

- Computability: $\hat{e}$ is efficiently computable.

Tate and Weil pairings [13], [8] are the most used pairing functions. Several recent cryptographic schemes utilizes these pairings such as identity-based encryption [1], short signature [3], and efficient broadcast encryption [2]. Our scheme is based on Tate pairing which is, in general, more efficiently calculated than the Weil pairing.

2.1.7 Hash Functions

Private keys of users, in IBE systems, are elliptic curve points. Similarly, the identities are mapped to elliptic curve points using a public function. A hash function,

$H_1 : \{0.1\}^* \rightarrow G_1$, is employed to convert a string of arbitrary length (i.e. identity) to a point on the underlying elliptic curve. In addition to H_1, another hash function, $H_2 : G_2 \rightarrow \{0, 1\}^n$ is used in encryption and decryption phases. For further information about elliptic curves and pairing based cryptography one can profitably refer to [11] and [5].

2.1.8 Homomorphic Encryption

In the proposed infrastructure; Homomorphic Encryption algorithm is used, especially in the registration phase, in order to apply the secret sharing method for the user and the PKG. The Homomorphic Encryption algorithm works as follows:

$$E(m_1)*E(m_2) = E(m_1+m_2) \text{ where } m_1 \text{ and } m_2 \text{ are the encrypted messages.} \quad (2.1)$$

2.2 Work flow of IBE

In general, an IBE System consists of four phases [15]:

1. **Setup phase:** consists of two steps and is performed by the PKG.

    - Selection of the elliptic curve and the master key, s, and the generation of the public key of the system, $P_{SYS} = sP$, where P is the generator point of G_1, group of chosen elliptic curve.
    - Selection of hash functions, H_1, H_2 and the bilinear mapping function.

2. **Extraction:** The private key generator generates the users' private. The public key of a user, having the identity, ID, is denoted as Q_{ID} while the private key of the user is denoted as D_{ID}.

    $$Q_A = H_1(A) \text{ and } D_A = sQ_A \text{ where A stands for the identity of Alice}$$

3. **Encryption:** Encryption is performed by using the receiver's public key (say Alice) as follows:

    - $(U, V) = (rP, M \oplus H_2(g_Q))$

- where $r \in_R Z_q^*$ (i.e. r is randomly selected in Z_q)
- and $g_Q = \hat{e}(Q_A, P_{SYS})^r$ and $\oplus$ denotes exclusive-OR operation.

Here M is the plaintext and the pair (U, V) is the ciphertext, which is consequently sent to Alice.

4. **Decryption:** In decryption phase, the ciphertext (U, V) can only be decrypted if the receiver's private key (D_A) is known. The following steps are applied in decryption process:

$$V \oplus H_2(g_{Q'}) \text{ where } g_{Q'} = \hat{e}(D_A, U)$$

The decryption works since

$$\begin{aligned} V \oplus H_2(\hat{e}(D_A, U)) &= V \oplus H_2(\hat{e}(sQ_A, rP)) = V \oplus H_2(\hat{e}(rQ_A, sP)) \\ &= V \oplus H_2(\hat{e}(Q_A, P_{SYS})^r) \\ &= V \oplus H_2(g_Q) = M \oplus H_2(g_Q) \oplus H_2(g_Q) = M. \end{aligned}$$

Chapter 3

The Proposed Infrastructure

This section describes the main steps in the proposed infrastructure omitting the encryption and decryption phases since they are identical to the original IBE encryption and decryption schemes outlined in the previous section.

3.1 Setup Phase

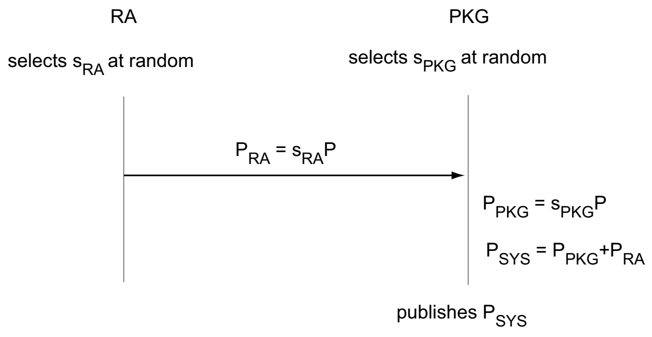

RA
PKG
selects s_{RA} at random
selects s_{PKG} at random
$P_{RA} = s_{RA}P$
$P_{PKG} = s_{PKG}P$
$P_{SYS} = P_{PKG}+P_{RA}$
publishes P_{SYS}

Figure 3.1: The two-party protocol for computing P_{SYS}

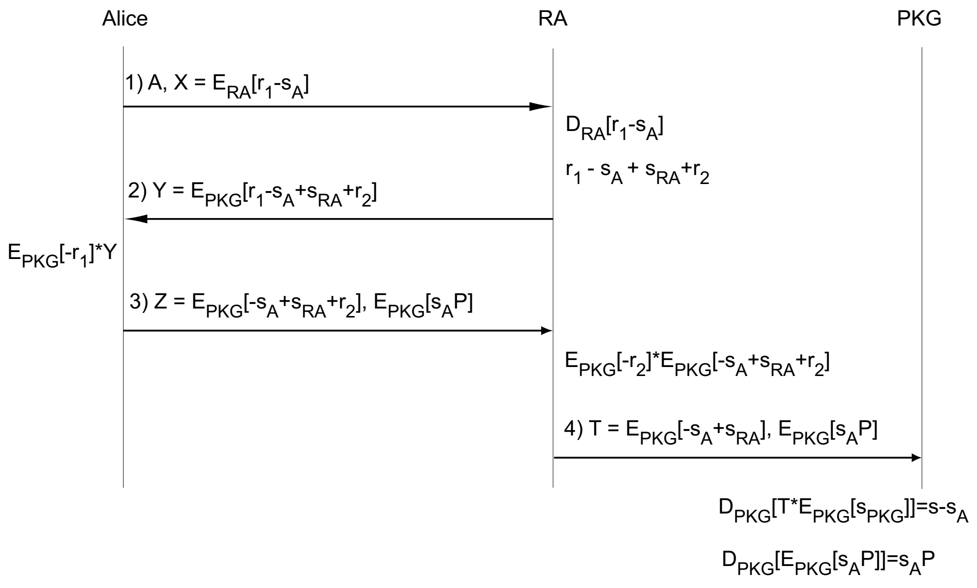
We utilize secret sharing of the master key, s. With this purpose, two semi-honest parties[1] are formed; the Private Key Generator (PKG) and the Registration Authority (RA). The RA is responsible for the registration of users in the beginning while the PKG is responsible for the distribution of private keys. In addition, the RA and PKG share the master secret key as follows: Initially, the RA and the PKG choose two random secret keys, s_{RA} and s_{PKG}, where $s = s_{RA} + s_{PKG}$ is the master

[1]A semi-honest party follows the protocol steps exactly as defined, and does not involve in extra-protocol activities. This is somewhat a weaker assumption than the fully-trusted authority.

key. Since s must not be known by two semi-honest parties, we stipulate that the RA and the PKG do not collude with each other. Indeed, so long as the PKG and the RA do not collude nobody knows the master secret key. A two-party protocol for generating the secret share and the public key of the system P_{SYS} is illustrated in Figure 3.1.

After selecting its secret share of master secret key, the RA computes $P_{RA} = s_{RA}P$, which is its share of public key of the system, and sends it to the PKG. Similarly, the PKG computes its share of system public key, $P_{PKG} = s_{PKG}P$ and performs the elliptic curve addition $P_{SYS} = P_{PKG} + P_{RA}$. Consequently, the PKG publishes the system public key, P_{SYS}.

3.2 Registration phase



Alice
RA
PKG
1) A, X = $E_{RA}[r_1\text{-}s_A]$
$D_{RA}[r_1\text{-}s_A]$
$r_1 - s_A + s_{RA}+r_2$
2) Y = $E_{PKG}[r_1\text{-}s_A+s_{RA}+r_2]$
$E_{PKG}[\text{-}r_1]$*Y
3) Z = $E_{PKG}[\text{-}s_A+s_{RA}+r_2]$, $E_{PKG}[s_AP]$
$E_{PKG}[\text{-}r_2]$*$E_{PKG}[\text{-}s_A+s_{RA}+r_2]$
4) T = $E_{PKG}[\text{-}s_A+s_{RA}]$, $E_{PKG}[s_AP]$
$D_{PKG}[T*E_{PKG}[s_{PKG}]]=s\text{-}s_A$
$D_{PKG}[E_{PKG}[s_AP]]=s_AP$

Figure 3.2: The registration protocol

In the registration phase, the user is first introduced to the system by a secure three-party protocol that involves the user, the RA, and the PKG. The aim of the three-party protocol is two-fold: i) check the uniqueness of the user identity, and ii) securely compute new shares of the master secret and give one share to the user and the other to the PKG. The protocol steps are illustrated in Figure 3.2. The registration phase utilizes public key cryptography and we assume that the user (i.e. Alice in Figure 3.2) knows the public keys of the RA and PKG. $E_{RA}[x]$ and $E_{PKG}[x]$ stand for encryption of x with public key of the corresponding party, i.e. the public keys of RA and PKG, respectively. The PKG uses a homomorphic public key cryptosystem similar to the one in [16]. Therefore, we have $E_{PKG}[m_1] \cdot E_{PKG}[m_2] = E_{PKG}[m_1 + m_2]$.

The protocol steps are explained as follows:

- **Step 1** The user (Alice in Figure 3.2), for the first and last time, contacts the RA by sending her identity (A) in the first message. Alice also encrypts the difference between her secret share s_A and a random number r_1 using the public key of the RA and sends the resulting ciphertext $X = E_{RA}[r_1 - s_A]$ along with her identity A to the RA.

- **Step 2** The RA first checks whether the ID of Alice, A is unique; if not, it helps Alice choose a unique identity. It then obtains the difference $r_1 - s_A$ by decrypting X and adds its own share of the master secret, s_{RA} and another random number r_2 to the difference. It, subsequently, encrypts $r_1 - s_A + s_{RA} + r_2$ and sends the resulting ciphertext $Y = E_{PKG}[r_1 - s_A + s_{RA} + r_2]$ back to Alice. The value r_2 is a random number chosen by the RA.

- **Step 3** Alice removes the random number r_1 by performing the operation $E_{PKG}[r_1 - s_A + s_{RA} + r_2] \times E_{PKG}[-r_1] = E_{PKG}[s_{RA} - s_A + r_2]$. The resulting ciphertext $E_{PKG}[s_{RA} - s_A + r_2]$ is sent to the RA. Alice also sends $E_{PKG}[s_A P]$ to the RA, which serves as a token to authenticate Alice to the PKG in their subsequent transactions.

- **Step 4**[2] The RA removes the random number r_2 similarly and forward $E_{PKG}[s_{RA} - s_A]$ to the PKG which first performs $E_{PKG}[s_{RA} - s_A] \times E_{PKG}[s_{PKG}] = E_{PKG}[s - s_A]$. It then decrypts the resulting ciphertext and obtain its share of master secret $s_{A'}$. Note that $s = s_A + s_{A'}$. The PKG also decrypts $E_{PKG}[s_A P]$ and obtains $s_A P$. Finally, it checks whether the following equality holds: $s_A P + s_{A'} P = P_{SYS}$.

[2]Step 4 of the registration protocol is more complicated than described here due to security considerations. The full description of this step is given in Section 4.2

As a result of the registration phase, the user and the PKG come to posses different shares of the master secret s. Therefore, a user and the PKG must collaborate to generate a private key corresponding to any public key chosen by the user. The fact that they collaborate makes the generation and secure transmission of private key to the user simpler and more efficient as explained in subsequent sections. Provided that none of the users and the PKG do not collude, the master secret will never be revealed. Note that no coalition of users is able to construct the master secret since the user shares themselves do not contain any information about the master secret.

We have two motivations to believe that non-collusion assumption is valid and realistic: i) the PKG is semi-honest, and therefore does not try to learn about the secret shares of the users unless openly told by the users, and ii) a user does not want to reveal its share to the PKG since doing so gives the PKG the ability to access the messages intended for the user and to generate signatures on behalf of the user. Furthermore, the secret share s_A of a user can always be kept in a trusted zone of its hardware and will never leave this zone in the clear. And, we can prevent the user from learning the secret share by employing tamper-proof crypto module as explained in Section 4.2.

3.3 Public Key Selection and Private Key Extraction

In identity-based encryption system, public keys are generally arbitrary strings that contain identity of the user and other relevant publicly available information. Furthermore, the public keys can contain descriptive information about the intended recipient. This clearly alleviates the problem of public key certification used to establish a binding between the public key and the identity of public key owner. Apparently, this bond is inherent in IBE systems. This, nevertheless, complicates the key revocation problem since changing a user's public key entails changing of its identity. Changing one's identity raises certain concerns since finding another descriptive name for an individual may be difficult per se. However, the more important point is the complicated infrastructure (e.g. certification revocation lists) required for informing other users on the compromised or stale public keys.

In messaging applications, on the other hand, the problem of key revocation can be addressed using "ever-changing" public keys. Namely, public key of a user can contain strings related to situational information such as the location, time, date, and role of the user besides the unique identity of the user. We simply propose to include date (or time) information in the identity (hence the public key) of the user. Therefore, the users in our messaging infrastructure has public keys, that are updated frequently. For instance, a user ID may contain date information, such as March 14, 2008, which is a public information and can be appended to the ID easily. The string "bob@openuniv.edu:14/03/2008" is an example for ever-changing public keys.

If the public keys change as frequently as every day, then the corresponding private keys must be re-computed as frequently. As mentioned earlier, both the user and the PKG must participate in the private key generation procedure. In classical IBE systems, the secret key is generated by the PKG and then securely transmitted to the user. Before, the key generation, the user must authenticate itself to the PKG and secure channel must be established between the user and the PKG. Otherwise, the private key can be fallen in the hands of other users or worse yet adversaries. The proposed scheme, on the other hand, utilizes only implicit authentication of the user and does not require a secure channel. The private key generation scheme is illustrated in Figure 3.3.

The user, Alice, selects a public key by appending date and other relevant information to her identity and obtains Q_A, which is sent to the PKG. The PKG then computes its share of the public key $s_{A\prime} \cdot Q_A$ and sends it to Alice. Alice then computes $D_A = s_A \cdot Q_A + s_{A\prime} \cdot Q_A = s \cdot Q_A$, which is her private key, D_A corresponding to the public key Q_A.

3.4 Identification

In case there is a need for explicit identification of the user to the PKG, they can use a modified version of Schnorr's identification protocol as illustrated in Figure 3.4. The effort undertaken by the user is one elliptic curve point multiplication with a scalar and one multiplication and one subtraction in modulo n, where n is the order

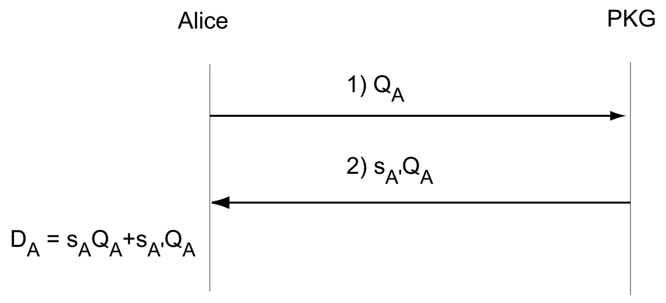


Figure 3.3: Private key extraction protocol

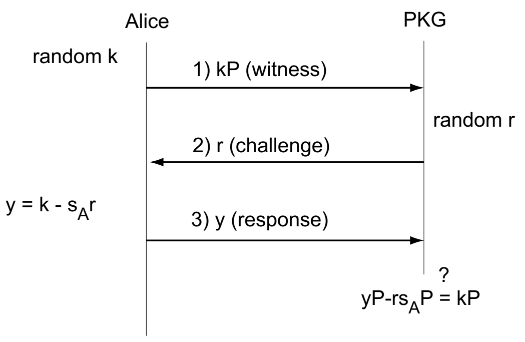


Figure 3.4: User Identification to the PKG

of base point P.

The steps of the identification protocol are summarized below:

- **Step 1** The user Alice, first selects a random integer k and performs the elliptic curve scalar multiplication, kP, where P is the base point for the underlying elliptic curve group. Alice sends kP to the PKG as a witness.

- **Step 2** The PKG selects a random integer r and sends it to Alice as a challenge.

- **Step 3** Alice, upon reception of r, computes $y \equiv k - s_Ar \pmod{n}$ and send the resulting value y to the PKG.

- **Step 4** The PKG computes $yP - rs_AP$, where s_AP serves as the public key of Alice obtained during the registration, and authenticate Alice if the result is the same as the witness kP.

If the PKG needs to authenticate itself to Alice, they can use any identification scheme utilizing the public key of the PKG which is assumed to be in possession of Alice.

3.5 Using Pseudonyms for Anonymity

The users, for anonymity reasons, may want to use nicknames or so-called pseudonyms in their interaction with other users in the system. In the classical setting of IBE systems, the PKG knows both the public key (identity) and private key of every user; hence the anonymity cannot be achieved. One simple trick can, however, help users generate pseudonyms on their own without a help from the PKG. Recall that a user, say Alice, has public and private keys, Q_A and $D_A = sQ_A$ where s is the master secret key. Alice can select a random number k, calculates $R_A = kQ_A$, and declares R_A as her pseudonym. Alice also calculates kD_A and uses it as her new private key in decryption and signing operations. A similar approach is taken in [10], where users can compute their private keys without any assistance from the PKG. One important problem, however, associated with this technique is that the pseudonyms are meaningless random looking bit-strings. Although pseudonyms in this scheme tend do be unique they are also hard to remember. This may be a concern in certain applications such as messaging where nicknames are specifically chosen to be easy-to-remember.

Our approach is based on a technique we call *blinding* of the pseudonyms. As illustrated in Figure 3.5, after selecting a pseudonym, Q_{PN}, Alice blinds it by performing elliptic curve scalar multiplication, kQ_{PN}, where k is the randomly selected blinding factor. The resulting blinded point Q_{BL} is then sent to the PKG that computes $s_{A'}Q_{BL}$ and sends it back to Alice. Alice, finally, computes $k^{-1}(s_{A'}Q_{BL}) + s_AQ_{PN} = sQ_{PN}$. Consequently, Alice declares Q_{PN} (or more specifically PN) as her public key and uses $D_{PN} = sQ_{PN}$ as her private key.

Note that no other party including the PKG and the RA can discover the identity of the user in the proposed anonymity protocol since they do not have the knowledge of the users private keys under the non-collusion assumption. Blinding does not fully achieve the anonymity in the *classical* setting where the PKG is able to compute the private key corresponding to any given pseudonym. Therefore, the PKG can

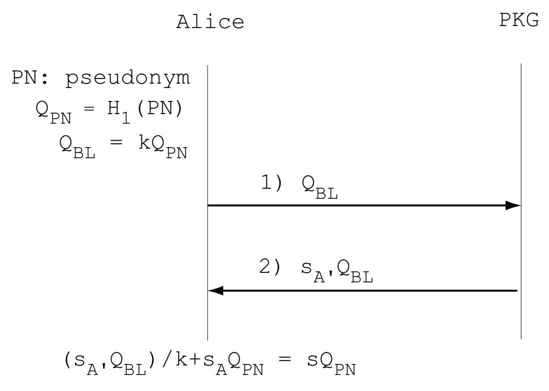


Figure 3.5: Pseudonym generation

decrypt any message being exchanged in the classical setting. However, in the proposed protocol the PKG and RA needs to collaborate to compute a private key, which they will do when a legitimate need arises such as revoking the pseudonym of an ill-behaving user.

Another issue with the anonymity is the uniqueness of chosen pseudonyms. As pointed out in [6], having two users sharing the same pseudonym will result in the loss of security and privacy. Thus, users should check whether it is available before they adopt a pseudonym. One solution to this problem is that the RA publishes an authentic list of used pseudonyms. The users check the pseudonym against this list and notify the RA that the chosen pseudonym is no longer available if it is not in the list. The RA, in turn, updates the list of used pseudonyms. In certain applications users may not want the RA to know the used pseudonyms. In this case, the RA keeps a list for the hash values of the used pseudonyms and users can decide if a pseudonym is used by performing the comparison over the hash values[3].

Another issue using pseudonyms is that there is still a risk that a user may take a pseudonym which is already adopted by another user; a problem which exists in other pseudonym-based schemes as well. Since we do not explicitly address this problem, the protocol in Figure 3.5 does not prevent users from adopting other

[3]If the so-called dictionary attack is of a concern, we may require that the RA perform the uniqueness check. Having a *semi-honest* RA which never involves in extra protocol activities we assume that it does not apply the dictionary attack. Alternatively, the RA can utilize a *secure* comparison protocol.

users' pseudonyms. Instead, a three-party protocol between user, the RA, and the PKG is to be developed to ensure that the users register with unique pseudonyms. A sketch of the protocol, which is based on the one outlined in Figure 3.5 can be given as follows: User anonymously contacts with RA and sends Q_{PN} value. The RA checks if another user has already registered with the same Q_{PN}; if not, the blinding operation is performed by multiplying Q_{PN} with a random value of k, the result is digitally signed with the RA's signature and sent together with k back to the user. In the following step, the user sends $(kQ_{PN})_{sign}$ to the PKG that verifies the RA's signature. If the signature is verified, the PKG computes $s_{A'}kQ_{PN}$ and sends the result back to the user. The received $s_{A'}kQ_{PN}$ value is multiplied with k^{-1} and $s_{A'}Q_{PN}$ is obtained on the user side. Finally; user's private key, sQ_{PN}, is calculated by adding s_AQ_{PN} and $s_{A'}Q_{PN}$. Since our primary application area is that of e-mail, where pseudonyms are not used, a secure pseudonym-creation protocol for other messaging applications requires further effort and security analysis depending on the applications' requirements.

Chapter 4

Analysis of the Proposed Infrastructure

In this section, we analyze the proposed infrastructure from four different perspectives, namely i) security, ii) non-repudiation, iii) validity of public keys, and iv) key revocation.

4.1 Security

Employing two or more trusted parties that do not collude was already proposed by Boneh and Franklin in [1] and also in [4]. In both schemes, a user has to contact all trusted parties to obtain its private key and furthermore the user has to establish a secure channel with each trusted party in this key extraction phase. Our scheme diverges from the previous schemes in two aspects. Firstly, it introduces two trusted-third parties, the private key generator (PKG) and the registration authority (RA), which secret share the master secret s and again do not collude with each other. Secondly, each user shares the same master secret with the PKG in a different way. Therefore, a user does not need to contact both trusted parties to acquire his/her private key since s/he can do so using a protocol involving itself and the PKG (cf. Figure 3.3). Furthermore, the communication between the user and the PKG does not need to be encrypted. Users do not wish to collude with the PKG since otherwise would mean the loss of their privacy and/or anonymity in their messaging transactions.

Our second assumption involves the semi-honest nature of the PKG and the RA. We do not make any assumption on the users of the system; they only try to protect their own interest. Property of semi-honest party was first introduced by Goldreich in [9] and it simply assumes that such parties are honest but curious. In other words, they do not participate in extra protocol activities but gather any leaked information from the protocol. For instance, the PKG will never try to register as a user in the system since this would compromise the master secret to the PKG. The interface for user registration is not available to the PKG. Unless users openly encrypt their private shares of the master secret with the public key of the PKG and send it to the PKG the semi-honest PKG will never learn the private shares of the users. A user will not reveal his/her private share to the PKG or RA since this share also serves as his/her private key in the identification protocol illustrated in Figure 3.4. In other words, a user should not collude with the PKG since doing so will enable the PKG to calculate the master secret s.

Another advantage of the proposed infrastructure is that it provides convenience in key distribution. Only assumption we hold in key distribution is that a user who would like to register knows the public keys of the PKG and RA. Users can acquire this knowledge from publicly available resources such as web pages. Furthermore, a user does not necessarily authenticate oneself to the PKG to obtain the private key since the value sent by the PKG, i.e. $s_{A'}Q_A$, does not contain any information on the private key of the user. The information sent by the PKG becomes useful only if it is received by the intended user.

Considering the difficulty of initial identification of users during the registration as pointed out in [6], we assume that the user is able to prove her identity to the RA during the registration protocol. It could be the case where the user personally goes to the RA and show a piece of identification to prove her identity.

4.2 Non-Repudiation

Non-repudiation, by which a user cannot deny her own transactions with the other entities in the system, is a property almost non-existent in IBE systems. Our infrastructure provides the non-repudiation property under certain assumptions. The first assumption is non-collusion assumption between the PKG and the RA, and a user and the PKG. Since a user's share of the master secret serves also her private key in her interaction with the PKG, such as identification protocol, she can be held responsible for protecting her share from compromise as in the case of private key in conventional public key cryptosystems. However, a user can claim that some other user compromises his own share to the PKG not her and that the PKG becomes able to sign messages for every user (hence the loss of non-repudiation). If this is the case, our infrastructure reduces to classical IBE system. However, the situation with our infrastructure is indeed better than the classical IBE systems since a user's ability to compromise her share of master secret to the PKG can be constrained.

One way of doing it is to employ a trusted tamper-proof crypto module that operates on the secret share of the user and performs all the functions in a protected manner. This module or engine will have a certain interface to the outside applications that can be designed not to leak information on user's share of the master secret. It should be noted that we only try to prevent the leakage on the user's share since its compromise to the PKG will destroy the non-repudiation property of the whole system. Therefore, the primary goal in providing the non-repudiation property is to ensure correct functioning of the system against the disruptive user activities.

There are five instances that a user makes calls to the crypto module's functions that involve her secret share:

1. $E_{RA}[r_1 - s_A]$ in registration phase (cf. Figure 3.2).
2. $E_{PKG}[-r_1] \times E_{PKG}[r_1 - s_A + s_{RA}]$ in registration phase (cf. Figure 3.2).
3. $D_A = s_A Q_A + s_{A'} Q_A$ in private key extraction phase (cf. Figure 3.3).
4. $y = k - s_A r$ in user identification phase in Figure 3.4.
5. $s_{A'} Q_{BL}/k + s_A Q_{PN}$ in pseudonym generation (cf. Figure 3.5)

In instance 1, only the difference of the share s_A to a random value chosen by the module leaves the module; hence it leaks no information on s_A. In instance 4, a zero knowledge protocol is used, which was proved to leak no information on the secret value. In instances 3 and 5, s_A leaves the module as the multiple of an elliptic curve point. This value also does not leak any information based on hardness of ECDLP. In instance 2, the user can call $E_{PKG}[-r_1] \times E_{PKG}[r_1 - s_A + s_{RA} + r_2] \times E_{PKG}[s_{RA}]$ if it obtains $E_{PKG}[s_{RA}]$, which will never be available to her. However, the registration phase must be inspected more closely to reveal the attack possibilities, which we do in the following.

The user may cause the share s_A to be revealed to herself or the PKG by changing the public keys used in the registration protocol. Assuming that the user is capable of changing both public keys, she applies the man-in-the-middle-attack in the registration phase and learns s_A as illustrated in Figure 4.1.

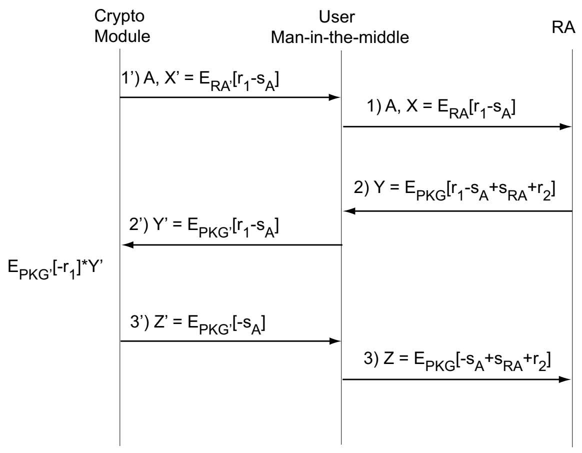


Figure 4.1: Man-in-the-middle attack when user changes the public keys of the RA and the PKG

The user makes the crypto module to use a different public key with which she encrypts $r_1 - s_A$ and returns the ciphertext X'. After having decrypted the ciphertext X' and obtained $r_1 - s_A$, the user re-encrypts it using the RA's real public key and forwards the ciphertext X to the RA. The RA responds to this message in usual way and sends Y to the user, who replaces it with Y'. The ciphertext Y' is the ciphertext of $r_1 - s_A$ encrypted under another public key which the crypto module treats it as the authentic public key of the PKG. However, this public key is in fact chosen by the user who naturally knows the corresponding private key. The user decrypts Z' and obtains the secret share s_A; since she also knows $r_1 - s_A$, the user is able to recover r_1 as well. The user performs $E_{PKG}(-r_1) \times E_{PKG}(r_1 - s_A + s_{RA} + r_2)$ and sends the result $E_{PKG}(-s_A + s_{RA} + r_2)$ to the RA. Therefore, we must prevent the user from changing the public keys employed in the crypto module. A tamper-proof and trusted crypto module can have hardcoded public keys, or public keys which can be changed by only authorized users. The remaining questions to resolve is that whether we protect both public keys or only one of them.

If the user is able to change only one of the public keys then the situation will require another two analysis. Firstly, assume that the user can change only the public key of the RA. Then the man-in-the-middle-attack works as in Figure 4.2. As can be observed from the figure, the user herself cannot recover the secret share s_A, but enables the PKG to do so when the PKG decrypts $E_{PKG}(-s_A)$. Note that the user has to use the correct public key of the PKG since we assume that it cannot replace it.

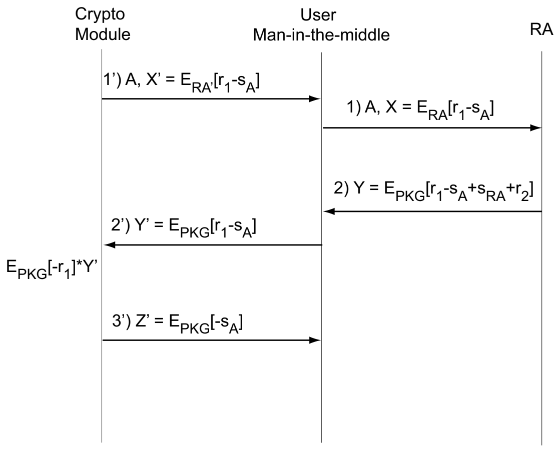


Figure 4.2: Man-in-the-middle attack when user is able to change only the public key of the RA

Finally, in case the user is able to change only the public key of the PKG, since the RA and the hardware module use different public keys no information is revealed to anyone. Therefore, if the user is not able to change the public key of the RA, then the two attacks described above cannot be applied. Therefore, it is sufficient that the tamper-proof crypto module must prevent the user from changing the public key of the RA.

Another type of attack that can be applied by anyone who knows the two public keys is called *query attack* and depicted in Figure 4.3, wherein the secret share of the RA is revealed to the PKG. To prevent this attack, Step 4 of the registration protocol (cf. Figure 3.2) must be modified as in Figure 4.4.

In the modified version of Step 4, the RA does not immediately remove its random number r_2 after it receives the message Z from the user. Instead, it relays

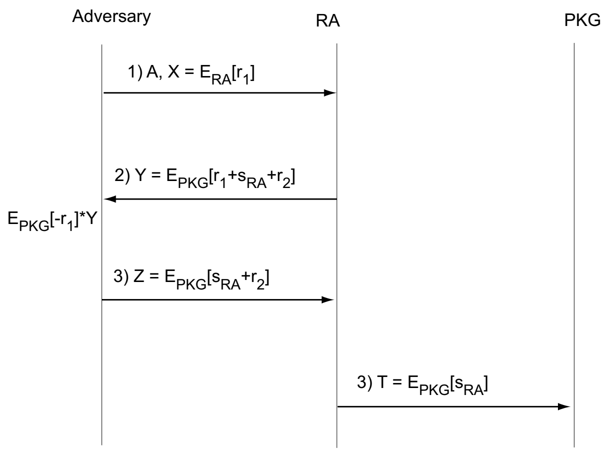
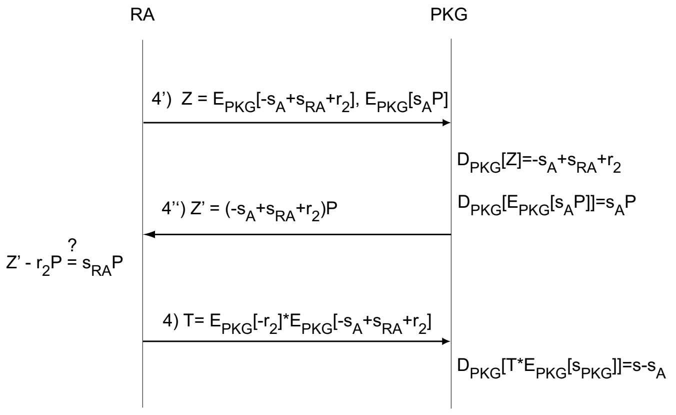


Figure 4.3: Query Attack on the RA

Z to the PKG that decrypts it and obtains $-s_A + s_{RA} + r_2$ (or $s_{RA} + r_2$ in case query attack is applied). The secret share of the RA s_{RA} is protected by the random number r_2 even if the query attack is applied. The PKG performs elliptic curve point multiplication of the base point P by the integer $-s_A + s_{RA} + r_2$ and sends the resulting elliptic curve point Z' to the RA. The RA in turn computes $(-s_A + s_{RA} + r_2)P - r_2P = (-s_A + s_{RA})P$, which would be $s_{RA}P$ in case of the query attack. Therefore, the RA can detect the query attack when $Z' - r_2P$ is equal to $s_{RA}P$; if this is the case it aborts the registration protocol. Otherwise, it continues with regular execution of the step 4 of the registration protocol as described in Figure 3.2.



RA
PKG
4') Z = $E_{PKG}[-s_A+s_{RA}+r_2]$, $E_{PKG}[s_AP]$
$D_{PKG}[Z]=-s_A+s_{RA}+r_2$
$D_{PKG}[E_{PKG}[s_AP]]=s_AP$
4'') Z' = $(-s_A+s_{RA}+r_2)P$
$Z' - r_2P \overset{?}{=} s_{RA}P$
4) T= $E_{PKG}[-r_2]*E_{PKG}[-s_A+s_{RA}+r_2]$
$D_{PKG}[T*E_{PKG}[s_{PKG}]]=s-s_A$

Figure 4.4: Secured Version of Step 4 of Registration Protocol

One can claim that the registration protocol is complicated and may take quite a long time. However, this complication can be tolerated since it is executed only once for each user initially or in case of secret share refreshment which occurs not very frequently. On the other hand, our private key generation and distribution protocols are very efficient and convenient.

In summary, it is not possible to provide non-repudiation in IBE systems without a tamper-proof crypto module that protects the secret share of the users and public key of the RA used in the registration protocol. Best way to implement a tamper-proof crypto module is doing it in the hardware whereby there are many techniques to guarantee the protected and secure execution of cryptographic primitives. Software obfuscation methods to hide secret keys and protect the public key of the RA against replacement can also be used to provide a similar protection. However, obfuscation methods are proven to give way to certain attack types; they only provide limited protection.

4.3 Validity Period of Public Keys

Another issue in the proposed infrastructure is the validity duration of users' public keys. As mentioned earlier, we propose to append date information to IDs of the users. The issue then becomes what sort of date information to use in the IDs. Our approach is to define the duration, depending on the application and the underlying communication model used in the message exchange. For instance, we propose to append day information to the IDs in instant messaging applications where users must be on-line and the communication is transient. The user acquires the PKG's part of the belonging private key in the first login in that day and it computes the private key, which expires next day.

For asynchronous messaging systems such as e-mail, where users are most of the time off-line, we propose to use either date of current week or month information appended to users ID. We believe that to change the public key of the user every week does not constitute too much overhead in e-mail applications. Considering many e-mail messages an average user receives in a week, storing PKG's share of user's private key (a point on the underlying elliptic curve) in the same directory as the e-mails received in that week only marginally increases the storage requirements allocated for that user. Once the user connects to the exchange server for the first time in a week it downloads PKG's share of the belonging private key for that week. If the user has not connected to the e-mail server for more than a week, the e-mail program downloads PKG's share of the private key for previous weeks as well. Note that these downloading operations are done transparently to the user or rather it is pushed to the user by the e-mail server. By adding the user's own share to the PKG's share, the user obtains the private key for the current week and will be able to decrypt any message received that week. Note that when the need arises, for instance by an explicit request from the user, PKG can re-generate its share of any user's private key.

Another approach in determining the validity period of the public keys is to give the sender of the message as an option in e-mailing applications. This way, the sender will choose whether it uses monthly or weekly public key of the recipient.

Our infrastructure can easily accommodate role-based messaging applications as proposed in [14]. Instead of using names, e-mail addresses, pseudonyms, any description for a role or time and space constraints can be used as a public key in our infrastructure.

4.4 Key Revocation Problem

In the proposed IBE scheme, revocation becomes an issue in two different circumstances: i) a particular time-dependent private key, e.g. sQ_A, or ii) secret share of any particular user, e.g. s_A, is compromised. When the former happens, the adversary can decrypt the messages intended for the corresponding user or sign messages on behalf of that user until the expiration date of the corresponding public key. This is the reason why we would like to use frequently changing public keys. The shorter the validity period of a public key, the less likely the corresponding private key being fallen in the hands of an adversary assuming that capturing a private key requires substantial efforts. In addition, the damage is also minimized when a public key is used only for short amount of time. In order to guarantee that no compromised key is used in encryption or signature verification operations, the PKG can publish a (revocation) list of compromised keys. Compromised public keys can be extended with a known public information to generate a new public/private key pair.

If a user compromise her share of the master secret, which we believe is less likely than the former case, the situation must be handled in a different way. The adversary that has the secret share can impersonate the corresponding user, decrypt any messages intended for the user and sign messages on behalf the user. In addition, the adversary can generate a new private key in collaboration with the PKG. Therefore, shortening the validity period does not remedy this situation. In this case, the user must change its share and initiate a new registration phase. With the new share of the master secret, the user implicitly invalidate the old one, with which the adversary cannot extract the private keys. There is no need to keep revocation lists since the user does not have to change its public key after the new share is generated. Adversary revealing the compromised share to the PKG will, however, result in loss of non-repudiation property.

4.5 Scalability

In order to improve the scalability, we can envisage an architecture where there are many private key generators while a single RA is responsible for registering users by associating them with an appropriate PKG. The RA shares the same master secret with every PKG in a different way. A user that wants to register in the system indicates the PKG it prefers to the RA which in turn uses it in the registration protocol. The master secret key is determined by the RA and the first PKG in the system; and a new PKG is introduced into the system by running a protocol similar to the registration protocol as described in Figure 4.5. We assume that private key generators are semi-honest and non-colluding.

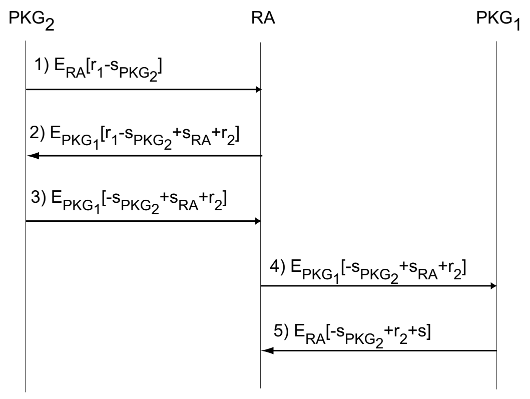

PKG$_2$
RA
PKG$_1$
1) $E_{RA}[r_1 - s_{PKG_2}]$
2) $E_{PKG_1}[r_1 - s_{PKG_2} + s_{RA} + r_2]$
3) $E_{PKG_1}[-s_{PKG_2} + s_{RA} + r_2]$
4) $E_{PKG_1}[-s_{PKG_2} + s_{RA} + r_2]$
5) $E_{RA}[-s_{PKG_2} + r_2 + s]$

Figure 4.5: Introducing a new PKG into the system

Chapter 5

The Implementation

The proposed identity-based infrastructure is built by utilizing Miracl library[12] and is integrated with one of the most used open-source mail applications, Mozilla Thunderbird.

5.1 Multiprecision Integer and Rational Arithmetic C/C++ Library

Multiprecision Integer and Rational Arithmetic C/C++ Library(Miracl) is a big number library provided by Shamus Software Limited and maintains a rich platform to design elliptic curve cryptography implementations. The main asset of of Miracl is that the bilinear mapping functions and the elliptic curve operations are efficient and fast when compared to other big number libraries. The underlying infrastructure of Miracl is built by using C/C++ coding language. Our infrastructure is built upon a supersingular elliptic curve with the equation $y^2 = x^3 + 1 \ mod \ p$ where p has a size of 512 bits so as to provide an equivalent security to 1024-bit RSA. In addition, both Tate pairing and hash functions are provided by Miracl library.

5.2 Mozilla Thunderbird

IBE systems, as pointed out earlier, are convenient for messaging applications. Therefore, a simplified version of the proposed infrastructure is integrated with one of the widely used open-source e-mail applications, Mozilla Thunderbird. In order to embed the proposed infrastructure in an e-mail application, Thunderbird provides a flexible and rich platform. Developing an extension requires a knowledge of XUL and JavaScript languages. While XUL language is generally used to define new window elements such as bars, menu items or buttons on the user interface, JavaScript is utilized to assign events on the created window elements and also executes the underlying C++ built processes such as encryption and decryption routines.

Before building an extension, the development environment has to be prepared. Preparing the development environment involves populating the appropriate extension folders with specific files. This extension (add-in) folder contains the following items.

- /chrome.manifest: Specifies where the chrome files are located.
- /install.rdf: The description file of the extension.
- /locale/*:
- /locale/en-US: Contains translation for text string codes in *.xul files.
- /defaults/:
- /defaults/preferences/*.js: Contains
- /content/: Contains two types of files with the extensions, .xul and .js. While xul file contains the definitions of user interface elements, such as buttons, menu items; the javascript file is the script file in which the actions are defined.
- /skin/:

After the development environment is prepared, one may start developing the extension by first writing the definition of the extension in the XML formatted install.rdf file, which is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:em="http://www.mozilla.org/2004/em-rdf#">
  <Description about="urn:mozilla:install-manifest">
    <em:id>encry@encry.org</em:id>
    <em:name>encry</em:name>
    <em:version>1.0</em:version>
    <em:creator>encry Team</em:creator>
    <em:homepageURL>http://www.encry.org/</em:homepageURL>
<em:optionsURL>chrome://encry/content/options.xul</em:optionsURL>
    <em:targetApplication>
      <Description>
        <em:id>{3550f703-e582-4d05-9a08-453d09bdfdc6}</em:id>
        <em:minVersion>1.5</em:minVersion>
        <em:maxVersion>2.0.0.*</em:maxVersion>
      </Description>
    </em:targetApplication>
  </Description>
</RDF>
```

The description of the install.rdf is given in Table 5.1.

Table 5.1: The Properties of Install.rdf

Tag	The Functionality
xml version	Defines the xml version of the document
id	The ID of the extension, given in e-mail address format
name	The name of the extension
version	The version of the extension
creator	String information that represents the the designer of the extension
homepageURL	Homepage URL of the extension
Description/id	Thunderbird's application ID
Description/minVersion	The minimum version of the Thunderbird the extension can work with
Description/maxVersion	The maximum version of the Thunderbird the extension can work with

The chrome.manifest file references the chrome, or namely, the source files, where the scripts and xul files defining the functionality of the extension, are located. A sample of chrome.manifest file for the plug-in *encry* is shown in Table 5.2.

Table 5.2: The chrome.manifest File

```
content encry content/
locale encry en-US locale/en-US/
locale edithtml en-US locale/en-US/
skin encry classic/1.0 skin/

overlay chrome://messenger/content/messenger.xul
chrome://encry/content/messengerOverlay.xul
overlay chrome://messenger/content/messageWindow.xul
chrome://encry/content/messengerOverlay.xul
overlay chrome://messenger/content/messengercompose/messengercompose.xul
chrome://encry/content/composeOverlay.xul
```

The locale folder shall include a folder, named en-US, in which the translation of the text string codes in *.xul files are located. Every extension has an option box that provides a user-friendly interface for the users, allowing to modify the preferences of the extension. The structure and functionality of this box is provided by the javascript files, located in the preferences folder. The functionality and behavior of the extension are defined with the javascript and xul files, located in the content folder. While the XUL file provides adding new user interface elements on Thunderbird, the Javascript file contains the action and behavior information of these elements together with the extension's functionality.

5.3 The Integration of MIRACL and Thunderbird

As pointed out earlier, the implementation comprises two tiers, MIRACL and Mozilla Thunderbird. While MIRACL covers the security aspect, Mozilla Thunderbird constitutes the communication part of the project. Since these two components are implemented in different programming languages, it is also important to explain how they are integrated to each other in order to conceive their cooperation.

In the implementation part of this thesis; three add-ins have been developed, for the RA, the PKG and the user separately, since these entities have different roles. Each plug-in contains the executable files and the plug-in source code. The c++ built processes, that involve the cryptographic operations, are invoked automatically by Thunderbird whenever the appropriate action takes place. For instance, the encryption process is called from Thunderbird, each time the user sends an email. It is important to mention that the plug-in also serves as an environment to store the cryptographic parameter files, such as private keys and public keys. As shown in Figure 5.1, the entities communicate via mail server. The e-mail server plays the central role and generally works as a bridge between PKG, RA and the user. In addition, two additional email addresses have been registered for the RA and the PKG, namely 'ibe_ra@sabanciuniv.edu' and 'ibe_pkg@sabanciuniv.edu'.

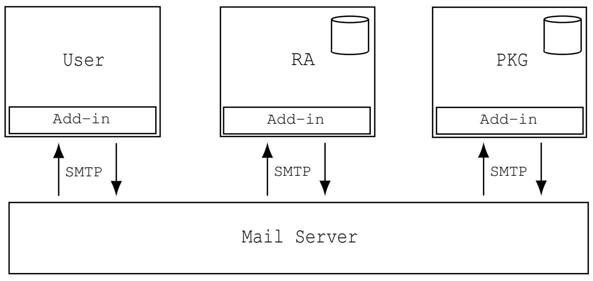


Figure 5.1: The System Architecture

After the extension is installed all phases are handled automatically except for the decryption phase. The specific details are given below:

1. **Setup Phase:** When started, both add-ins on RA and PKG's side randomly select secret numbers and each calculates their shares of the system public key. The setup phase is initiated when the RA automatically sends its share of the system public key to the PKG via an e-mail message whose subject is 'IBE_Setup'. The add-in on PKG's side has an event listener which checks for the sender and the subject field of the newly arrived mails. If the received

mail's subject is 'IBE_Setup' and the sender is 'ibe_ra@sabanciuniv.edu' then PKG executes its setup phase and ends up with calculating the public key of the system. We assume that the PKG and RA have a secure channel to communicate.

2. **Registration Phase:** When a user installs the Modified Identity-based Encryption(MIBE) plug-in, the registration steps are transparently done and automatically executed on each party. Specifically the registration phase is initiated by the user's plug-in when Thunderbird is firstly launched on user side. The plug-in automatically selects the user secret share(s_{ID}) and the random number, r_1, encrypts the subtraction of these values and sends the result to RA. If the user's mail address is unique, RA adds the user's mail address to its database and executes the following registration phase steps. Otherwise, if the user mail address is not unique, it sends an e-mail with the subject 'REGISTRATION DENIED' to the user and warns to register with another e-mail address.

3. **Encryption Phase:** When a user clicks on the send button; the encryption process in the add-in is called, the mail body is automatically encrypted and sent to the recipient. The encryption is performed using enveloping method whereby a symmetric secret key is encrypted by the public key of the recipient which is equal to its e-mail address concatenated with the date information(Day.Month.Year).The body of the message is encrypted by the symmetric key using AES.

4. **Decryption Phase:** Users can decrypt their messages by clicking on the decrypt menu item as shown in Figure 5.2.

Since we do not employ a secure crypto module to guarantee a secure execution of cryptographic primitives, we hardcoded the public key of the RA in the user add-in program and do not allow users to change it. Similarly, the secret share of the user s_A is not kept in a place where the user cannot easily access. We did not implement the user identification and pseudonym generation protocols. The private key extraction protocol is implemented in a different manner from Figure 3.3. The PKG periodically e-mails its share of user private key to each user, from

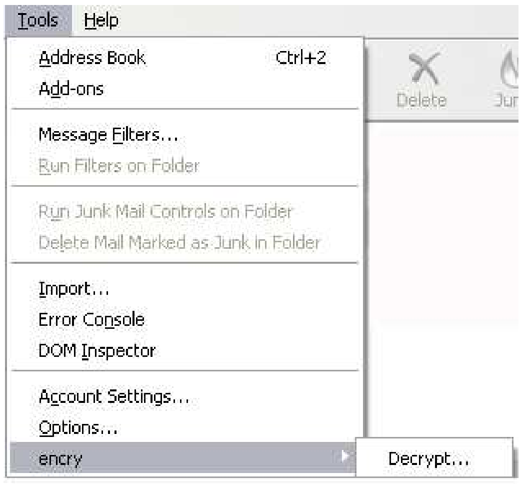


Figure 5.2: The Decrypt Menu Item

which the user add-in computes the private key of the user. The messages in all the implemented protocols are sent as e-mails with a unique identifying description in the subject field. The corresponding add-in programs constantly check for the subject fields of every e-mail messages to take an appropriate action.

5.4 Performance

Intel Celeron 1.5 Ghz computer is used as a base platform together with its Windows XP operating system. Table 5.3[1] features the execution times of the cryptographic operations in different protocols for each party, namely PKG, RA and the user. The numbers indicated below each party, show the execution times of the corresponding processes, relevantly in terms of milliseconds. Clearly, our infrastructure not only offers a secure infrastructure but also provides an efficient system with high execution performance.

The data to be stored on each party is also another important aspect to be taken into account since it heavily affects the implementation in terms of its performance. The size of the data in bytes, to be stored on each entity, is shown in Table 5.4.

[1]Note that the communication latencies are excluded. Table 1 is constructed by running user side only one time, considering the latency requirement on user side; and both PKG and RA 100 times, since throughput is a concern.

Table 5.3: The Performance

Process	PKG(ms)	RA(ms)	User(ms)
Computing P_{SYS}	16	17	-
Registration	242	271	140
Private Key Extraction	16	-	20
Pseudonym Generation	16	-	60

Whereas the C++ built processes are indicated as executables, the parameters and variables are stored in data files with the extension '.ibe'. It should also be pointed out that the size of executable files may vary depending on the implementation and deployment method. Supporting this fact, the larger size of database connection executables deployed in PKG and RA side are introduced by their implementation in C#.

Table 5.4: The File Storage

Entity	File name	File size(Mb)
PKG	Data Files	2.39
	Executables	16.6
RA	Data Files	2.63
	Executables	32.6
User	Data Files	2.00
	Executables	1.70

Chapter 6

Conclusion

In this thesis, a new IBE infrastructure that is intended for utilization in messaging applications is proposed. The proposed infrastructure aims to solve some inherent drawbacks of the IBE systems while retaining their advantage. Key escrowing problem is solved by a method where users and the private key generator secret shares the master secret key. The omniscient private key generator in classical IBE systems which knows all private keys is replaced by a semi-honest third party that does not have information about these private keys. In the presence of the semi-honest private key generator, it is possible to have anonymous and secure communication under the non-collusion assumption. We also made investigations as to how the non-repudiation property can be provided in our infrastructure. We implemented the cryptographic protocols used in the proposed infrastructure and demonstrated that computational requirements for the parties are acceptable. We also integrated an e-mail system with the proposed infrastructure and are currently experimenting with the implementation to evaluate its convenience to the users.

Bibliography

[1] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology - CRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA*, volume 2139, page 213, CA, USA, 2001. Springer Berlin / Heidelberg.

[2] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphetexts and private keys. In *CRYPTO 2005*, volume LNCS 3621, pages 258–275. Springer Berlin / Heidelberg, 2005.

[3] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *ASIACRYPT 2001*, volume LNCS 2240, pages 514–532. Springer Berlin / Heidelberg, 2001.

[4] L. Chen, K. Harrison, D. Soldera, and N. Smart. Applications of multiple trust authorities in pairing based cryptosystems. Hewlett-Packard Trusted Systems Laboratory, HPL-2003-17, February 2003.

[5] H. Cohen and G. Frey. *Handbook of elliptic and hyperelliptic curve cryptography.* Chapman and Hall/CRC, 2006.

[6] Y. Desmedt and M. Burmester. *Identity-Based Key Infrastructures*, page 167. IFIP International Federation for Information Processing. Springer, 2004.

[7] R. Dutta, R. Barua, and P. Sarkar. Pairing based cryptography: A survey. 2004.

[8] G. Frey and H.-G. Ruck. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Math. Comp.*, 62(206):865–874, 1994.

[9] Oded Goldreich. Secure multi-party computation. Working Draft, 2000.

[10] D. Huang. Pseudonym-based cryptography for anonymous communications in mobile ad hoc networks. *International Journal of Security and Networks*, 2(3–4):272–283, 2007.

[11] N. Koblitz. *A Course in Number Theory and Cryptography.* Springer-Verlag, 1994.

[12] Shamus Software LTD. Miracl. http://www.shamus.ie, 2005.

[13] A. Menezes, T. Okamoto, and S. A. Vanstone. Reducing elliptic curves logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.

[14] M. C. Mont, P. Bramhall, C. R. Dalton, and K. Harrison. A flexible role-based secure messaging service: Exploiting ibe technology in a health care trial. Hewlett-Packard Trusted Systems Laboratory, HPL-2003-21, February 2003.

[15] L Owens, A. Duffy, and T. Dowling. An identity based encryption system. In *ACM International Conference Proceeding Series; Vol. 91, Las Vegas, Nevada, USA*, volume 2139, pages 154–159, Las Vegas, Nevada, USA, 2004. Trinity College Dublin.

[16] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - EUROCRYPT '99, LNCS Vol. 1592*, pages 223–238, Las Vegas, Nevada, USA, 1999. Springer.

[17] A. Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology, Santa Barbara, California, USA*, pages 47–53, CA, USA, 1985. Springer-Verlag New York, Inc.