An Efficient Identity-based Cryptosystem for End-to-end Mobile Security

Jing-Shyang Hwu, Rong-Jaye Chen, *Member, IEEE*, and Yi-Bing Lin, *Fellow, IEEE*

Department of Computer Science & Information Engineering
National Chiao Tung University
{jshwu, rjchen, liny}@csie.nctu.edu.tw

Abstract—In the next generation mobile telecommunications, any third party that provides wireless data services (e.g., mobile banking) must have its own solution for end-to-end security. Existing mobile security mechanisms are based on public-key cryptosystem. The main concern in a public-key setting is the authenticity of the public key. This issue can be resolved by *identity-based* (*ID-based*) cryptography where the public key of a user can be derived from public information that uniquely identifies the user. This paper proposes an efficient ID-based encryption algorithm. We actually implement the ID-based encryption schemes and compare the performance to show the advantage of our approach. Our study indicates that our solution outperforms a previously proposed algorithm by 20~35%.

Keywords: End-to-end Security, Identity-based Cryptography, Mobile Banking, Public Key.

1. Introduction

In the recent years, the third generation (3G) and beyond 3G (B3G) mobile telecommunications networks [12] have been widely deployed or experimented. These networks offer large bandwidths and high transmission speeds to support wireless data services besides traditional voice services. For circuit-switched voice services, mobile operators have provided security protection including authentication and encryption. On the other hand, wireless data services (such as mobile banking) are likely to be offered by the third parties (e.g., banks) who cannot trust the security mechanisms of mobile operators. In this case, the third parties must have their own solution for end-to-end security [13]. End-to-end security mechanisms used in mobile services are typically based on public-key cryptosystem.

In public-key cryptosystem each user has a key pair (K_U, K_R), where K_U is the public key and K_R is the private key. To generate the key pair, one first chooses a

private key K_R and applies some one-way function to K_R to obtain a random and uncontrollable K_U. The main concern in a public-key setting is the authenticity of the public key. If an attacker convinces a sender that a receiver's public key is some key of the attacker's choice instead of the correct public key, he can eavesdrop and decrypt messages intended for the receiver. This is the well known man-in-the-middle attack [23]. This authentication problem is typically resolved by the use of verifiable information called *certificate*, which is issued by a trusted third party consisting of the user name and his public key. In 1984, Shamir [21] introduced the concept of *identity-based* (*ID-based*) cryptography where the public key of a user can be derived from public information that uniquely identifies the user. For example, the public key of a user can be simply his/her email address or telephone number, and hence implicitly known to all other users. A major advantage of ID-based cryptosystem is that no certificate is needed to bind user names with their public keys. The first complete ID-based encryption scheme was proposed by Boneh and Franklin in 2001 [6]. They used a bilinear map (the Weil pairing) over elliptic curves to construct the encryption/decryption scheme. After that, the bilinear pairings have been used to design numerous ID-based schemes, such as key exchange [14] and short signature [7].

In addition to the Weil pairing, there exists another bilinear map on the group of points on an elliptic curve, which is known as the Tate pairing [10]. From a computational point of view, the Tate pairing can be done approximately twice as fast as the Weil pairing as it requires half the evaluations of rational functions in Weil pairing. As our proposed algorithm improves the evaluation of a rational function, it can be similarly applied to the computation of the Tate pairing theoretically. A disadvantage of the Tate pairing is that the outcome is not a unique value, which cannot be used in many applications. This problem can be solved by performing an exponentiation on the outcome of the Tate pairing [20]. The advantage of the Weil pairing is that its definition is more comprehensible than that of the Tate pairing, which involves equivalence classes of quotient groups. For the reader to easily follow the derivation of our proposed algorithm, we introduce the Weil pairing and implement our proposed algorithm on it.

ID-based cryptosystem transparently provides security enhancement to the mobile applications without requiring the users to memorize extra public keys. For example, sending an ID-based encrypted short message is exactly the same as sending a normal short message [11] if the mobile phone number of the short message recipient is used as the public key. Therefore, the mobile user (the sender) does not need to memorize the public key of the receiver. This feature is especially desirable

for mobile applications such as bank or stock transactions. However, in the existing ID-based cryptosystem, the pairing computing has significant overhead. Therefore, efficient algorithm for ID-based cryptosystem is essential in mobile devices with limited computing power.

The original algorithm for computing Weil pairing was proposed by Miller [18]. The most important part of the algorithm is the evaluation of a rational function associated with an *m*-torsion point of the elliptic curve. In this paper, we extend the idea of point halving, which was proposed by Knudsen [15], to speed up the evaluation of a rational function. We first introduce the complete ID-based encryption scheme and provide the background about the elliptic curve group and Weil pairing used in their scheme. We also describe the original Miller's algorithm for computing Weil pairing. Then we present a new algorithm for computation of Weil pairing using the point halving technique. We actually implement the ID-based encryption schemes and compare the performance to show the advantage of our approach over a previously proposed popular solution. We then illustrate an applicable ID-based end-to-end mobile encryption system.

2. Previous Work

The first complete ID-based encryption scheme is constructed by using an associated bilinear map called Weil pairing on elliptic curves. To realize Weil pairing, we need a mathematical tool called divisor on elliptic curves. With this background, we then present the original algorithm proposed by Miller for computing the Weil pairing [18].

2.1 ID-based Encryption

Boneh and Franklin proposed the first complete ID-based encryption (IBE) scheme using a bilinear map called Weil pairing over elliptic curves. The *bilinear map* transforms a pair of elements in group G_1 and sends it to an element in group G_2 in a way that satisfies some properties. The most important property is the bilinearity that it should be linear in each entry of the pair. Assume that P and Q are two elements (e.g., points on elliptic curves) of an additive group G_1. Let $e(P, Q)$ be the element of a multiplicative group G_2, which is the pairing applied to P and Q. Then the pairing must have the following property: $e(rP, Q) = e(P, Q)^r = e(P, rQ)$, where r is an integer and rP denotes the element generated by r times of additions on P, e.g., 2P=P+P, 3P=P+P+P and so on. Weil pairing on elliptic curves is selected as the bilinear map. That is, they use the elliptic curve group (the set of point collection to be defined in Section 2.2) as G_1 and the multiplicative group of a finite field as G_2. Their ID-based encryption scheme works as follows. A trusted third party called the

private key generator (PKG) initially chooses a secretive master key s and announces the public information including elliptic curve equation, the base point P, the public key sP of the system, and other needed hash functions. Each user has the public key $K_U=Q_{ID}$ that is a point on elliptic curve corresponding to his ID and is known to all other users. The private key is generated by $K_R=sQ_{ID}$, which is obtained from the PKG. To encrypt a message M, the sender randomly chooses an integer r and sends $(U, V) = (r\text{P}, M \oplus h(e(Q_{ID}, s\text{P})^r))$ to the receiver, where h is a hash function announced by PKG in the public information and e is the Weil pairing function to be elaborated in Section 2.4. To decrypt the received cipher text (U, V), the receiver uses the private key sQ_{ID} to compute $M = V \oplus h(e(sQ_{ID}, U))$. This decryption procedure yields the correct message due to the bilinearity of the Weil pairing (i.e., $e(sQ_{ID}, U) = e(sQ_{ID}, r\text{P}) = e(Q_{ID}, s\text{P})^r$).

The standard security model for a public key encryption scheme involves indistinguishability of encryptions against fully adaptive chosen ciphertext attack (IND-CCA) [1]. A strengthened model of IND-CCA called IND-ID-CCA is the standard notion of security for ID-based encryption schemes. Boneh and Franklin's ID-based encryption scheme is IND-ID-CCA secure, i.e. their encryption scheme is indistinguishably secure against adaptively chosen ciphertext attacks, under an assumption that the *Bilinear Diffie-Hellman* (BDH) problem is hard [6]. The BDH problem is defined as follows. Given P, aP, bP, cP in G_1, find out $e(\text{P}, \text{P})^{abc}$ in G_2. Since the scheme is constructed on an elliptic curve, the security level depends on the size of the finite field. For example, an elliptic curve over a 163-bit finite field currently gives the same level of security as a 1024-bit RSA [2].

The most significant overhead in implementing the ID-based encryption scheme is the computation of Weil pairing, a bilinear pairing defined on the elliptic curve to be described next.

2.2 Elliptic Curves

Let p be a prime larger than 3. An elliptic curve over a finite field of size p denoted by $GF(p)$ can be given by an equation of the form: $y^2 = x^3 + ax + b$, where a, $b \in GF(p)$. (The equation over a finite field of size 2^n denoted by $GF(2^n)$ looks slightly different and will be given later.) The set of points on the curve is the collection of ordered pairs (x, y) with coordinates in the field such that x and y satisfy the equation defining the curve, plus an extra point O called the *infinity point*. These points form an abelian group E under a certain addition over $GF(p)$. That is,

$$E = \{(x,y) \cup \text{O} \mid (x,y) \text{ satisfies the equation } y^2 = x^3 + ax + b, x, y \in GF(p)\}.$$

The group addition operation is defined as follows: to add two points P=(x_P, y_P) and Q=(x_Q, y_Q) on the curve, we first pass the straight line through them, find out the third point (x_{P+Q}, y_{P+Q}') intersected with the curve, and then reflect the point over the x-axis to obtain point P+Q=(x_{P+Q}, y_{P+Q}), i.e., y_{P+Q}= -y_{P+Q}' (see Fig. 1).

Assume that P=(x_P, y_P) and Q=(x_Q, y_Q) are on the curve, λ is the slope of the line passing through P and Q, then the coordinates of P+Q = (x_{P+Q}, y_{P+Q}) are

$$\begin{aligned} x_{P+Q} &= \lambda^2 - x_P - x_Q \\ y_{P+Q} &= \lambda(x_P - x_{P+Q}) - y_P \end{aligned}, \text{ where } \lambda = \begin{cases} \dfrac{y_Q - y_P}{x_Q - x_P} & \text{if } P \neq Q \\ \dfrac{3x_P^{\,2} + a}{2y_P} & \text{if } P = Q \end{cases}.$$

The infinity point O plays a role as the identity element, that is, P+O = O+P = P for any point P. Each point P has a unique inverse element -P such that P+(-P)=O. For P=(x_P, y_P) in elliptic curve E over $GF(p)$, the unique additive inverse of P is defined by -P=(x_P, -y_P).

Figure 1 Group law on an elliptic curve

Another category of elliptic curves is defined over the finite field of size 2^n denoted by $GF(2^n)$. The equation defining elliptic curves over $GF(2^n)$ is of the form $y^2 + xy = x^3 + ax^2 + b$, where $a, b \in GF(2^n)$. The addition operation on points P and Q is the same as before except that $y_{P+Q} = x_{P+Q} + y_{P+Q}'$. Therefore we can obtain the addition formula as follows. Let P=(x_P, y_P), Q=(x_Q, y_Q)$\in E$. Let λ be the slope of the line passing through P and Q, and the coordinates of P+Q be (x_{P+Q}, y_{P+Q}). Then

$$\begin{aligned} x_{P+Q} &= \lambda^2 + \lambda + x_P + x_Q + a \\ y_{P+Q} &= \lambda(x_P + x_{P+Q}) + x_{P+Q} + y_P \end{aligned}, \text{ where } \lambda = \begin{cases} \dfrac{y_Q + y_P}{x_Q + x_P} & \text{if } P \neq Q \\ x_P + \dfrac{y_P}{x_P} & \text{if } P = Q \end{cases}.$$

The inverse of P=(x_P, y_P) is defined by -P=(x_P, x_P+y_P) when P is in elliptic curve E over binary field $GF(2^n)$.

For elliptic curves, the group operation is written as addition instead of multiplication. Thus the exponentiation in general multiplicative group can be appropriately referred to as the scalar multiplication in elliptic curve group. That is, we denote rP as $\underbrace{P + P + ... + P}_{r \text{ times}}$ for an integer r.

2.3 Divisor

A divisor is a useful device for keeping track of the zeros and poles[1] of rational functions [17]. A divisor provides a representation to indicate which points are zeros or poles and their orders for a rational function over the elliptic curve. A divisor D can be defined as a formal sum of points on elliptic curve group E: $D = \sum_{P \in E} n_P(P)$, where n_P is a non-zero integer that specifies the zero/pole property of point P and its respective order. Inequality $n_P > 0$ indicates that point P is a zero, and $n_P < 0$ indicates that P is a pole. For example, for P, Q, R$\in E$, D_1 = 2(P) + 3(Q) – 3(R) indicates that divisor D_1 has zeros at P and Q with order 2 and 3 respectively, and a pole at R with order 3. And D_2= 2(P) + (-2P) – 3(O) indicates that P and -2P are zeros with order 2 and 1, and O is a pole with order 3 for the divisor D_2. Note that the parenthesis is used to separate the order and the specific point. For example, (2P) indicates that 2P is a zero with order 1, while 2(P) indicates that P is a zero with order 2.

The group of divisors on E, denoted as $Div(E)$, forms an abelian group with the following addition operation. For D_1, $D_2 \in Div(E)$, if $D_1 = \sum_{P \in E} n_P(P), D_2 = \sum_{P \in E} m_P(P)$, then $D_1 + D_2 = \sum_{P \in E} n_P(P) + \sum_{P \in E} m_P(P) = \sum_{P \in E} (n_P + m_P)(P)$. For a divisor $D = \sum_{P \in E} n_P(P)$, we define $supp(D)=\{P \in E \mid n_P \neq 0\}$ as the support of divisor D, and $deg(D) = \sum_{P \in E} n_P$ as

[1] Let f be a non-zero rational function, and P$\in E$. If f(P)=0 then f is said to have a zero at P. If f is not defined at P then f is said to have a pole at P and we write f(P)=∞.

the degree of divisor D. For example, if $D_1 = 2(P) + 3(Q) – 3(R)$, $D_2 = 2(P) + (-2P) – 3(O)$, then $supp(D_1)=\{P, Q, R\}$, $supp(D_2)=\{P, -2P, O\}$ and $deg(D_1)=2+3-3=2$, $deg(D_2)= 2+1-3=0$.

From now on, we consider only the set of divisors of degree zero, denoted as $Div^0(E)$. Let f be a rational function from $K\times K$ to K, where K is a finite field. For example, $f(x,y)=\frac{3y-2x-5}{5y+3x-2}$. The evaluation of a rational function f on a point $P=(x_P, y_P)$ is defined by $f(P)=f(x_P, y_P)$ and the evaluation of f on a divisor $D=\sum_{P\in E} n_P(P)$ is defined by $f(D)=\prod_{P\in supp(D)} f(P)^{n_P}$. Define the divisor of a rational function f as $div(f)=\sum_{P\in E} n_{P,f}(P)$, where $n_{P,f}$ is the zero/pole order of point P on f. It is well known that the degree of the divisor of a rational function must be zero [17]; that is, $div(f)\in Div^0(E)$ for any rational function f. For example, let $P=(x_P, y_P)\in E$, $f(x,y)= x-x_P$, then $div(f) = div(x-x_P) = (P) + (-P) – 2(O)$. P and -P are the zeros of f because only they are on both the vertical line $x-x_P=0$ and the elliptic curve E. The infinity point O is a pole of order 2 because $div(f)\in Div^0(E)$. Then for two rational functions f_1 and f_2, we have $div(f_1) + div(f_2) = div(f_1f_2)$ and $div(f_1) – div(f_2) = div(f_1/f_2)$.

As an example, let E be the elliptic curve defined by $y^2=x^3+7x$ over $GF(13)$. We have P=(4,1), Q=(5,2)$\in E$, and P+Q=(5,11). Assume that $f(x,y)=\frac{y-x+3}{x-5}$. Since P, Q, -(P+Q)=(5,2)=Q are on the line $y-x+3=0$, $div(y-x+3) = (P) + (Q) + (-(P+Q)) – 3(O) = (P) + 2(Q) – 3(O)$. Also, $div(x-5) = (Q) + (-Q) + 2(O) = (Q) + (P+Q) – 2(O)$ because Q, -Q=(5,11)=P+Q are on the line $x-5=0$. Therefore, we have $div(f) = div(y-x+3) – div(x-5) = (P) + (Q) – (P+Q) – (O)$.

A divisor $D\in Div^0(E)$ is defined to be *principal* if $D=div(f)$ for some rational function f. The principal divisor $D=\sum_{P\in E} n_P(P)$ is characterized by $\sum_{P\in E} n_P P = O$ [17], where $\sum_{P\in E} n_P P$ denotes the sum by applying addition operation on the points in elliptic curve E. For example, let $D_3 = (P) + (-P) – 2(O)$, then D_3 satisfies $deg(D_3)=0$ and $P+(-P)–2O = P–P = O$. Therefore D_3 is principal. In fact, $D_3=div(x-x_P)$ for the function $x-x_P$.

Two divisors $D_1, D_2\in Div^0(E)$ are said to be equivalent (denoted as $D_1 \sim D_2$) if $D_1–D_2$ is principal. For any divisor $D=\sum_{R\in E} n_R(R)\in Div^0(E)$, there is a unique point $P=\sum_{R\in E} n_R R\in E$ such that $D \sim (P) – (O)$. In other words, D can be always written in

canonical form: D = (P) – (O) + *div*(*f*), where *f* is a rational function.

Now we introduce a formula for adding two divisors in canonical form, such that the result is still in canonical form. This formula provides a method of finding a rational function *f* such that div(*f*) = D for a given divisor D, and is critical for computing Weil pairing. Let D_1, $D_2 \in Div^0(E)$ be given by $D_1 = (P_1) - (O) + div(f_1)$ and $D_2 = (P_2) - (O) + div(f_2)$. Assume that $P_1+P_2=P_3$. Let $h_{P_1,P_2}(x,y) = ay + bx + c$ be the equation of the straight line passing through P_1 and P_2, and $h_{P_3}(x,y) = x + d$ be the equation of vertical line passing through P_3. (Note that if $P_1 = P_2$, $h_{P_1,P_2}(x,y)$ is the line tangent to P_1. And if $P_3 = O$, we have $h_{P_3}(x,y)=1$, a constant equation.) Then we have $div(h_{P_1,P_2}) = (P_1) + (P_2) + (-P_3) - 3(O)$ where P_1, P_2, and $-P_3$ are zeros because they are on line h_{P_1,P_2}, and $div(h_{P_3}) = (P_3) + (-P_3) - 2(O)$ where P_3, $-P_3$ are zeros because they are on line h_{P_3} (see Fig. 1). From the above discussion, the sum of divisors $D_1 + D_2$ is written as:

$$\begin{aligned} D_1 + D_2 &= (P_1) + (P_2) - 2(O) + div(f_1 f_2) \\ &= (P_3) - (O) + div(f_1 f_2) + div(h_{P_1,P_2}) - div(h_{P_3}) \\ &= (P_3) - (O) + div(f_1 f_2 h_{P_1,P_2} / h_{P_3}). \end{aligned} \quad (1)$$

Eq. (1) will be used in the computation of Weil pairing in the following subsection.

2.4 Weil Pairing

Given an elliptic curve *E* over a finite field *K*, let *m* be an integer prime to *char*(*K*), the characteristic of *K* [16]. For example, *char*(*GF*(*p*))=*p* and *char*(*GF*(2^n))=2. The Weil pairing is a function

$$e : E[m] \times E[m] \to U_m,$$

where $E[m] = \{P \mid mP = O, P \in E\}$ is called the *m*-torsion group, U_m is the group of the m^{th} roots of unity in $\overline{K}$, the algebraic closure of *K* [16].

Weil pairing $e(P, Q)$ is defined as follows. Given $P, Q \in E[m]$, there exist divisors D_P, $D_Q \in Div^0(E)$ such that $D_P \sim (P) - (O)$ and $D_Q \sim (Q) - (O)$. Here we randomly choose points T, U, and assign $D_P = (P+T) - (T)$ and $D_Q = (Q+U) - (U)$. It is easy to verify that $D_P \sim (P) - (O)$ and $D_Q \sim (Q) - (O)$. As $mP = mQ = O$, divisors mD_P and mD_Q are principal and there exist rational functions f_P , f_Q such that $div(f_P) = mD_P$ and $div(f_Q) = mD_Q$. Suppose that D_P and D_Q have disjoint supports, i.e., $supp(D_P) \cap supp(D_Q) = \phi$, then the Weil pairing of P and Q is defined as:

$$e(P,Q) = \frac{f_P(D_Q)}{f_Q(D_P)}.$$

The Weil pairing has the *bilinearity* property: for $P, Q, R \in E[m]$, we have $e(P+Q, R) = e(P, R)e(Q, R)$ and $e(P, Q+R) = e(P, Q)e(P, R)$. The first algorithm for $e(P, Q)$ computation is described as follows.

Miller's Algorithm [18]

INPUT: $P, Q \in E[m]$
OUTPUT: $e(P, Q)$

Step 1. Select random points $T, U \in E$ such that P+T, T, Q+U, U are distinct.
Let $D_P = (P+T) - (T)$ and $D_Q = (Q+U) - (U)$.

Step 2. Use an evaluation algorithm to compute $f_P(Q+U)$, $f_P(U)$, $f_Q(P+T)$ and $f_Q(T)$, where f_P and f_Q satisfy that $div(f_P) = mD_P$ and $div(f_Q) = mD_Q$.

Step 3. Compute $e(P,Q) = \frac{f_P(D_Q)}{f_Q(D_P)} = \frac{f_P(Q+U)f_Q(T)}{f_Q(P+T)f_P(U)}$.

A crucial part in Miller's algorithm is the evaluation algorithm in Step 2. The evaluation algorithm for $f_P(S)$ produces f_P such that $div(f_P) = mD_P$, and computes $f_P(x_S, y_S)$ for $S=(x_S, y_S)$. Recall that $D_P = (P+T) - (T)$. For each integer k, there exists a rational function f_k such that

$$div(f_k) = k(P+T) - k(T) - (kP) + (O).$$

If $k = m$, then $div(f_m) = m(P+T) - m(T) - (mP) + (O) = m(P+T) - m(T)$, and $f_P = f_m$. For any points R, S, let $h_{R,S}$ and h_R be linear functions, where $h_{R,S}(x, y) = 0$ is the straight line passing through R, S, and $h_R(x, y) = 0$ is the vertical line passing through R. Then we have

$$
\begin{aligned}
div(f_{k_1+k_2}) &= (k_1+k_2)(\mathrm{P}+\mathrm{T}) - (k_1+k_2)(\mathrm{T}) - ((k_1+k_2)\mathrm{P}) + (\mathrm{O}) \\
&= k_1(\mathrm{P}+\mathrm{T}) - k_1(\mathrm{T}) - (k_1\mathrm{P}) + (\mathrm{O}) \\
&\quad + k_2(\mathrm{P}+\mathrm{T}) - k_2(\mathrm{T}) - (k_2\mathrm{P}) + (\mathrm{O}) \\
&\quad + (k_1\mathrm{P}) + (k_2\mathrm{P}) + (-(k_1+k_2)\mathrm{P}) - 3(\mathrm{O}) \\
&\quad - [((k_1+k_2)\mathrm{P}) + (-(k_1+k_2)\mathrm{P}) - 2(\mathrm{O})] \\
&= div(f_{k_1}) + div(f_{k_2}) + div(h_{k_1\mathrm{P},k_2\mathrm{P}}) - div(h_{(k_1+k_2)\mathrm{P}})
\end{aligned},
$$

and hence

$$
f_{k_1+k_2} = \frac{f_{k_1} f_{k_2} h_{k_1\mathrm{P},k_2\mathrm{P}}}{h_{(k_1+k_2)\mathrm{P}}}. \qquad (2)
$$

Eq. (2) is recursive with initial conditions $f_0 = 1$ and $f_1 = \frac{h_{\mathrm{P+T}}}{h_{\mathrm{P,T}}}$ since

$$
\begin{aligned}
div(f_1) &= (\mathrm{P}+\mathrm{T}) - (\mathrm{T}) - (\mathrm{P}) + (\mathrm{O}) \\
&= (\mathrm{P}+\mathrm{T}) + (\text{-}(\mathrm{P}+\mathrm{T})) - 2(\mathrm{O}) \\
&\quad - [(\mathrm{P}) + (\mathrm{T}) + (\text{-}(\mathrm{P}+\mathrm{T})) - 3(\mathrm{O})] \\
&= div(h_{\mathrm{P+T}}) - div(h_{\mathrm{P,T}}).
\end{aligned}
$$

Based on Eq. (2), a conventional double-and-add method was proposed for evaluation of a rational function f_{P} on a given point S, where f_{P} satisfies $div(f_{\mathrm{P}}) = m(\mathrm{P+T}) - m(\mathrm{T})$. The algorithm denoted as double-and-add evaluation algorithm is described as follows.

Double-and-Add Evaluation Algorithm (Step 2, Miller's Algorithm) [3]

INPUT: the points P, T, S, and the order $m = \sum_{i=0}^{n-1} b_i 2^i$ with $b_i \in \{0,1\}$, $b_{n-1} = 1$

OUTPUT: $f_m(\mathrm{S}) = f_{\mathrm{P}}(\mathrm{S})$

$f_1 \leftarrow \frac{h_{\mathrm{P+T}}(\mathrm{S})}{h_{\mathrm{P,T}}(\mathrm{S})}$;
$f \leftarrow f_1$; $\mathrm{Z} \leftarrow \mathrm{P}$;
for $j \leftarrow n$-2, n-3, …, 0 **do**
$f \leftarrow f^2 \frac{h_{\mathrm{Z,Z}}(\mathrm{S})}{h_{2\mathrm{Z}}(\mathrm{S})}$; $\mathrm{Z} \leftarrow 2\mathrm{Z}$;
if $b_j = 1$ **then**
$f \leftarrow f_1 f \frac{h_{\mathrm{Z,P}}(\mathrm{S})}{h_{\mathrm{Z+P}}(\mathrm{S})}$; $\mathrm{Z} \leftarrow \mathrm{Z} + \mathrm{P}$;

endif

endfor

return *f*

In the next section, we propose a halve-and-add method to speed up the evaluation for Weil pairing.

3. Efficient Computation for Weil Pairing

We first introduce the point halving operation proposed by Knudsen in speeding up scalar multiplication on elliptic curve over $GF(2^n)$. The advantage of point halving relies on the fast arithmetic operations over $GF(2^n)$ in a normal basis. We then propose an efficient halve-and-add evaluation algorithm in Weil pairing computation.

3.1 Point Halving

We restrict our attention to elliptic curves E over finite field $GF(2^n)$ defined by the equation $y^2 + xy = x^3 + ax^2 + b$ where a, $b \in GF(2^n)$, $b \neq 0$. The finite field $GF(2^n)$ can be viewed as a vector space of dimension n over $GF(2)$. That is, each $c \in GF(2^n)$ can be represented as a vector $(c_{n-1}...c_1c_0)$ where $c_i \in \{0,1\}$. Let P=(x_P, y_P) be a point on E, where P $\neq$ –P. The coordinate of Q = 2P = (x_Q, y_Q) can be computed as follows:

$$\lambda = x_P + \frac{y_P}{x_P} \tag{3}$$

$$x_Q = \lambda^2 + \lambda + a \tag{4}$$

and

$$y_Q = x_P^{\,2} + x_Q(\lambda + 1) \tag{5}$$

Point halving was first proposed by Knudsen with the following operation: given Q=(x_Q, y_Q), compute P=(x_P, y_P) such that Q = 2P. Point halving provides fast computation for scalar multiplication on elliptic curve. The basic idea for halving is to solve Eq. (4) for λ, Eq. (5) for x_P, and finally Eq. (3) for y_P if needed. If G is a subgroup of odd order m in E, point doubling and point halving are automorphisms in G [15]. Therefore, given a point Q$\in G$, there is a unique point P$\in G$ such that Q = 2P. To uniquely find P, Fong et al. [9] designed a point halving computation algorithm using the *trace function* Tr : $GF(2^n) \rightarrow GF(2)$ defined by $Tr(c) = \sum_{i=0}^{n-1} c_i$, where $c = (c_{n-1}...c_1c_0) \in GF(2^n)$.

The halve-and-add method for scalar multiplication uses two kinds of point representation: the usual affine representation P=(x_P,y_P) and the λ-representation (x_P,λ_P), where $\lambda_P = x_P + y_P/x_P$ denotes the slope of the tangent line to the curve at P.

As shown in the following point halving algorithm, repeated halving can be performed directly on the λ-representation of a point. Only when a point addition is required, a conversion to affine coordinate is needed.

Point Halving Algorithm [9]

INPUT: λ-representation (x_Q, λ_Q) of Q
OUTPUT: λ-representation (x_P, λ_P) of P=(x_P, y_P), where Q = 2P

Step 1. Find a solution $\hat{\lambda}$ for equation $\lambda^2 + \lambda = x_Q + a$.

Step 2. Compute $c = x_Q(x_Q + \lambda_Q + \hat{\lambda})$.

Step 3. If $Tr(c) = 0$ then $\lambda_P \leftarrow \hat{\lambda}$, $x_P \leftarrow \sqrt{c + x_Q}$,

else $\lambda_P \leftarrow \hat{\lambda} + 1$, $x_P \leftarrow \sqrt{c}$.

Step 4. Return (x_P, λ_P).

The point halving algorithm requires one field multiplication (Step 2) and three operations: solving the quadratic equation $\lambda^2 + \lambda = x_Q + a$ (Step 1), one trace computation (Step 3), and computing a square root (Step 3). In a normal basis, the time needed for these three operations is negligible compared to the time needed for a multiplication or an inversion. An inversion can be computed using a number of multiplications. The ratio of inversion to multiplication cost is about 8 in our Pentium III platform. In the next subsection we select a normal basis and introduce arithmetic operations over it. With the normal basis we select, the square root operation at Step 3 can be significantly simplified.

3.2 Normal Basis

Recall that the binary field $GF(2^n)$ can be viewed as a vector space of dimension n over $GF(2)$. That is, there exists a set of n elements α_0, α_1, …, α_{n-1}, in $GF(2^n)$ such that each $c \in GF(2^n)$ can be written in the form $c = \sum c_i \alpha_i = (c_{n-1}...c_1c_0)$. In general, there are many bases of $GF(2^n)$. A typical one is the polynomial basis of the form $\{1, x, x^2, \ldots, x^{n-1}\}$, and a kind of special basis called *normal basis* is the set of the form $\{\beta, \beta^2, ..., \beta^{2^{n-1}}\}$. In a normal basis, a field element c on $GF(2^n)$ is represented by $c = \sum c_i \beta^{2^i} = (c_{n-1}...c_1c_0)$. The squaring of c can be obtained by

$c^2 = \sum_{i=0}^{n-1} c_i \beta^{2^{i+1}} = \sum_{i=0}^{n-1} c_{i-1} \beta^{2^i} = (c_{n-2} ... c_0 c_{n-1})$. That is, squaring of c can be accomplished by a simple left rotation on the vector representation of c. On the other hand, the square root computation is just a right rotation, i.e., $\sqrt{c} = (c_0 c_{n-1} ... c_1)$ at Step 3 in point halving algorithm. Therefore the quadratic equation $x^2 + x = c$ can be solved bitwise at Step 1 in point halving algorithm. These operations are expected to be inexpensive relative to the field multiplication or the field inversion. The field multiplication in a normal basis is more complicated, but, with optimization, it can be reduced to a series of n cyclic shifts of the two vector multiplicands. Mullin, Onyszchuk, Vanstone and Wilson [19] introduced optimal normal bases that can optimize the time complexity for field multiplication in $GF(2^n)$. The normal basis implementation we proposed [26] is the basic architecture in the halve-and-add evaluation algorithm to be described in the next subsection.

3.3 Halve-and-Add Method for Weil Pairing

Now we propose a halve-and-add method for the evaluation of rational functions used in the Miller's algorithm. The evaluation algorithm described in Section 2.4 applies the double-and-add method to compute Weil pairing. To take advantage of point halving, we propose a halve-and-add version of the evaluation algorithm.

Let the λ-representation of a point P=(x_P, y_P) be (x_P, λ_P), and the canonical form of a divisor D_P be (P) – (O) + $div(g)$, where g is a rational function. Assume that Q=2P with λ-representation (x_Q, λ_Q) corresponds to a divisor D_Q with canonical form (Q) – (O) + $div(f)$. Let $h_{P,P}(x, y) = 0$ be the equation of the tangent line at P and $h_{2P}(x, y) = 0$ be the vertical line through Q=2P. By the addition formula of two divisors with canonical form (see Eq. (1)), we have

$$\begin{aligned} D_P + D_P &= (2P) - (O) + div(g^2 \frac{h_{P,P}}{h_{2P}}) \\ &= (Q) - (O) + div(f). \end{aligned} \quad (6)$$

We also have

$$h_{P,P}(x, y) = y + y_P + \lambda_P (x + x_P) = y + \lambda_P x + x_P^2 \text{ and } h_{2P}(x, y) = x + x_Q . \quad (7)$$

From Eqs. (6) and (7), we have $f = g^2 \frac{h_{P,P}}{h_{2P}} = g^2 \frac{y + \lambda_P x + x_P^2}{x + x_Q}$ and thus

$$g = \sqrt{f \frac{x + x_{\mathrm{Q}}}{y + \lambda_{\mathrm{P}} x + x_{\mathrm{P}}{}^2}}\,.$$

Denote the 1/2-representation of m as $(m)_{1/2}$ = ($\hat{b}_{n-1} \ldots \hat{b}_0$) such that $m = \sum_{i=0}^{n-1} \hat{b}_i \left(\frac{1}{2^i} \right) \bmod r$, where r is the order of point P. In order to apply the halve-and-add operation in the evaluation of f, we first determine $(m)_{1/2}$. A simple translation was described in [15]. For Weil pairing computation, integer m is not only the scalar in evaluating f but also the order of the point P, i.e., mP=O. To evaluate the rational function f_m, we first evaluate $f_{m\text{-}1}$ by using halve-and-add method, and then obtain f_m from $f_{m\text{-}1}$. This is because $(m)_{1/2}$ is always the zero string (00…0) after translation and we can not evaluate f_m by using the halve-and-add method directly. The translation of $(m\text{-}1)_{1/2}$ in our algorithm is given as follows. Let $n = \lceil log_2 m \rceil$, and $2^{n-1}(m\text{-}1) \bmod m = \sum_{i=0}^{n-1} c_i 2^i = (c_{n-1} \ldots c_0)$. Then $(m\text{-}1)_{1/2}$ = ($\hat{b}_{n-1} \ldots \hat{b}_0$), where $\hat{b}_i = c_{n-1-i}$ for i = 0, …, n-1. For example, let m = 25 and $n = \lceil log_2 25 \rceil = 5$, we can compute $2^4 \times (25\text{-}1) \bmod 25 = 9$ and represent it as (01001). Thus the 1/2-representation of 24 is (10010).

Now we compute Step 2 of the Miller's algorithm by the following halve-and-add evaluation algorithm.

Halve-and-Add Evaluation Algorithm

INPUT: points P, T, S, where P is given by λ-representation (x_{P}, λ_{P}), and the order m

OUTPUT: $f_m(\mathrm{S}) = f_{\mathrm{P}}(\mathrm{S})$

Find the 1/2-representatin $(m\text{-}1)_{1/2}$.= ($\hat{b}_{n-1} \ldots \hat{b}_0$) with $\hat{b}_i \in \{0,1\}$, $\hat{b}_{n-1}$=1;

$f_1 \leftarrow \frac{h_{\mathrm{P+T}}(\mathrm{S})}{h_{\mathrm{P,T}}(\mathrm{S})}$;

$f \leftarrow f_1$; Z $\leftarrow$ P;

for $j \leftarrow n\text{-}1, n\text{-}2, \ldots, 0$ **do**

$$f \leftarrow \sqrt{f \frac{x_{\mathrm{S}} + x_{\mathrm{Z}}}{y_{\mathrm{S}} + \lambda_{\mathrm{Z}/2} x_{\mathrm{S}} + x_{\mathrm{Z}/2}{}^2}}\,; \quad \mathrm{Z} \leftarrow \frac{1}{2}\mathrm{Z};$$

if $\hat{b}_j = 1$ **then**

$$f \leftarrow f_1 f \frac{h_{Z,P}(S)}{h_{Z+P}(S)}; \; Z \leftarrow Z + P;$$

endif

endfor

$$f \leftarrow f_1 f \frac{h_{Z,P}(S)}{h_{Z+P}(S)}; \; Z \leftarrow Z + P;$$

return f

In our halve-and-add evaluation algorithm, the halving stage requires 1 inversion, 3 multiplications, 1 squaring, and 1 square root computing, and has an advantage over the doubling. A detailed comparison will be given in the next section.

4. Performance Evaluation

In this section we estimate the saved operations in our halve-and-add evaluation algorithm compared with the double-and-add evaluation algorithm. When we consider the arithmetic operations in a normal basis, the time saved by using halving instead of doubling is significant. In affine coordinates, both elliptic doubling and addition for scalar multiplication require 1 inversion, 2 multiplications, and 1 squaring. In the λ-representation, halving stage for scalar multiplication requires 1 multiplication and three extra operations: solving the quadratic equation, trace computation, and square root computation. The addition stage requires an extra multiplication for the recovery of y-coordinate in the λ-representation. Let the order of the Weil pairing m be represented in binary format by a bit string of length n with k non-zero entries, obviously $n \geq k$. Note that our halve-and-add method requires the 1/2-representation of m-1 to apply halving and addition. By the translation of $(m\text{-}1)_{1/2}$, $(m\text{-}1)_{1/2}$ is a bit string of length n with k-1 non-zero entries. Since we need an extra addition in the final step to obtain mP from $(m\text{-}1)$P, the total addition in halve-and-add method is still k. The operations needed for the scalar multiplication are listed in Table 1.

Table 1 Arithmetic Operations for Scalar Multiplication ($n \geq k$)

Operation	**Double-and-Add**	**Halve-and-Add**
Inversion	$n + k$	k
Multiplication	$2n + 2k$	$n + 3k$
Squaring	$n + k$	k
Solving $\hat{\lambda}^2 + \hat{\lambda} = x_Q + a$	0	n
Square root	0	n
Trace computing	0	n

In affine coordinates, the doubling stage requires 2 inversions (one for the slope of $h_{Z,Z}(S)$, another for $h_{2Z}(S)$), 4 multiplications, and 1 squaring. Our halving stage requires 1 inversion, 3 multiplications, 1 squiring, and 1 square root computing in the λ-representation. The addition in our halve-and-add method requires two extra multiplications for the recovery of y-coordinate. The operations needed for the evaluation of a rational function in Weil pairing are listed in Table 2.

Table 2 Arithmetic Operations for Rational Function Evaluation ($n \geq k$)

Operation	**Double-and-Add**	**Halve-and-Add**
Inversion	$2n + 2k$	$n + 2k$
Multiplication	$4n + 5k$	$3n + 7k$
Squaring	n	n
Square root	0	n

As shown in Tables 1 and 2, by using halvings, we can save $2n$ inversions, $2n$-$3k$ (in general, $n \geq 2k$ as shown in Table 3) multiplications and n squarings at the cost of solving n quadratic equation, $2n$ square roots, and n trace computing. Note that, in a normal basis, the time needed to calculate the quadratic equation, square root, and the trace is negligible compared with the time needed to compute a multiplication or an inversion.

To investigate our improvement in computing the Weil pairing, we implement the Boneh-Franklin's ID-based encryption (IBE) scheme over the NIST recommended curves [8] on a 700MHz Intel Pentium III. Their scheme requires one pairing operation for both encryption and decryption. The recent IBE schemes proposed by Boneh-Boyen [4][5] and Waters [25] pre-compute one pairing operation before

encryption, thus require no pairing for encryption but use two pairings for decryption. Their contributions focus on constructing secure provable IBE schemes in different security models such as selective-ID model and standard model without random oracle. As our algorithm improves the computation of pairing which is primitive in IBE schemes, we can implement these new schemes in the future work. Our implementation is programmed in C and uses the free GNU Multiple Precision (GMP) arithmetic library to deal with the big number operations. The traditional double-and-add method and our halve-and-add method are both implemented for computation of Weil pairing in the ID-based encryption scheme over NIST recommended curves of different strength. Curves B-163, B-233 and B409 have the same form: $y^2 + xy = x^3 + x^2 + b$ over binary fields $GF(2^{163})$, $GF(2^{233})$ and $GF(2^{409})$, respectively. The orders of the Weil pairing m chosen in these curves are listed in Table 3. The representation of elements in the binary field is over a normal basis [26]. The size of message encrypted in our implementation is 160 ASCII characters. Table 4 lists the execution times in the ID-based encryption scheme using double-and-add method and halve-and-add method, and shows the improvements. The Weil pairing is the primitive operation for both encryption and decryption in the ID-based encryption scheme. Therefore, the efficient computation for Weil pairing improves both encryption and decryption.

Table 3 The Orders of Weil Pairing in the NIST Curves

NIST Curve	m (order of Weil pairing in decimal)	n (length of $(m)_2$)	k (weight of $(m)_2$)
B-163	5846006549323611672814742442876390689256843201587	163	41
B-233	6901746346790563787434755862277025555839812737345013555379383634485463	233	59
B-409	6610559687902485989519153080327710398284046829642812192846487983041577482737480520814372376217911096597986728836656567526771	409	103

Table 4 Execution Times (in msec) in ID-based Encryption Schemes for the NIST Curves

		Double-and-Add	Halve-and-Add	Improvement
B-163	Weil Pairing Evaluation	10.76	6.95	35 %
	Encryption	16.72	12.45	26 %
	Decryption	12.95	8.37	35 %
B-233	Weil Pairing Evaluation	41.56	30.48	27 %
	Encryption	76.28	50.13	34 %
	Decryption	46.75	37.48	20 %
B-409	Weil Pairing Evaluation	126.48	91.35	28 %
	Encryption	198.43	135.97	31 %
	Decryption	150.25	110.64	26 %

Our method reduces a number of inversions and multiplications which are expensive in computing the Weil pairing. Overall a 20~35% improvement in encryption/decryption has been accomplished.

5. ID-based End-to-End Mobile Encryption System

End-to-end security mechanisms used in mobile services are typically based on public-key cryptosystem. Under traditional public-key cryptosystem, the sender has to request the receiver's public key and verify its validity before encrypting a message. The sender can not communicate with the receiver to request the public-key when the receiver is off-line. On the other hand, in an ID-based cryptosystem, the sender can use the receiver's ID (i.e., the telephone number) as his public key without any request and verification. Thus, even if the receiver's device is power-off, the sender can still send an encrypted short message (which will be queued in the Short Message Service Center, and is forwarded to the receiver later when it is power-on) to achieve the end-to-end security. Based on the algorithm proposed in Section 3, an efficient ID-based end-to-end encryption scheme for mobile services is illustrated in Fig. 2. The PKG (Fig. 2 (1)) constructs the ID-based cryptosystem and uses, for example, the phone number as the ID (Fig. 2 (2)). Every mobile user involved in the ID-base cryptosystem is given a SIM card (Fig. 2 (3)) at the subscription time. The ID (phone number) and its corresponding private key K_R are loaded in the SIM card by the network operator. The mobile equipment contains two security modules: ID-based encryption module (Fig. 2 (4)) and ID-based decryption module (Fig. 2 (5)). When a mobile user Alice (the sender; (Fig. 2 (6))) wants to encrypt a message to Bob

(the receiver), she uses Bob's phone number (Fig. 2 (7)) as the public key and encrypts the message through the ID-based encryption module. Once Bob receives the cipher (the encrypted message), he uses the private key K_R stored in the SIM card to decrypt the cipher through the ID-based decryption module and obtain the original message.

Figure 2 ID-based End-to-end Encryption System

In some applications, the sender may choose an arbitrary string as the public key instead of the receiver's phone number. For instance, an account number can be used as the public key to encrypt the transaction information. While receiving the account number and the encrypted message, the receiver authenticates himself to the network operator and requests for corresponding private key to the account number. Then the network operator extracts the correct private key and sends it securely to the receiver. The dynamic private key extraction provides flexible security for end-to-end network services.

6. Conclusion

This paper proposed an efficient ID-based cryptography scheme for end-to-end mobile security system. We proposed a fast method for computing the Weil pairing using point halving. With the λ-representation in a normal basis, significant improvement in terms of time saving has been demonstrated in computing Weil

pairing. Our study indicates that this new approach significantly outperforms a well-known, previously proposed ID-based solution. To sum up, our contribution is twofold: firstly, we are the first to apply point halving algorithm to the ID-based scheme; secondly, we proposed an efficient approach to compute the point halving algorithm. By reducing the computation complexity, our approach provides an appropriate ID-based encryption solution for mobile services where the mobile terminals have limited computing power.

Appendix

The following notation is used throughout this paper.

K	finite field
$char(K)$	characteristic of finite field K
$GF(p)$	finite field of size p, p is a prime larger than 3
$GF(2^n)$	finite field of size 2^n
i, j, k, m, n, r, s	integer
x, y, a, b, c	element of finite field
λ	element of finite field indicating the slope of a line
α, β	element of finite field $GF(2^n)$
E	group of points on elliptic curve
P, Q, R, S, T, U	point on elliptic curve
x_P, y_P	coordinate of point P=(x_P, y_P)
O	point at infinite in elliptic curve group
$Div(E)$	group of divisors on elliptic curve
$Div^0(E)$	group of divisors on elliptic curve of degree zero
D	divisor on elliptic curve
$supp$(D)	set of supporting points of divisor D
deg(D)	degree of divisor D
f, g, h	rational function over elliptic curve
$div(f)$	divisor of rational function f
e(P, Q)	Weil pairing of points P and Q
$E[m]$	group of m-torsion points on elliptic curve
U_m	group of m^{th} roots of unity in a finite field
$Tr(c)$	trace of element c in a finite field

References

[1] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, “Relations among notions of security for public-key encryption schemes”, *Advances in Cryptology-CRYPTO'98*, pp.

26-45.

[2] I. F. Blake, G. Seroussi and N. P. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, Cambridge, (1999).

[3] I. Blake, K. Murty and G. Xu, "Refinements of Miller's Algorithm for Computing Weil/Tate Pairing", to appear in *Journal of Algorithms*.

[4] D. Boneh and X. Boyen, "Efficient Selective-ID Secure Identity Based Encryption Without Random Oracle", *Advances in Cryptology-EUROCRYPTO'04*, pp. 223-238.

[5] D. Boneh and X. Boyen, "Secure Identity Based Encryption Without Random Oracles", *Advances in Cryptology-CRYPTO'04*, pp. 443–459.

[6] D. Boneh and M. Franklin, "Identity-based Encryption from the Weil Pairing", *Advances in Cryptology-CRYPTO'01*, pp. 213–239.

[7] D. Boneh, B. Lynn and H. Shacham, "Short signatures from the Weil pairing", *Advances in Cryptology-ASIACRYPTO'01*, pp. 514–532.

[8] FIPS 186-2, Digital Signature Standard (DSS), Federal Information Processing Standards Publication 186-2, NIST 2000.

[9] K. Fong, D. Hankerson, J. Lopez and A. Menezes, "Field Inversion and Point Halving Revisited", *IEEE Trans. on Computers*, vol. 53, No. 8, 2004, pp. 1047-1059.

[10] G. Frey, M. Muller, and H.G. Ruck, "The Tate Pairing and the Discrete Logarithm Applied to Elliptic Curve Cryptosystems", *IEEE Trans. on Information Theory*, vol. 45, No 5, 1999, pp. 1717-1719.

[11] H.-N. Hung, Y.-B. Lin, M.-K. Lu, and N.-F. Peng, "A Statistic Approach for Deriving the Short Message Transmission Delay Distributions", *IEEE Trans. on Wireless Communications*, vol. 3, No. 6, 2004.

[12] Y.-B. Lin and I. Chlamtac, *Wireless and Mobile Network Architectures*, John Wiley and Sons, 2001.

[13] Y.-B. Lin, M.-F. Chen, and H. C.-H. Rao, "Potential Fraudulent Usage in Mobile Telecommunications Networks", *IEEE Trans. on Mobile Computing*, vol. 1, No. 2, 2002, pp. 123-131.

[14] A. Joux, "A One Round Protocol for Tripartite Diffie-Helman", *Algorithm Number Theory Symposium*, vol. 1838, Springer-Verlag Heidelberg, 2000, pp. 385–393.

[15] E. Knudsen, "Elliptic Scalar Multiplication Using Point Halving", *Advances in Cryptology-ASIACRYPTO'99*, pp. 135-149.

[16] R. Lidl and H. Niederreiter, *Finite Fields*, Cambridge University Press.

[17] A. J. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers.

[18] V. Miller, "Short Programs for Functions on Curves", Unpublished Manuscript, 1986.

[19] R. Mullin, I. Onyszchuk, S. Vanstone and R. Wilson, "Optimal normal bases in $GF(p^n)$", *Discrete Applied Mathematics*, vol. 22, 1988, pp. 149-161.

[20] M. Scott, The Tate Pairing. Available from www.computing.dcu.ie/~mike/tate.html.

[21] A. Shamir, “Identity-based Cryptosystems and Signature Schemes”, *Advances in Cryptology–CRYPTO’84*, pp. 47-53.

[22] J. H. Silverman, *The Arithmetic of Elliptic Curves*, Graduate Texts in Mathematics, 106, Springer-Verlag, 1986.

[23] W. Stallings, *Cryptography and Network Security*, Prentice Hall, 1999.

[24] L. C. Washington, *Elliptic Curves: Number Theory and Cryptography*, Chapman & Hall/CRC, 2003.

[25] B. R. Waters, “Efficient Identity-Based Encryption Without Random Oracles”, *Advances in Cryptology-EUROCRYPTO’05*, pp. 114-127.

[26] S.-H. Yang, J.-S. Hwu, K.-C. Huang, and R.-J. Chen, "Performance Analysis for Arithmetic in Finite Field $GF(p^n)$", *Proceedings of the 11th National Conference on Information Security*, Taiwan, 2001, pp. 185-192.

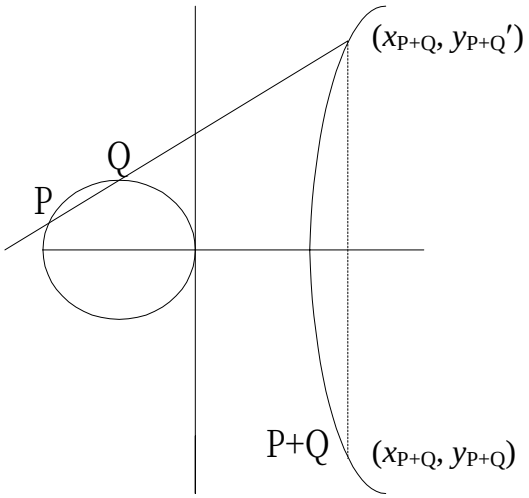

Figure 1 Group law on an elliptic curve

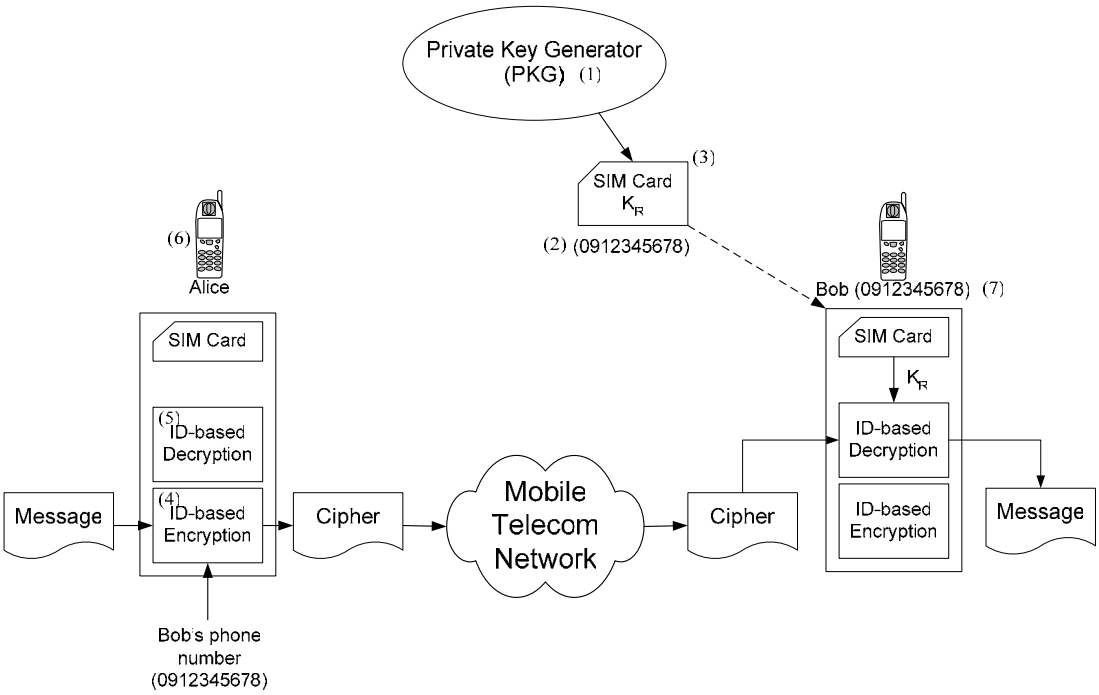

Figure 2 ID-based End-to-end Encryption System