Int. Computer Symposium, Dec. 15-17, 2004, Taipei, Taiwan.

# Efficient Computation of the Weil Pairing in ID-based Cryptosystems

**Jing-Shyang Hwu**
Department of Computer Science &
Information Engineering
National Chiao Tung University
jshwu@csie.nctu.edu.tw

**Rong-Jaye Chen**
Department of Computer Science &
Information Engineering
National Chiao Tung University
rjchen@csie.nctu.edu.tw

**Huei-Shyong Lue**
Department of Computer Science &
Information Engineering
Yuanpei Institute of Science and
Technology
hslue@mail.yust.edu.tw

**Jer-Shyong Lin**
Department of Information
Management
Yuanpei Institute of Science and
Technology
linjs@mail.yust.edu.tw

***Abstract***- *Although identity-based (ID-based) cryptography has a number of advantages over conventional public key methods; the computational cost is significantly greater. The dominant part of this cost is the Weil pairing. In this paper, we propose an efficient algorithm for computing the Weil pairing using point halving.*

**Keywords:** Weil pairing, ID-based cryptosystem, Elliptic curve.

## 1. Introduction

In 1984, Shamir [9] first invented the concept of identity-based (ID-based) cryptography, which addresses the authenticity problem of public keys in a different way. His central idea is that the public key of a user is simply their identity and hence implicitly known to all other users. More precisely, the public key of a user can be derived from public information that uniquely identifies the user. Alice@hotmail.com, for instance, could be treated as Alice's identity (ID) and used as her public key. The advantage of an ID-based cryptosystem is that no certificate is needed to bind user names and their public keys. Some practical ID-based signature schemes (IBS) have been devised since 1984, but a fully satisfying ID-based encryption scheme (IBE) was first proposed by Boneh and Franklin [4] in 2001. They use a bilinear map (the Weil pairing) over supersingular elliptic curve to construct the encryption/decryption scheme. After that, the bilinear pairings have been used to design numerous identity based schemes, such as key exchange [7], short signature [5], and many others. Compared with other arithmetic in public key cryptography, the pairing computing has significant overhead. Barreto, Kim, Lynn, and Scott [1] and Galbraith, Harrison and Soldera [6] focus on another bilinear pairing, called the Tate pairing, and they propose methods for speeding up the computation. In this paper, we extend the idea of the point halving, which was proposed by Knudsen [8], to speed up the computation of the Weil pairing.

This paper is organized as follows. Section 2 provides the background about divisors and Weil pairing on elliptic curves. The Miller's algorithm for computing Weil pairing is also described. Section 3 presents a detailed version of our algorithm for computation of Weil pairing using the point halving skill. The performance analysis compared with original Miller's algorithm is provided in section 4, and section 5 concludes the paper.

## 2. Background

### 2.1. Divisors

Given an elliptic curve E over finite field K, a divisor D is a formal sum of points on E(K)

$$D = \sum_{P \in E} n_p(P)$$

The group of divisors of E, denoted Div(E), is the free abelian group generated by the points of E, where addition is given by

$$\sum_{P \in E} n_P(P) + \sum_{P \in E} m_P(P) = \sum_{P \in E} (n_P + m_P)(P) \cdot$$

The support of a divisor $D = \sum_{P \in E} n_p(P) \in Div(E)$ is given by the set of points

$$\text{supp}(D) = \{P \in E \mid n_P \neq 0\}\,.$$

This work was supported by the National Science Council, R.O.C. , under contract no. NSC 93-2213-E-009-010-.

Further, its degree deg(D) is defined by

$$\deg(D) = \sum_{P \in E} n_p$$

It is easily verified that the divisors of degree 0, denoted $\mathrm{Div}^0(E)$, form a subgroup of Div(E). Since the number of zeros and poles of a non-zero rational function $f \in \overline{K}(E)^*$ is finite, we can define the divisor of a function *f*, denoted div(*f*), as

$$div(f) = \sum_{P \in E} ord_P(f)(P)$$

A divisor D∈Div(E) is called principal if D=div(*f*) for some ration function *f*. Further, two divisors $D_1$, $D_2$ ∈Div(E) are said to be equivalent, denoted $D_1 \sim D_2$, if $D_1 - D_2$ is principal. A characterization of principal divisors is:

$$D = \sum_{P \in E} n_p(P) \in Div^0(E) \text{ is principal}$$

iff $\sum_{P \in E} n_p P = O$ where *O* is the point at infinity.

We next describe how to evaluate a rational function *f* ∈ K(E) in a divisor $D = \sum_{P \in E} n_p(P)$ that satisfies supp(div(*f*)) ∩ supp(D) = ϕ. The evaluation of *f* in D is given by $f(D) = \prod_{P \in \sup(D)} f(P)^{n_P}$

Recall that for any degree zero divisor D∈$\mathrm{Div}^0$(E), there is a unique point P ∈ E such that D ~ (P) – (O). In other words, D can be written in what we call canonical form:

$$D = (P) - (O) + div(f),$$

where *f* is a rational function. Now we give a formula for adding two divisors in canonical form, such that the result is in canonical form as well. The formula provides a method of finding a rational function *f* such that div(*f*) = D for a given divisor D.

Let $D_1, D_2 \in \mathrm{Div}^0(E)$ be given by

$$D_1 = (P_1) - (O) + div(f_1),$$

$$D_2 = (P_2) - (O) + div(f_2).$$

Let $P_1 + P_2 = P_3$, and let $l: l_1 y + l_2 x + l_3 = 0$ be the equation of the line through $P_1$ and $P_2$, $v: x + v_1 = 0$ be the vertical line through $P_3$. If $P_1 = P_2$ then *l* is the line tangent to $P_1$, and if $P_3$ = O then take *v* = 1. Then

$$div(l) = (P_1) + (P_2) + (-P_3) - 3(O),$$

$$div(v) = (P_3) + (-P_3) - 2(O)$$

Now we can write the sum of divisors $D_1 + D_2$ as:

$$\begin{aligned} D_1 + D_2 &= (P_1) + (P_2) - 2(O) + div(f_1 f_2) \\ &= (P_3) - (O) + div(l) - div(v) + div(f_1 f_2) \\ &= (P_3) - (O) + div(f_1 f_2 f_3) \end{aligned}$$

where $f_3 = l/v$. .

## 2.2. Weil Pairing

Let *m* be an integer which is prime to p = char(K). The Weil pairing is a function:

$$e_m : E[m] \times E[m] \to \mu_m,$$

where $E[m] = \{P \in E(\overline{K}) : mP = O\}$ called the *m*-torsion group, $\mu_m$ is the group of $m^{\text{th}}$ roots of unity in $\overline{K}$ .

Given P, Q∈E[*m*], there exist $D_P$, $D_Q \in \mathrm{Div}^0(E)$ such that

$$D_P \sim (P) - (O) \text{ and } D_Q \sim (Q) - (O)$$

As divisors $mD_P$ and $mD_Q$ are principal, there exist rational functions $f_P$ , $f_Q$ such that div($f_P$) = $mD_P$ , div($f_Q$) = $mD_Q$. Suppose that $D_P$ and $D_Q$ have disjoint supports, and then the Weil pairing of P and Q can be computed by:

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

**Algorithm 1** Weil pairing

(Miller's probabilistic algorithm)

INPUT: P, Q∈E[*m*], *m* is prime to char(K)

OUTPUT: $e_m$(P, Q)

```
1   Pick random point T, U∈E(K)
    such that P + T, T, Q+U, U are distinct
2.  Compute $f_P$, $f_Q$ such that
    div($f_P$) = $m(P+T) - m(T)$,
    div($f_Q$) = $m(Q+U) - m(U)$
3.  Evaluate $\frac{f_P(Q+U)\, f_Q(T)}{f_Q(P+T)\, f_P(U)}$
```

An important part of computing the Weil pairing is the evaluation of $f_P$(R) for each point R in the support of $D_Q$. Recall that $D_P$ = (P + T) – (T). Then for each integer *k*, there is a rational function $f_k$ such that

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

Let *k* = *m*,

$$\begin{aligned} &div(f_m) \\ &= m(P+T) - m(T) - (mP) + (O), \\ &= m(P+T) - m(T) \end{aligned}$$

we have $f_P = f_m$. For any points R, S, let $h_{R,S}$ and $h_R$ be linear functions, where $h_{R,S} = 0$ is the line passing through R, S, and $h_R = 0$ is the vertical line passing through R. Then we have

$$\begin{aligned} div(f_{k_1+k_2}) &= (k_1+k_2)(P+T) - (k_1+k_2)(T) - ((k_1+k_2)P) + (O) \\ &= k_1(P+T) - k_1(T) - (k_1P) + (O) \\ &\quad + k_2(P+T) - k_2(T) - (k_2P) + (O) \\ &\quad + (k_1P) + (k_2P) + (-(k_1+k_2)P) - 3(O) \\ &\quad - [((k_1+k_2)P) + (-(k_1+k_2)P) - 2(O)] \\ &= div(f_{k_1}) + div(f_{k_2}) + div(h_{k_1P,k_2P}) - div(h_{(k_1+k_2)P}) \end{aligned}$$

, and hence $f_{k_1+k_2} = \dfrac{f_{k_1} f_{k_2} h_{k_1P,k_2P}}{h_{(k_1+k_2)P}}$.

This is a recursive equation with initial conditions $f_0 = 1$ and $f_1 = \dfrac{h_{P+T}}{h_{P,T}}$

**Algorithm 2** Evaluation of $f_P$ on a point S [3]

```
INPUT: $m = \sum_{i=0}^{t} b_i 2^i$ with $b_i \in \{0,1\}$
        and $b_t = 1$ , and a point S
OUTPUT: $f_m(S) = f_P(S)$
  $f \leftarrow f_1(S)$; Z ← P;
  For $j$ ← t-1, t-2, …, 0 do
    $f \leftarrow f^2 \dfrac{h_{Z,Z}(S)}{h_{2Z}(S)}$ ; Z ← 2Z;
        If $b_j = 1$ then
          $f \leftarrow f_1 f \dfrac{h_{Z,P}(S)}{h_{Z+P}(S)}$ ; Z ← Z + P;
        Endif
  Endfor
  Return $f$
```

This is a conventional double-and-add method for evaluation of rational function $f_P$ on a given point S. In the next section, we will propose a halve-and-add method to speed up the evaluation and hence have an efficient computation of the Weil pairing.

## 3. The Algorithm

We restrict our attention to elliptic curves E over Galois field $F_{2^n}$ defined by the equation: $y^2 + xy = x^3 + ax^2 + b$ , where $a, b \in F_{2^n}, b \neq 0$. Let P = (*x*, *y*) be a point on E with P ≠ –P. The coordinate of Q = 2P = (*u*, *v*) can be computed as follows:

$$\lambda = x + y/x \qquad (1)$$

$$u = \lambda^2 + \lambda + a \qquad (2)$$

$$v = x^2 + u(\lambda + 1) \qquad (3)$$

Point halving was first proposed by Knudsen [8] with the following operation: given Q = (*u*, *v*), compute P = (*x*, *y*) such that Q = 2P. It provides a fast method for scalar multiplication on elliptic curve. The basic idea for halving is to solve (2) for λ, (3) for *x*, and finally (1) for *y*. When G is a subgroup of odd order *m* in E, point doubling and point halving are automorphisms in G (see [8]). Therefore, given a point Q∈G, there is a unique point P∈G such that Q = 2P. To uniquely find P, the trace function plays a central role in the algorithm for point halving.

The trace function $Tr : F_{2^n} \to F_{2^n}$ is defined by $Tr(c) = c + c^2 + c^{2^2} + \ldots + c^{2^{n-1}}$ . Given Q = (*u*, *v*), point halving seeks the unique point P = (*x*, *y*) such that Q = 2P. The first step is to find $\hat{\lambda}$ by solving the equation

$$\lambda^2 + \lambda = u + a \qquad (4)$$

It is easily verified that $\lambda \in \{\hat{\lambda}, \hat{\lambda}+1\}$ and $\lambda = \hat{\lambda}$ if and only if $Tr(v + u\hat{\lambda}) = 0$. Hence λ can be identified, and then (3) is solved for the unique root *x*. Finally, if needed, $y = x(x+\lambda)$ can be recovered with one field multiplication.

Let the λ-representation of a point Q = (*u*, *v*) be (*u*, $\lambda_Q$), where $\lambda_Q = u + v/u$. Given the λ-representation of Q as the input to point halving, we may compute $t = v + u\hat{\lambda} = u(u + \lambda_Q + \hat{\lambda})$ without converting to affine coordinate. So in the point multiplication, 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.

**Algorithm 3** Point halving

```
INPUT: λ-representation $(u, \lambda_Q)$ of Q∈G
OUTPUT: λ-representation $(x, \lambda_P)$ of P = (x, y)
        ∈G, where Q = 2P
  1   Find a solution $\hat{\lambda}$ of $\lambda^2 + \lambda = u + a$ .
  2   Compute $t = u(u + \lambda_Q + \hat{\lambda})$.
  3   If Tr(t) = 0, then $\lambda_P \leftarrow \hat{\lambda}$, $x \leftarrow \sqrt{t+u}$
      else $\lambda_P \leftarrow \hat{\lambda}+1$, $x \leftarrow \sqrt{t}$
  4   Return $(x, \lambda_P)$.
```

The point halving algorithm requires a field multiplication and three main steps:

1. Solving the quadratic equation $\lambda^2 + \lambda = u + a$

2. Computing the trace of t

3. Computing a square root

In a normal basis, a field element on $F_{2^n}$ is represented in terms of a basis of the form { $\beta, \beta^2, ..., \beta^{2^{n-1}}$ }. Given a field element $c = \sum c_i \beta^{2^i} = (c_{n-1}, ..., c_0)$, the squaring is a left rotation, i.e. $c^2 = (c_{n-2}, ..., c_0, c_{n-1})$. Therefore the quadratic equation $x^2 + x = c$ can be solved bitwise. The square root computation is a right rotation, i.e. $\sqrt{c} = (c_0, c_{n-1} ..., c_1)$. These operations are expected to be inexpensive relative to field multiplication. A detailed comparison will be given in the next section.

Let the λ-representation of a point P = (*x*, *y*) be (*x*, $\lambda_P$), and the canonical form of a divisor $D_P$ be (P) – (O) + div(*g*), where *g* is a rational function. We have

$$D_P + D_P = (2P) - (O) + \text{div}(\frac{g^2 l}{v}).$$

Assume Q = 2P with λ-representation (*u*, $\lambda_Q$) corresponding to a divisor $D_Q$ with canonical form (Q) – (O) + div(*f*); then

$$l = Y + \lambda_P X + x^2,$$

$$v = X + u,$$

$$f = \frac{g^2 l}{v} = g^2 \frac{Y + \lambda_P X + x^2}{X + u},$$

and we have

$$g = \sqrt{f \frac{X + u}{Y + \lambda_P X + x^2}}$$

Apply the halving operation to the evaluation of *f* on a point S; we have an efficient algorithm for Weil pairing computation.

**Algorithm 4** Evaluation of $f_P$ on a point S using halving

```
INPUT: $m = \sum_{i=0}^{t} b_i 2^i$ with $b_i \in \{0,1\}$ and $b_t = 1$ ,
       and a point S = ($X_S$, $Y_S$),
       λ-representation of P = ($x$, $\lambda_P$)
OUTPUT: $f_m(S) = f_P(S)$
  Translate $m$-1 to be the form $\sum_{i=0}^{t} \hat{b}_i \frac{1}{2^i}$
  $f \leftarrow f_1(S)$; Z ← P ;
  For $j$ ← t-1, t-2, …, 0 do
    $f \leftarrow \sqrt{f \frac{X_S + x_Z}{Y_S + \lambda_Z X_S + x_{Z/2}^{\ 2}}}$ ; Z ← $\frac{1}{2}$ Z;
      If $\hat{b}_j$ = 1 then
        $f \leftarrow f_1 f \frac{h_{Z,P}(S)}{h_{Z+P}(S)}$ ; Z ← Z + P;
      Endif
  Endfor
  $f \leftarrow f_1 f \frac{h_{Z,P}(S)}{h_{Z+P}(S)}$ ; Z ← Z + P;
  Return $f$
```

## 4. Performance Comparison

In this section we estimate the saved operations in our algorithm compared with the original Miller's algorithm. When we consider the arithmetic operations in normal basis, the time saved by using halving instead of doubling is significant. In affine coordinates, both elliptic doubling and addition require 1 inversion, 2 multiplications and 1 squaring. While representing with λ-representation, we can save 1 inversion and 1 multiplication in point halving. But one additional multiplication is needed to recover the y-coordinate while performing addition. If the order of the Weil pairing *m* is represented by a bit string of length *n* with *k* non-zero entries, the operations needed for the scalar multiplication are:

| Operation | Double-and-Add | Halve-and-Add |
|---|---|---|
| Inversions | $n + k$ | $k$ |
| Multiplications | $2n + 2k$ | $n + 3k$ |
| Squarings | $n + k$ | $n + k$ |
| Solving $\lambda^2 + \lambda = u + a$ | 0 | $n$ |
| Square roots | 0 | $n$ |
| Trace computing | 0 | $n$ |

and the operations needed for the evaluation of rational functions in the given point are:

| **Operation** | **Double-and-Add** | **Halve-and-Add** |
|---|---|---|
| Inversions | $2n + 2k$ | $n + k$ |
| Multiplications | $4n + 5k$ | $3n + 4k$ |
| Squarings | $n$ | $2n$ |
| Square roots | 0 | $n$ |

Thus, by using point halving, we can save $2n+k$ inversions, $2n$ multiplications with additional cost in solving $n$ quadratic equation, $n$ squaring, $2n$ square roots and $n$ trace computing. However, in a normal basis, the time needed to calculate the quadratic equation, squaring, square root, and the trace is negligible compared to the time needed to compute a multiplication or an inversion. As indicated in [2], we have the following assumptions on equivalence of timing:

1 inversion ~ 3 multiplications

1 multiplication ~ 10 squarings

Our method reduces a number of inversions and multiplications which are expensive in computing the Weil pairing and thus provide a significant improvement.

## 5. Conclusion

We have proposed an efficient method for computing the Weil pairing. With the λ-representation in a normal basis, a significant improvement is presented while running point halving instead of doubling. The time saving is an important merit in the implementation of many new and interesting ID-based protocols that have been developed using the Weil pairing.

## References

[1] P. S. L. M. Barreto, H. Y. Kim, B. Lynn and M. Scott, “Efficient Algorithms for Pairing-based Cryptosystems”, *Advances in Cryptology-CRYPTO ’02*, pp. 354–368.

[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”, *ePrint 2004*.

[4] D. Boneh and M. Franklin, “Identity-based Encryption from the Weil Pairing”, *Advances in Cryptology-CRYPTO’01*, pp. 213–239.

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

[6] S. Galbraith, K. Harrison and D. Soldera, “Implementing the Tate Pairing”, *Algorithm Number Theory Symposium*, vol. 2369, Springer-Verlag Heidelberg, 2002, pp. 324–337.

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

[8] E. Knudsen, “Elliptic Scalar Multiplication Using Point Halving”, *Advances in Cryptology-ASIACRYPTO’99*, pp. 135-149.

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

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