Improving Miller's Algorithm Using the NAF and the Window NAF

Chapter · January 2013

DOI: 10.1007/978-3-642-40148-0_24

3 authors, including:

Siham Ezzouak
Université Mohammed Premier
5 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Abdelmalek Azizi
Université Mohammed Premier
211 PUBLICATIONS **559** CITATIONS
SEE PROFILE

International Journal of Security and Its Applications
Vol.8, No.4 (2014), pp.171-182
http://dx.doi.org/10.14257/ijsia.2014.8.4.16

Optimizing the Computing of Pairing with Miller's Algorithm

Siham Ezzouak, Mohammed El Amrani and Abdelmalek Azizi

ACSA Laboratory
The Department of Mathematics and Computer Science, Faculty of science,
University Mohammed First, Oujda, BP 60000 Morocco
{sezzouak@gmail.com, elamranimohammed001@yahoo.fr,
abdelmalekazizi@yahoo.fr}

Abstract

The Miller's algorithm is the best known algorithm for computing pairing. For this reason, numerous optimizations are applied to this algorithm. One of them is for making the basic loop of Miller's algorithm quicker with efficient arithmetic. In this paper, we try to do this by using Non Adjacent Form (NAF) and the window NAF (NAF_w) instead of the binary form of the key in the original Miller's algorithm. We show how this improvement can reduce the number of addition steps by 1/6 in the NAF representation or 1/2(w+1) in the NAF_w where w is the size of the window in the NAF. Thereby both methods speed up Miller for efficient pairing implementation over extension field but with the NAF_w some extra memory are needed with some restriction for w value.

Keywords: *elliptic curves, Pairing, Miller's algorithm, Non-adjacent Form, the window NAF*

1. Introduction

The use of pairing in cryptography was developed in an extraordinary pace and many researchers have devoted their time to the study of the elliptic curve. On the one hand, it allows us to simplify existing protocols, e.g.: the tripartite Diffe-Hellman protocol of Joux [1] and the decision problem of Diffe-Hellman [2]. On the other hand, we can construct the new protocol such as an Encryption based on the identity [3]. During the Crypto 2001 conference, Boneh and Franklin suggest an encryption scheme based on the identity using the pairing [4], and thus resolved the Shamir's Problem [5] exposed in 1984.

Moreover in cryptanalysis, we can reduce the elliptic curve discrete logarithm problem to a discrete logarithm problem over the finite field where some attacks known to be Sub-exponential [6].

In the majority of applications, one of the following pairing is used to construct cryptosystems:

- The Weil Pairing.
- The Tate pairing and the reduced Tate pairing.
- The Ate pairing and the Twist-Ate pairing.

For computing these pairings, we use the famous Miller algorithm.

In this paper, we attempt to improve this algorithm firstly by using the NAF and secondly with the window NAF. The remainder of this paper is organized as follows:

In Section 2, we introduce some basic definitions of the pairing mentioned above (the Weil, Tate and Ate pairing). In Section 3, we describe the original Miller algorithm. In

ISSN: 1738-9976 IJSIA

Section 4, we recall the definition of the NAF and the NAF_w representation, we replace the binary representation in Miller's algorithm by one of the two and we compare the number of iterations and the running time of two methods. Finally, in section 4, we conclude the paper.

2. Background

In this section, we provide a brief introduction to the pairing on elliptic curves especially the most known ones (Weil, Tate and Ate). For further details, please refer to reference [7].

2.1. The Pairing

Let G_1 and G_2 be two additive groups with identity 0 and G_3 be the multiplicative group (generally cyclic with the same order n) with identity 1. The pairing is a map:

$e_n : G_1 * G_2 \rightarrow G_3$ With the following properties ([8] p.183):

- Bilinearity.

  $\forall P, P' \in G_1 \setminus \{O\} \forall Q, Q' \in G_2 :$
  $- e_n(P + P', Q) = e_n(P,Q) + e_n(P', Q).$
  $- e_n(P, Q + Q') = e_n(P,Q) + e_n(P, Q').$

- Non-degenerate.

  $- \forall P \in G_1 \setminus \{O\} \exists Q \in G_2 \; such \; that \; e_n(P,Q) \neq 1.$ 정정정정
  $- \forall Q \in G_2 \setminus \{O\} \exists P \in G_1 \; such \; that \; e_n(P,Q) \neq 1.$ 정정정정

To define the pairing, we need a number of standards definitions of the theory of curves. We denote:

- O the point at infinity.
- $E[n] = \{P \in E(\overline{K}) : nP = O\}$ be the set of n-torsion points of E, which are not necessarily defined over K.
- E(K)[n] = E[n] ∩ E(K) the part of the E[n] defined over K.
- μ_n the algebraic group of nth roots of unity.
- q is a power of positive prime number p.

2.1.1. The Weil Pairing

Let P, Q ∈ E[n] , DP , DQ two divisors such as $D_P \backsimeq (P)-(O)$, $D_Q \backsimeq (Q)-(O)$ and f_P, f_Q two functions such as $div(f_P) = nD_P$ and $div(f_Q) = nD_Q$.

Definition: The Weil pairing on an elliptic curve E defined over a field K is a family of maps e_n each defined over K:

$$e_n : E[n] \times E[n] \rightarrow \mu_n$$
$$(P, Q) \rightarrow e_n(P, Q) = \frac{f_P(D_Q)}{f_Q(D_P)}$$

with the following properties [9]:

1. Identity $e_n(P,P) = 1$ for all P∈E[n].
2. Alternating $e_n(P,Q) = e_n(Q,P)^{-1}$ ∀ P, Q ∈E[n].
3. Compatibility with isogenies: if P∈ E[mn] and Q ∈E[n] then $e_{mn}(P, Q) = e_n(mP,Q)$.
4. Galois action: let P, Q ∈ E[n] and σ ∈ Gal($\overline{K} / K$), then $e_n(P,Q)^\sigma = e_n(P^\sigma, Q^\sigma)$.

We can compte the Weil pairing e_n by using the Miller function $f_{n,P}$ and $f_{n,Q}$ defined in section three such that:

$$e_n(P,Q)=(-1)^n \frac{f_{n,P}(Q)}{f_{n,Q}(P)}$$

2.1.2. The Tate Pairing

To define Tate pairing we consider:

- E an ordinary elliptic curve over finite field Fq.
- k the embedding degree defined us the least integer such us r divide qk-1.
- P $E(F_q)[r]$, s an integer $f_{s,P}$ rational function defined up to a constant factor by $div(f_{s,P}) = s(P)-(sP)-(s-1)(O)$.

Let $P \in E(F_q)[r]$ and $Q \in E(F_q^k)/rE(F_q^k)$, the Tate pairing is a map ([8] p.188):

$$e_T : E(F_q) * E(F_{q^k}) / rE(F_{q^k}) \rightarrow F_{q^k}^* / \left(F_{q^k}^*\right)^r$$

$$(P,Q) \rightarrow f_{r,P}(Q)^{\frac{q^k-1}{r}}.$$

The Tate pairing is non-degenerate bilinear pairing compatible with isogenies but it's neither alternating nor the identical to 1. Moreover, it's more efficient than the Weil pairing because we calculate the Miller's function once instead of twice with the Weil pairing. However, for most applications in cryptography, it's painful to present quotient group. So, several papers are devoted to improve this pairing and become the reduced Tate pairing when the algebraic group of roots of unity μ is used rather than the quotient group.

This modification can be held because there is an isomorphism between $F_{q^k}^* / \left(F_{q^k}^*\right)^r$ and r^{th} roots of unity μ it is given by:

$$\varphi : F_{q^k}^* \rightarrow \mu$$
$$x \rightarrow x^{(q^k-1)/r}$$

2.1.3. The Ate Pairing

The Ate pairing is introduced by Hess. It is an improved version of the Tate and generalization of the eta pairing available on certain super singular curves. It has half of the length of the Miller loop compared to the Tate pairing but the elliptic curve E must be defined with small values of the traces of Frobenius t.

Theorem 1 ([10]): Let E be an elliptic curve over F_q, r a large prime with r divide $\sharp E(F_q)$ and denote the trace of Frobenius with t, *i.e.*, $E(F_q) = q + 1 - t$ and π_q the Frobenius endomorphism defined such that:

$$\pi_q : E \rightarrow E$$
$$(x, y) \rightarrow (x^q, y^q)$$

Let $P \in E(F_q)[r] \cap \ker(\pi_q-[1])$, $Q \in E(F_q^k)/rE(F_q^k)) \cap \ker(\pi_q-[q])$ and T = t-1, then the Ate pairing of P and Q is computed with the Miller's function as :

$$e_A(Q,P) = f_{T,Q}(P)^{(q^k-1)/r}$$

3. Miller's Algorithm

Since we are defined one of the pairing, we must build the fs,P function. Miller's algorithm allows us to do this.

It is based on "Multiply and Add" a point in an elliptic curve combined with an update of some intermediate functions. We start with the following definition:

Definition ([13]): Let $P \in G_1$ and $Q \in G_2$ then the Miller's function $f_{s,P}$ is a rational function on E with s zeroes at P , one pole at [s]P and s-1 poles at O.
We denote ($f_{s,P}$) the divisor of $f_{s,P}$: ($f_{s,P}$) = s [P]-[sP] (s-1) [O] $\forall$ s $\in$ Z.
We construct $f_{s,P}$ using the following iterative formula:

$$f_{i+j,P}(Q) = f_{i,P}(Q) * f_{j,P}(Q) * \frac{l_{[iP,jP]}}{v_{[i+j]P}} \forall i, j \in \square$$

$v_{[i+j]P}$ is the equation of the vertical line through point [i + j]P.
$l_{[iP,jP]}$ is the equation of the line through points [i]P and [j]P.

3.1. Implementation

The equation of the line through points P and T is as following:

$$l_{T,P}(x,y) = \left\{ \begin{array}{l} y - y_p - \frac{y_T - y_P}{x_T - x_P}(x - x_p) T \neq O, P \\ y - y_p - \frac{3x_p^2 + a}{2y_P}(x - x_p) T = O, P \neq O \end{array} \right\}$$

Moreover the equation of the vertical line through point P is:

$$v_P(x,y) = x - x_p.$$

We present here the most used version of Miller's algorithm by updating numerators and denominators separately, thus avoiding division in line 6 and 10 in F_q^k so that just one inversion is needed at the end.

Let $Dl_{T,P}$, $Nl_{T,P}$, Dv_{2T} et Nv_{2T} be the denominator and the numerator of the line through T and P and the vertical line through point 2T respectively.

The Miller's algorithm is described by the pseudo-code bellow:

Algorithm 1 Miller's Original

Require: $r = \sum_{i=0}^{l-1} r_i 2^i$ with $r_i \in \{0,1\}$ P $\in$ E(F_q) and Q $\in$E(F_q^k)
Ensure: $f_{s,P}(Q)$
1: T $\leftarrow$ P
2: $f1 \leftarrow 1$
3: tmp $\leftarrow$ O
4: for i = l − 1 to 0 do
5: tmp $\leftarrow$ [2]T
7: T$\leftarrow$tmp
6: $f_1 = f_1^2 * l_{T;T}(Q) / v_{tmp}(Q)$

7: if $r_i = 1$ then
8: tmp⟵T + P
9: $f_1 = f_1 * l_{T;P}(Q) / v_{tmp}(Q)$
10: T⟵tmp
11: end if
12: end for
14: Return f_1

Algorithm 2 Simplified Miller's algorithm

Require: $r = \sum_{i=0}^{l-1} r_i 2^i$ with $r_i \in \{0,1\}$ $P \in E(F_q)$ and $Q \in E(F_q^k)$
Ensure: $f_{s,P}(Q)$
1: $T \leftarrow P$
2: $f_1 \leftarrow 1$
3: $f_2 \leftarrow 1$
4: tmp ← O
5: for $i = l - 1$ to 0 do
6: tmp ← [2]T
7: $f_1 \leftarrow f_1^2 * Nl_{T,T}(Q)*Dv_{tmp}(Q)$;
8: $f_2 \leftarrow f_2^2 *Dl_{T,T}(Q)*N\ v_{tmp}(Q)$;
9: T ← tmp
10: if $r_i = 1$ then
11: tmp ← T + P
12: $f_1 \leftarrow f_1 * Nl_{T,P}(Q)*Dv_{tmp}(Q)$;
13: $f_2 \leftarrow f_2 * Dl_{T,P}(Q)*N\ v_{tmp}(Q)$;
14: T ← tmp
20: end if
21: end for
22: Return $\frac{f_1}{f_2}$

Many useful techniques have been suggested to optimize the computing of the algorithm including:

- Speeding scalar multiplication [14].
- Reducing the loop length in Miller's algorithm [15].
- Performing the computing over the field $F_q^{k/d}$ instead of the field F_q^k using the twists ([16]).
- Using other variant of Miller's formula [17].
- Deleting the computing for the denominator [18].

In this paper, we present a generic approach that allows us to reduce the number of addition steps using the NAF and the NAF_w representations.

1.1 Complexity

The analysis of the computational complexity of the Miller's algorithm requires knowledge of the cost of two elements:

- Point doubling and addition of two points in $E(F_q)$.
- The multiplication and division in F_q^k for update f_i.

The cost of the remainder of the operations is negligible.

Inversions are the most costly; the cost is related to the particular implementation feature. For example, it is very high in smart cards equipped with cryptographic coprocessors. The commonly used ratio is $\frac{I}{M}=6$, but thanks to Fermat's theorem, the inverse requires only log(q) multiplications. Moreover, the cost of operations is generally measured by multiplication in the field.

In this article, we assume that $S \approx M$ and $S_k \approx M_k$, in some environments the ratio $\frac{S}{M}$ can be reduced to $\frac{S}{M} = 0.8$ [11].or $\frac{S}{M} = 0.6$ [12].

To compute the complexity, we compute operations performed in the algorithm. In the line 6-9, we found that the cost of doubling step is $2S_k+2 M_k + (2 + 3k)M+3S + I$ and the cost of adding step is $2M_k + (2 + 3k)M + S + I$ or $I \backsimeq 6M$ and $S \backsimeq M$ so the Miller's algorithm require $(5k + 15.5) \log(r) M + (5 \log(r) + 7)M_k)$.

We can multiply integers of n bits by:

- Naive algorithm $O(n^2)$ binary operations.
- Karatsuba algorithm in $O(n^{1.59})$ binary operations.
- Schonhage-Strassen algorithm in O(n logn log logn) binary operations.

For classical algorithms to multiply two elements of a finite field F_q, the complexity is $O((\log q)^{\mu})$, where μ depends on the multiplication algorithm used. The complexity of multiplication in F_q^k is $O((k \log q)^{\mu})$. Thus, the complexity of the Miller's algorithm is $O((5k + 15.5) \log r(\log q)^{\mu} + (5 \log(r) + 7)(k \log q)^{\mu})$.

This complexity is logarithmic in q and r, but polynomial in k.

4. A modified Miller's Algorithm

4.1 . Miller's Algorithm with the NAF

Instead of representing the key k with the binary representation in the original Miller's algorithm, we use the NAF representation known as the canonical representation with the fewest number of non-zero digits. In fact, the number of addition points in the Miller's algorithm linked to number of non-zero digits *i.e.,* the hamming weight of the key. If one decreases the last one, the number of operations is reduced and the running time will be improved. Furthermore, in the NAF representation, one must compute –P which does not require any operation $\forall P = (x,y) \in E(F_q), -P = (x,-y)$. Therefore, the cost of the additive operations in the NAF is ignored.

Definition ([19] p.98): A non-adjacent form (NAF) of a positive integer r is an expression $r = \sum_{i=0}^{l-1} r_i 2^i$ where $r_i \in \{0, \pm 1\}$, $r_{l-1} \neq 0$ and no two consecutive digits r_i are non-zero i.e. $\forall i$ $r_i r_{i+1}=0$. The length of the NAF is *l*.

Theorem 2 (Some properties of NAFs): Let k be a positive.

1. NAF (k) has the fewest non-zero digits of any signed digit representation.
2. The length of NAF(k) is at most one more than the length of the binary representation of k.

3. The average density of non-zero digits among all NAFs of length l is approximately 1/3.

If we use this presentation for computing kP, the expected running time will *l/3*A+ lD such that A and D the cost of the addition and the doubling point respectively instead of *l/2*A + lD in the binary representation. So including this representation in the Miller's algorithm will decrease the numbers of addition by approximately *l/6*.

Algorithm 3 NAF method for computing kP

Require: $k=\sum_{i=0}^{l-1} k_i 2^i$ with $k_i \in \{0, \pm 1\}$ $P \in E(F_q)$

Ensure: kP

1: $Q \leftarrow 0$
2: for $i = l-1$ to 0 do
3: $Q \leftarrow [2]Q$
4: if $r_i = 1$ then
5: $Q \leftarrow Q + P$
6: end if
7: if $r_i = -1$ then:
8: $Q \leftarrow Q - P$
9: end if
10: end for
11: Return Q.

The pseudo-code of Miller's algorithm using the NAF becomes:

Algorithm 4 The Miller's algorithm with the NAF

Require: $r = \sum_{i=0}^{l-1} r_i 2^i$ with $r_i \in \{0; 1\}$ $P \in E(F_q)$ and $Q \in E(F_q^k)$

Ensure: $f_{s,P}(Q)$

1: $T \leftarrow P$
2: $f1 \leftarrow 1$
3: $f2 \leftarrow 1$
4: tmp $\leftarrow$ O
5: for $i = l-1$ to 0 do
6: tmp $\leftarrow [2]T$
7: $f_1 \leftarrow f_1^2 * N_{l_{T,T}}(Q) * Dv_{tmp}(Q)$;
8: $f_2 \leftarrow f_2^2 * Dl_{T,T}(Q) * N v_{tmp}(Q))$;
9: $T \leftarrow$ tmp
10: if $r_i = 1$ then
11: tmp $\leftarrow T + P$
12: $f_1 \leftarrow f_1 * Nl_{T,P}(Q) * Dv_{tmp}(Q)$;
13: $f_2 \leftarrow f_2 * Dl_{T,P}(Q) * N v_{tmp}(Q)$;

14: $T \leftarrow tmp$
15: else if $r_i = -1$ then
16: $tmp \leftarrow T - P$
17: $f_1 \leftarrow f_1 * Nl_{T,-P}(Q) * Dv_{tmp}(Q)$;
18: $f_2 \leftarrow f_2 * Dl_{T,-P}(Q) * Nv_{tmp}(Q)$;
19: $T \leftarrow tmp$
20: end if
21: end for
22: Return $\frac{f_1}{f_2}$

The probability that $r_i = 0$ is 1/3 in this algorithm instead of 1/2 with the original Miller's algorithm. So the instructions from line 8 to 16 are performed with the probability 1/3 rather than 1/2.

Therefore, the complexity becomes:

$$(4.3k + 14) \log(r) M + (4.6 \log(r) + 7) M_k)$$

when we substitute M and M_k by the complexity of the multiplication in F_q and F_q^k, we find the complexity of Miller's modified with the NAF is

$$(4.3k +14) \log(r) (\log q)^\mu + (4.6 \log(r) + 7)(k \log q)^\mu$$

5.1 . Miller's Algorithm with the NAF_w

The window NAF is an improved version of the NAF which processes w digits of k at a time instead of one digit with the NAF which reduces the hamming weight. On the one hand the running time can be decreased, on the other hand more memory are used to store the k_iP. If extra memory is available this presentation is advised.

Definition ([19] p.99): Let $w \geq 2$ be a positive integer. A width-w NAF of a positive integer k is an expression $NAF_w(k) = \sum_{i=0}^{l-1} k_i 2^i$ where $k_i \in [-2^{w-1}, 2^{w-1}-1]$ where each non-zero coefficient k_i is odd, $|k_i| < 2^w - 1$, $k_{l-1} \neq 0$ and at most one of any w consecutive digits is non-zero. The length of the width-w NAF is l.

Theorem 3 (Some properties of the NAFw):

- $NAF_2(k) = NAF(k)$.
- The length of $NAF_w(k)$ is at most one more than the length of the binary representation of k.
- The average density of non-zero digits among all width-w NAF s of length l is approximately $\frac{1}{w+1}$.

To compute kP, one begins by writing the width-w NAFs of k and computing k_iP, after one can update the Q point depending on the k_i value as follows:

Algorithm 5 NAF_w method for computing kP

Require: window-width w, $k=\sum_{i=0}^{l-1} k_i 2^i$ where $k_i \in [-2^{w-1}, 2^{w-1}-1]$ and $P \in E(F_q)$

Ensure: kP
1: Calculate $P_{ki} = k_i P$, $k_i \in [1, 2^{w-1}-1]$
2: $Q \leftarrow O$
3: for i = l − 1 to 0 do
4: $Q \leftarrow [2]Q$
5: if $k_i = 0$ then
6: if $k_i > 0$ then
7: $Q \leftarrow Q + P_{ki}$;
8: else
9: $Q \leftarrow Q - P_{|ki|}$;
10: end if
11: end if
12: end for
13: Return Q.

Since the NAF_w representation reduces the hamming weight from $\frac{1}{2}$ to $\frac{1}{w+1}$, it is then possible to optimize the Miller's algorithm with this latter as follows:

Algorithm 6 Miller's algorithm with NAF_w

Require: window-width w, $k=\sum_{i=0}^{l-1} k_i 2^i$ where $k_i \in [-2^{w-1}, 2^{w-1}-1]$, $P \in E(F_q)$ and $Q \in E(F_q^k)$.
Ensure: $f_{s,P}(Q)$
1: Calculate $P_{ki} = k_i P$, $k_i \in [1, 2^{w-1}-1]$
2: $T \leftarrow P$
3: $f_1 \leftarrow 1$
4: $f_2 \leftarrow 1$
5: tmp ← O
6: for i = l − 1 to 0 do
7: tmp ← [2]T
8: $f_1 \leftarrow f_1^2 * Nl_{T,T}(Q) * Dv_{tmp}(Q)$;
9: $f_2 \leftarrow f_2^2 * Dl_{T,T}(Q) * Nv_{tmp}(Q))$;
10: T ← tmp
11: if $k_i = 0$ then
12: if $k_i > 0$ then
13: tmp ← $T + P_{ki}$
14: $f_1 \leftarrow f_1 * Nl_{T,Pki}(Q) * Dv_{tmp}(Q)$;
15: $f_2 \leftarrow f_2 * Dl_{T,Pki}(Q) * Nv_{tmp}(Q)$;
16: T ← tmp

17: else
18: tmp ← T − $P_{|ki|}$
19: $f_1 \leftarrow f_1 * Nl_{T,-P|ki|}(Q) * Dv_{tmp}(Q)$;
20: $f_2 \leftarrow f_2 * Dl_{T,-P|ki|}(Q) * N\,v_{tmp}(Q)$;
21: T ← tmp
22: end if
23: end if
24: end for
25: Return $\frac{f_1}{f_2}$.

Both representations can decrease the running time in the Miller algorithm but some precomputations (computing k_iP (line 1)) and extra memory (storing k_iP) are needed in the NAF_w.

Moreover, we see that lines from 12 to 21 are executed with probability l/(w+1) instead of l/3 in Miller with NAF and l/2 in the Miller simplified.

Therefore, the complexity of the algorithm becomes $O((11+3k+1/w(9+4k)\log(r)M + ((4+2/w)\log(r)+7)M_k)$. When we substitute M and M_k by the complexity of the multiplication in F_q and F_q^k, we find that the overall complexity of algorithm 6 is $O((11+3k+1/w(9+4k))\log(r)(\log q)^{\mu} + ((4+2/w)\log(r)+7)(k\log q)^{\mu})$.

As a consequence, the complexity decreases compared to NAF for w over three and compared to the simplified Miller for w greater than two.

5.2. Performance Comparison

In this section, we compare the three algorithms cited below especially the numbers of operations and the running time.

Assume that the square and the multiplication have the same cost, it means that M = S and $M_k = S_k$ and assume that the inversion is six times more than a multiplication I = 6M and $I_k = 6M_k$.

Suppose that w equals to four. The next table compares the number of operations needed for each algorithm.

Table 1. Costs of the Three Algorithms

Algorithm	Costs
Simplified Miller	$(5k + 15.5)\log rM + (5\log r + 7)M_k$
Miller with NAF	$(4.3k + 14)\log rM + (4.6\log r + 7)M_k$
Miller with NAF_4	$(4k+13.25)\log rM+(4.5\log r + 7)M_k$

Now, we compare the running time of Miller's algorithm with the binary method, the NAF and NAF_w method. Thus, we implement our algorithm on Intel Pentium dual core processor 1.86 GHz and 782 MHz and 512 MB of memory using SAGE (Software Algebra Geometry Experimentation) [20].

Table 2. Comparison of the Running Time between the Methods in Seconds

W	Miller Original	Miller with NAF	Miller with NAF_w
3	0.610563	0.537977	0.5305344
4	0.624229	0.555278	0.430871
5	0.616770	0.552936	0.549892
6	0.612716	0.558225	0.892682
7	0.621291	0.554980	1.373136

The Table 2 shows the comparison of running time with both three methods, the results were obtained by generating six elliptic curves and ten different pairs of points for each elliptic curve. After this, the average running time for specific elliptic curves and a specific w value were calculated.

In the table, we find that the Miller's algorithm with the NAF method is always faster than the simplified Miller. However, Miller with NAF_w is the best for w between three and five. This is because the number of k_iP calculated at the beginning of the program is linked to w. When the w is large the number of k_iP computed is very important. Thus, the running time is increased.

4 Conclusion

Since the Miller's algorithm is the heart of pairings, several optimizations are applied in this algorithm. In this paper, we present one of them. We have decreased the running time for both representations with extra memory requirement in the NAF_w and specific values of the w. Our approach can be combined with the recent versions of the Miller for leading other optimizations such as the mixed coordinate and the denominator elimination. Our future work will analyze these two possibilities of optimizations.

References

[1] A. Joux, "A one round protocol for tripartite Diffe-Hellman", Journal of Cryptology, vol. 17, no. 4, **(2004)**, pp. 263-276.
[2] A. Joux and K. Nguyen, "Separating Decision Diffe-Hellman from Diffe-Hellman in cryptographic groups", Journal of Cryptology, vol. 16, no. 4, **(2003)**, pp. 239-247.
[3] F. Hess, "Exponent Group Signature Schemes and Efficient Identity Based Signature Schemes Based on Pairings", IACR Cryptology ePrint Archive, no. 12, **(2002)**.
[4] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing", SIAM J. Comput., vol. 32, no. 3, **(2003),** pp. 586-615.
[5] A. Shamir, "Identity-based Cryptosystems and Signature Schemes", Proceedings of CRYPTO 84 on Advances in Cryptology, Springer-Verlag New York, Inc., **(1985)**, pp. 47-53.
[6] A. Menezes, T. Okamoto and S. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field", Proceedings of the twenty-third annual ACM symposium on Theory of computing, New York, USA, **(1991)**, pp. 80-89.
[7] H. Cohen, G. Frey and R. Avanzi, "Handbook of elliptic and hyperelliptic curve cryptography", CRC Press, **(2005).**
[8] I. Blake, G. Seroussi and N. Smart, "Advances in Elliptic Curve Cryptography", London Mathematical Society Lecture Note Series, Cambridge University Press, **(2005)**.
[9] J. H. Silverman, "The Arithmetic of Elliptic Curves, Graduate Texts in Mathematics, Springer-Verlag, vol. 106, **(1986)**.
[10] F. Hess, N. Smart and F. Vercauteren, "The Eta Pairing Revisited", IEEE Transactions on Information Theory, vol. 52, **(2006),** pp. 4595-4602.

[11] M. Ciet, M. Joye, K. Lauter and P. Montgomery, "Trading Inversion for Multiplication in Elliptic Curve Cryptography. Journal Designs", Codes and Cryptography, Springer, vol. 39, no. 2, **(2006)**, pp.189-206.
[12] C. H. Lim and H. S. Hwang, C. H. Lim and H. S. Hwang, "Fast Implementation of Elliptic Curve Arithmetic in GF(p^n)", Public Key Cryptography, LNCS, vol. 1751, **(2000),** pp. 455-461.
[13] V.-S. Miller, "The Weil pairing, and its efficient calculation", Journal of Cryptology, vol. 17, no. 4, **(2004)**, pp. 235-261.
[14] K. Eisentrger, K. Lauter and P. L. Montgomery, "Fast Elliptic Curve Arithmetic and Improved Weil Pairing Evaluation", LNCS, Springer, vol. 2612, **(2003),** pp. 343-354.
[15] D. Lubicz and D. Robert, "A generalisation of Miller's algorithm and applications to pairing computations on abelian varieties", IACR Cryptology ePrint Archive, **(2013),** pp. 192.
[16] C. Costello, T. Lange and M. Naehrig, "Faster pairing computations on curves with high-degree twists", In Public Key Cryptography: 13th International Conference on Practice and Theory in Public Key Cryptography, Proceedings, Springer Verlag, Paris, **(2010),** pp. 224-242.
[17] J. Boxall, N. El Mrabet, F. Laguillaumie and P. Le Duc, "A Variant of Miller's Formula and Algorithm", The 4th International Conference on Pairing Based Cryptography, Pairing, **(2010)**.
[18] P. S. L. M. Barreto, H. Y. Kim and M. Scott, "e_cient algorithms for pairing based cryptosystems", CRYPTO, LNCS, Springer, Heidelberg, vol. 2442, **(2002),** pp. 354-369.
[19] D. Hankerson, A. Menezes and S. Vanstone, "Guide to Elliptic Curve Cryptography, Springer Verlag, **(2004),** pp. 92.
[20] W. Stein, "SAGE mathematical software", Version 4.6, Available at, **(2010),** http://www.sagemath.org.

