Efficient Countermeasure for Securing the Eta Pairing Computation over Binary Fields

Mustapha Hedabou[1,2]
Department of Computer Science, ENSA de Safi, route Sidi Bouzid , 46000 Safi. Morocco[1]
Lattis, INSA de Toulouse, 135, avenue de Rangueil, 31077 Toulouse cedex 4. France[2]
(Email: mhedabou@gmail.com)

(Received May 31, 2010; revised and accepted Nov. 13, 2010)

Abstract

Pairing based cryptosystems have became suitable for use on constrained devices with small resources. Recently, It has shown that side channel attacks are a serious threat for such cryptographic applications. In order to secure the pairing based cryptosystems against the side channel attacks, many countermeasures have been proposed but their cost is proved to be highly expensive. In this paper, we propose a new technique for securing the eta pairing η_T over binary fields $\mathbb{K} = \mathbb{F}_{2^m}$. The main idea of the proposed countermeasure is the use of the randomized values $x + f_i(r)$, where x is a variable involved in the computation of eta pairing, f_i is a polynomial function, and r is a random integer in $\mathbb{F}_{2^m}$. The overhead cost of the proposed countermeasure is only $(m + 1)/2$ field multiplications which makes it the most efficient known countermeasure for securing eta pairing against the side channel attacks over binary fields.

Keywords: Countermeasures, eta pairing, side channel attacks (SCA)

1 Introduction

Most identity based schemes use bilinear functions defined on elliptic curve such as Weil Pairing and Tate Pairing. A lot of researches have focused on the improvement of Tate and Weil Pairing computation, on which depends the efficiency of the pairing based cryptosystems. All these algorithms are based on Miller's algorithm [12]. In recent years, a large number of papers [1, 6, 7] have improved the efficiency of the pairing computation. A new pairings such as the eta pairing [3, 6] and ate pairing [8, 9] have been proposed.

Pairing based cryptosystems have been implemented on hardware devices with small resources like FPGAs [10, 15] and smart cards [18]. The security of such applications against Side Channel and Fault attacks [8, 11] have became a serious question. For a category of the Pairing based cryptosystems, like the short signature scheme by Boneh et al. [5], well known Countermeasures can be used. The point scalar randomization and projective coordinates are among the most famous of these countermeasures.

Only few techniques are known for securing the other category of the Pairing based cryptosystems, such as the Boneh-Franklin's encryption scheme [4], that uses a secret point as secret information. These countermeasures include the techniques proposed by Page and Vercauteren [14], Scott [16] and Kim et al. [13]. The study of of these countermeasures shows that in despite of their efficiency their costs are highly expensive.

In this paper, we propose a new countermeasure for securing the eta pairing against side channel attacks on elliptic curves defined over $\mathbb{K} = \mathbb{F}_{2^m}$. The proposed countermeasure randomizes each value x handled during the computation by using $x + f_i(r)$, where where f_i is a polynomial function, and $r \in \mathbb{F}_{2^m}$ is a random integer. The overhead cost of the proposed countermeasure is only $(m + 1)/2$ field multiplications.

2 Tate and Eta Pairing

Let $E(\mathbb{F}_q)$ be an elliptic curve defined over a field $\mathbb{F}_q$. Let l be a positive integer, co-prime to q, such that $E(\mathbb{F}_q)$ contains a point of order l. In cryptographic implementations, l is usually taken to be a large prime such that $l/\#E(Fq)$. Let k be the smallest integer satisfying $l/q^k - 1$. This value k, is the embedding degree of the curve with respect to l. The Tate pairing is defined in terms of rational functions over points of an elliptic curves evaluated in a divisor. Let $P \in E(\mathbb{F}_q)[l]$, then $l(P) - l(O)$ is a principal divisor. So there is a rational function $f_p \in \mathbb{F}_{q^k}(E)$ with $div(f_P) = l(P) - l(O)$. Let $Q \in E(\mathbb{F}_{q^k})[l]$ be a point with coordinates in $\mathbb{F}_{q^k}$; then we construct a divisor D such that $D_Q \sim (Q) - (O)$. D should be chosen so that its support is disjoint from the support of th divisor of f_p. Now let μ_l be the subgroup of $l - th$ roots of unity in $\mathbb{F}^*{}_q$.

The Tate pairing is defined as follow:

$$e_l : E(\mathbb{F}_q)[l] \times E(\mathbb{F}_{q^k})[l] \longrightarrow \mu_l$$
$$(P, Q) \longrightarrow f_{l,p}(D_Q)^{\frac{q^k-1}{l}}.$$

Miller's algorithm [12] provides a way to compute the Tate pairing as well as Weil pairing.

To speed up the pairing computation on elliptic curve of the form $y^2 = x^p - x - d$ over $\mathbb{F}_{p^m}$ where $p \geq 3$, Duursma and Lee [6] proposed to replace the group order l by a value of the form $N = hl$ which has a low Hamming weight. When $T = p^{mp} + 1$ the final exponentiation is to the power $p^{mp} - 1$ and it is computed efficiently by using the Frobenius map. Since $f_{hl,P} = f^h_{l,P}$, the Tate computation is done by $f_{T,p}(Q)^{\frac{q^k-1}{T}}$.

Barreto et al. [2] gave a further improvement on Duursma and Lee for supersingular elliptic curves. Their method is called eta pairing. A special choice of T allows to halve the length of the main loop. In characteristic two, we consider the supersingular elliptic curve $E(\mathbb{F}_{2^m})$ defined by $y^2 + y = x^3 + x + b$ where $b \in \mathbb{F}_2$ and m is odd. The order of $E(\mathbb{F}_{2^m})$ is $2^m \pm 2^{(m+1)/2} + 1$ and the embedding degree is $k = 4$. According to [6] we use the distortion map $\psi(x, y) = (x + s^2, y + sx + t)$. For some integer T the eta pairing is defined to be

$$\eta_T(P, Q) = f_{T,p}(\psi(Q)).$$

For an integer T such that $T^a + 1 = LN$ for some $a \in \mathbb{N}$ and $L \in \mathbb{Z}$, $T = q + cN$ for some $c \in \mathbb{Z}$ and $M = (q^k - 1)/N$ we have

$$\left(\eta_T(P, Q)^M\right)^{aT^{a-1}} = \left(e_l(P, \psi(Q))\right)^L.$$

The particular choices $T = \pm 2^{(m+1)/2} + 1$, $a = 2$, $c = -1$, and $L = 2$ give rise to the following relation

$$\begin{aligned} \left(\eta_T(P, Q)^M\right)^{2T} &= \left(e_l(P, \psi(Q))\right)^2 \\ \Longrightarrow \left(\eta_T(P, Q)^M\right)^T &= \left(e_l(P, \psi(Q))\right). \end{aligned}$$

The computation of the eta pairing requires first the computation of the rational function corresponding to $2^{(m+1)/2}P$ and $\pm P$. Barreto et al. showed by induction that:

$$[2^i]P = \phi^i({x_p}^{(2i)}, {y_p}^{(2i)}),$$

where $\phi(x, y) = (x+1, y+x)$ and $a^{(i)} = a^{2^i}$ for every field element a. Only the use of the Frobenius map is involved, and thus no adding and doubling point operations are needed for the computation of the eta pairing. On the other hand, to obtain the same result as the Tate pairing, a more exponentiation to the power of T is needed.

The field $\mathbb{F}_{2^{4m}}$ has elements s, t such that $s^2 = s + 1$ and $t^2 = t + s$. If $m \equiv 3 \bmod 8$ and the field $\mathbb{F}_{2^{4m}}$ is represented the basis $\{1, s, t, st\}$ the computation of the eta pairing is done as Algorithm 1.

Algorithm 1 Computation of the eta pairing on $E(\mathbb{F}_{2^m})$: $y^2 + y = x^3 + x + b$

```
Input: $P = (x_P, y_P)$, $Q = (x_Q, y_Q)$
Output: $\eta_T(P, Q)$
1. $u \leftarrow x_p + 1$
2. $f \leftarrow u(x_P + x_Q + 1) + y_P + y_Q + b + 1 + (u + x_Q)s + t$
3. for $i = 1$ to $(m + 1)/2$ do
4. $u \leftarrow x_p, x_p \leftarrow \sqrt{x_p}, y_p \leftarrow \sqrt{y_p}$
5. $g \leftarrow u(x_P + x_Q) + y_P + y_Q + x_P + (u + x_Q)s + t$
6. $f \leftarrow f.g$
7. $x_Q \leftarrow {x_Q}^2, y_Q \leftarrow {x_Q}^2$
8. end for
9. return $f^{(2^{2m}-1)(2^m-2^{(m+1)/2}+1)(2^{(m+1)/2}+1)}$
```

3 Side Channel Attacks and Their Countermeasures

The first side channel attack [11] against pairing based cryptosystems was proposed by Page and Vercauteren [14] in 2004. The attack is mounted on Tate pairing com- putation based on Duursma and Lee algorithm [6] on supersingular elliptic curves in characteristic three. Whelan and Scott [16] showed that the side channel attacks can also be mounted on Tate, Ate and Eta pairing. Kim et al. [13] explained that the Eta pairing computation on supersingular curves of characteristic 2 may succumb to SPA, DPA and timing attacks. Recently, Whelan et al. [19] validated successfully a CPA attack on Eta pairing computation and provided a practical analysis that show the correlation between the power consumption and the secret input.

The main idea of the side channel attacks on pairing based cryptosystems is the interaction between the known and secret data used on the computation via finite fields operation. Indeed, during the pairing computation a field operation like $y.r$, where y is an unknown and fixed value related with the y-coordinate of the secret point and r is a known and variable value related with the known point, is performed. Since a DPA attack can be mounted against field multiplication then a SCA attack can successfully mounted against pairing computation. However, for some kind of pairing, like Eta pairing in characteristic two, a more complicated operation is computed. The multiplication $a(b + r)$ where a and b are unknown is performed. Kim et al. [13] claimed that the power analysis is harder but feasible in this case.

As mentioned above, to prevent the side channel attacks on pairing based cryptosystems many countermeasures have been proposed. In [16], Scoot suggested to randomize all steps during the pairing computation by multiplying all intermediates values by a random element of $\mathbb{F}_q$. Each used intermediate value x is then replaced by $r.x$ where r is a random element in the finite field. The effect of this operation is removed by the final exponentiation. Kim et al. [13] proposed to use the projective coordinates $(r.x_Q;\ r.y_Q;\ r)$ of the point Q instead of the affine coordinates $(x_Q; y_Q)$, for a random integer r. Page and Vercauteren [14] used bilinearity to randomize the private data, i.e., $e_l(P,Q) = e_l(sP, tP)^{\frac{1}{st}}$ where s and t are random variables. Furthermore, the exponentiation to the power $\frac{1}{st}$ can be removed by selecting s and t satisfying $st = 1 \bmod l$, where l is the order of the underlying elliptic curve for the pairing. They also presented the method for blinding the input point by using the relation: $e_l(P,Q) = e_l(P, Q+R).e_l(P,R)^{-1}$.

We will assume that $P = (x_P; y_P)$ is the secret input and $Q = (x_Q; y_Q)$ is the known data. As Suggested by Whelan and Scott [17] and confirmed by practical analysis [19] there are two main points of attack on the eta pairing algorithm. The first point attack lies on the computation of $u(x_P + x_Q + 1)$ outside the loop and the second one on the computation of the square root function $\sqrt{x_p}$ inside the loop.

4 Proposed Countermeasure

In this section, we propose a new countermeasure that aims to secure the computation of the eta pairing on elliptic curves E defined over finite field $\mathbb{K} = \mathbb{F}_{2^m}$. The main idea of the proposed countermeasure is the use of a randomized values $x + f_i(r)$, where where f_i is a polynomial function, and $r \in \mathbb{F}_{2^m}$ is a random integer. The effect of these changes will be removed at the end of each loop. Here under we give an algorithm that implements this technique and show that the outputted result is the correct value of the η_T. In the next section we study the security and efficiency of the proposed countermeasure.

Algorithm 2 Secure computation of the eta pairing on $E(\mathbb{F}_{2^m}) : y^2 + y = x^3 + x + b$

```
Input: $P = (x_P, y_P)$, $Q = (x_Q, y_Q)$
Output: $\eta_T(P,Q)$
1.1 Generate a random integer $r \in \mathbb{F}_{2^m}$
1.2 $x_P \leftarrow x_P + r, y_P \leftarrow y_P + r, x_Q \leftarrow x_Q + r, y_Q \leftarrow y_Q + r, b \leftarrow b + r, s \leftarrow s + r, t \leftarrow t + r$,
1.3 $u \leftarrow x_p + 1$
2.1 $f \leftarrow u(x_P + x_Q + 1) + y_P + y_Q + b + 1 + (u + x_Q)s + t$
2.2 $\lambda \leftarrow r + r^2, v \leftarrow x_P, x_P \leftarrow x_P + \lambda, y_P \leftarrow y_P + \lambda$
3. for $i = 1$ to $(m+1)/2$ do
4.   $u \leftarrow v, x_P \leftarrow \sqrt{x_P}, y_P \leftarrow \sqrt{y_P}, w \leftarrow r.(x_P + u)$
5.   $g \leftarrow u(x_P + x_Q) + y_P + y_Q + x_P + (u + x_Q)s + t + w$
6.   $f \leftarrow f.g$
7.   $x_Q \leftarrow {x_Q}^2 + \lambda, y_Q \leftarrow {y_Q}^2 + \lambda, v \leftarrow x_P, x_P \leftarrow x_P + \lambda, y_P \leftarrow y_P + \lambda$
8. end for
9. return $f^{(2^{2m}-1)(2^m-2^{(m+1)/2}+1)(2^{(m+1)/2}+1)}$
```

4.1 Correctness of Algorithm 2

In this section, we prove that Algorithm 2 outputs the correct value of the eta pairing $\eta_T(P,Q)$. This will be done by induction. At each step, we will show that the proposed algorithm outputs the same result as Algorithm 1, which means that the effect of the randomized values will be removed at the end of each loop. The final result outputted by Algorithms 1 and 2 will then be the same.

The main idea of our approach is to show that at each step in Algorithm 2 and Algorithm 1 output the same intermediates values f and g. To distinguish the value used by each algorithm we will denote the values handled by Algorithm 2 by X instead of x. For example, the coordinates of $P = (x_P, y_P)$ will be noted $P = (X_P, Y_P)$ and the value s by S. Only t will be replaced by T' instead of T for avoiding a conflict of notation.

Since the characteristic of the field $\mathbb{F}_{2^m}$ is 2, i.e $2.r = 0$ for every element r of $\mathbb{F}_{2^m}$, it is easy to see that the result computed by Algorithm 2 outside the main loop is the same as in Algorithm 1. Indeed, Algorithm 2 performs

$$\begin{aligned} f &= U(X_P + X_Q + 1) + Y_P + Y_Q + B + 1 \\ &\quad\quad +(U + X_Q)S + T' \\ &= (u + r)(x_P + r + x_Q + r + 1) + y_P + r + y_Q + r \\ &\quad\quad +b + r + 1 + (u + r + x_Q + r)(s + r) + t + r \\ &= u(x_P + x_Q + 1 + 2r) + 2r(x_P + x_Q + 1 + 2r) + y_P \\ &\quad\quad +y_Q + 2r + 1 + b + (u + x_P + 2r)s + t + 2r \\ &= u(x_P + x_Q + 1) + y_P + y_Q + 1 + b \\ &\quad\quad +(u + x_Q+)s + t. \end{aligned}$$

Now, we will show by induction that Algorithms 1 and 2 output the same result at the end of each loop. At the beginning of the first loop we have

$$\begin{aligned} U &\leftarrow x_P + r, \\ X_P &\leftarrow \sqrt{x_P} + r, \\ Y_P &\leftarrow \sqrt{y_P} + r, \\ X_Q &\leftarrow x_Q + r, \\ Y_Q &\leftarrow y_Q + r. \end{aligned}$$

Here under we explain how this holds for x_P:

$$X_P \leftarrow \sqrt{x_P + \lambda} = \sqrt{x_P + r + r + r^2} = \sqrt{(x_P + r)^2} = \sqrt{x_P} + r.$$

The computation of g by Algorithm 2 gives

$$\begin{aligned} g &= (x_P + r)(\sqrt{x_P} + r + x_Q + r) + \sqrt{y_P} + r + y_Q \\ &\quad\quad +r + \sqrt{x_P} + r + (x_P + r + x_Q + r)(s + r) \\ &\quad\quad +t + r + r.(\sqrt{x_P} + r + x_P + r) \\ &= x_P(\sqrt{x_P} + x_Q) + \sqrt{y_P} + y_q + \sqrt{x_P} \\ &\quad\quad +(x_P + x_Q)s + t \end{aligned}$$

Which prove that the inductions hypothesis holds.

By the same way, it is easy to see that at the end of the first loop we get

$$\begin{aligned} X_Q &\leftarrow {x_Q}^2 + \lambda = {x_Q}^2 + r, \\ Y_Q &\leftarrow {y_Q}^2 + r, \\ V &\leftarrow \sqrt{x_P} + r, \\ X_P &\leftarrow \sqrt{x_P} + r^2, \\ Y_P &\leftarrow \sqrt{y_P} + r^2. \end{aligned}$$

By induction hypothesis, we suppose that Algorithms 1 and 2 output the same result at the i-th loop. At the end of the i-th loop of Algorithm 1 we have

$$\begin{aligned} u &\leftarrow {x_P}^{-(i-1)}; \\ x_P &\leftarrow {x_P}^{-(i)}, \\ x_Q &\leftarrow {x_Q}^{(i)}, \\ y_Q &\leftarrow {y_Q}^{(i)}, \end{aligned}$$

where $a^{-(i)}$ denotes the 2^i-th square root of a and $a^{(i)}$ the 2^i-th square of a. Thus, at the the $i+1$-th loop Algorithm 1 computes

$$\begin{aligned} g = {} & x_P^{-(i)}({x_P}^{-(i+1)} + {x_Q}^{(i)}) + {y_P}^{-(i+1)} + {y_Q}^{(i)} \\ & + {x_P}^{-(i+1)} + ({x_P}^{-(i)} + {x_Q}^{(i)})s + t \end{aligned}$$

On the other hand, at the same loop of Algorithm 2, we have

$$\begin{aligned} U &\leftarrow {x_P}^{-(i)} + r, \\ X_P &\leftarrow {x_P}^{-(i)} + r^2, \\ Y_P &\leftarrow {y_P}^{-(i)} + r^2, \\ X_Q &\leftarrow {x_Q}^{(i)} + r, \\ Y_Q &\leftarrow {y_Q}^{(i)} + r. \end{aligned}$$

Thus, at the $i+1$-th loop Algorithm 2 performs

$$\begin{aligned} g &= ({x_P}^{-(i)} + r)({x_P}^{-(i+1)} + r + {x_Q}^{(i)} + r) + {y_P}^{-(i+1)} \\ &\quad + r + {y_Q}^{(i)} + r + {x_P}^{-(i+1)} + r + ({x_P}^{-(i)} + r \\ &\quad + {x_Q}^{(i)} + r)(s + r) + r.({x_P}^{-(i+1)} + r \\ &\quad + {x_P}^{-(i)} + r) + t + r \\ &= {x_P}^{-(i)}({x_P}^{-(i+1)} + {x_Q}^{(i)}) + {y_P}^{-(i+1)} + {y_Q}^{(i)} \\ &\quad + {x_P}^{-(i+1)} + ({x_P}^{-(i)} + {x_Q}^{(i)})s + t \end{aligned}$$

Hence Algorithms 1 and 2 output the same result at the $i+1$-th loop, which prove the correctness of our assertion.

4.2 Security and Efficiency

This section discusses the security of the proposed countermeasure. As mentioned be- fore, there are two main points of attack on the eta pairing algorithm. The first point attacks lies on the computation of $u(x_P + x_Q + 1)$ outside the loop, and the second one on the computation of the square root function$\sqrt{x_P}$ inside the loop. With the pro- posed countermeasure, all the intermediate variables used at each loop are randomized. Indeed, instead of performing these vulnerable operations, Algorithm 2 performs $(u + r)((x_P + r) + (x_Q + r) + 1)$ and $\sqrt{(x_P + r^2)}$, where r is a random integer. Thus, the attacks mounted in [17, 19] would no longer be possible.

As mentioned in [17], to thwart the threat of the high order side channel attacks a new random value must be used at each loop. Minor changes are needed to make the proposed algorithm fulfils this requirement. Indeed, we have just to use two random integers r and β. The first integer r will be used to randomize the values outside the loop and the second one to randomize the values inside the loop. The algorithm implementing this technique can be found in the appendix.

Since we generate a new random value at each loop, all intermediates variables used by this algorithm are randomized with a new value at each step i, i.e $\alpha_i + \alpha_{i+1}$ where α_i and α_{i+1} are respectively the random integers generated at the i-th and $i+1$-th loops (cf. appendix). Thus the collected information from a previous step can not be used in the next step. According to [17], the proposed countermeasure can prevent the high order side channel attacks.

With some field additions, the extra field operations performed by Algorithm 2 than the eta pairing algorithm are r^2 outside the loop, and $r.(x_P + u)$ at each loop. If the changes required to defeat high order attacks are implemented the only difference that occurs is that the operation r^2 is performed inside the loop. Since addition and squaring in $\mathbb{F}_{2^m}$ are relatively inexpensive compared to multiplication we can ignore their cost. Thus, the total cost of our countermeasure in the both cases is $(m+1)/2$ field multiplications.

The costs of the other countermeasures for securing the eta pairing against side channel attack are already evaluated in [13]. The following table gives a summary of the cost of all known countermeasures.

In Table 1, where α is the cost of a final exponentiation plus 1 extension field multiplication in $\mathbb{F}_{2^{4m}}$ and 1 extension field inversion in $\mathbb{F}_{2^{4m}}$ [13].

5 Conclusion

In this paper we have proposed a new countermeasure for securing the Eta pairing η_T over binary fields $\mathbb{F}_{2^m}$. The proposed countermeasure prevents both SCA and high order SCA and it's overhead cost is only $(m+1)/2$ field multiplications.

References

[1] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott, "Efficient algorithms for pairing-based cryptosystems," *Advances in Cryptology*, LNCS 2442, pp. 354-368, Springer-Verlag, 2002.

[2] P. S. L. M. Barreto, S. D. Galbraith, C. O. hÉigeartaigh, and M. Scott, "Efficient pairing computation on supersingular abelian varieties," *Cryp-*

Table 1: Additional cost for securing Eta pairing over binary fields

Countermeasure	Additional Cost
Page-Vercauteren (randomized private value)	$12m$M
Page-Vercauteren (blinding public value)	$3.5(m+1)$M $+ \alpha$
Scott's Countermeasure	$4(m+1)$ M + 4M
Kim et al. Countermeasure	$3(m+1)$M +4M
proposed Countermeasure	$(m+1)/2$M

tology ePrint Archive, Report 2004/375, 2004. (http://eprint.iacr.org/2004/375)

[3] P. S. L. M. Barreto, S. D. Galbraith, C. O. hÉigeartaigh and Michael Scott, "Efficient pairing computation on supersingular abelian varieties," *Designs, Codes and Cryptography*, vol. 42, no. 3, pp. 239-271, 2007.

[4] D. Boneh and M. Franklin, "Identity based encryption from the Weil pairing," *Journal of Computing*, vol. 32, no. 3, pp. 586-615, 2003.

[5] D. Boneh, B. Lynn and H. Shacham, "Short signatures from the Weil pairing," *Asiacrypt*, LNCS 2248, pp. 514-532, Springer-Verlag, 2002.

[6] I. M. Duursma and H. S. Lee, "Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$," *Asiacrypt*, LNCS 2894, pp. 111-123, Springer-Verlag, 2003.

[7] S. D. Galbraith, K. Harrison, and D. Soldera, "Implementing the Tate pairing," *Algorithmic Number Theory*, LNCS 2369, pp. 324-337, Springer-Verlag, 2002.

[8] S. Ghosh, D. Mukhopadhyay and D. Chowdhury, "Fault attack and countermeasures on pairing based cryptography," *International Journal of Network Security*, vol. 12, no. 1, PP. 21-28, 2011.

[9] P. Grabher, "Hardware acceleration of the Tate pairing in characteristic three," *CHES*, LNCS 3659, pp. 398-411, Springer-Verlag, 2005.

[10] R. Granger, F. Hess, R. Oyono, N. Theriault, F. Vercauteren, and T. U. Berlin, "Ate pairing on hyperelliptic curves," em Cryptology in Eurocrypt, LNCS 4515, pp. 430-447, Springer-Verlag, 2007.

[11] F. Hess, N. P. Smart, and F. Vercauteren, "The eta pairing revisited," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4595-4602, 2006.

[12] T. H. Kim, T. Takagi, D. G. Han, H. W. Kim, and J. Lim, "Side channel attacks and countermeasures on pairing based cryptosystems over binary fields," *Cryptology and Network Security (CANS 2006)*, LNCS 4301, pp. 168-181, Springer-Verlag, 2006.

[13] P. Kocher, J. Jaffe, B. Jun, "Differential power analysis," *Advances in Cryptology-Crypto' 99*, LNCS 1666, pp. 388-397, Springer-Verlag, 1999.

[14] V. S. Miller, "The Weil pairing, and its efficient calculation," *Journal of Cryptology*, vol. 17, no. 4, pp.235-261, 2004.

[15] D. Page and F. Vercauteren, "Fault and side-channel attacks on pairing based cryptography," *Cryptology ePrint Archive*, Report 2004/283, 2004.

[16] M. Scott, N. Costigan, and W. Abdulwahab "Implementing cryptographic pairings on smartcards," *CHES*, LNCS 4249, pp. 134-147, Springer-Verlag, 2006.

[17] M. Scott, "Computing the tate pairing," *CT-RSA*, LNCS 3376, pp. 293-304, Springer-Verlag, 2005.

[18] C. Shu, S. Kwon, and K. Gaj, "FPGA accelerated Tate pairing based cryptosystems over binary fields," *Cryptology ePrint Archive*, Report 2006/179, 2006.

[19] C. Whelan and M. Scott, "Side channel analysis of practical pairing implementations: Which path is more secure," *VIETCRYPT*, LNCS 4341, pp. 81-98, Springer-Verlag, 2006.

6 Appendix

The following algorithm implements the technique that allows to secure the eta pairing against high order side channel attacks.

Algorithm 3 Secure computation of the eta pairing on $E(\mathbb{F}_{2^m}) : y^2 + y = x^3 + x + b$

```
Input: $P = (x_P, y_P)$, $Q = (x_Q, y_Q)$
Output: $\eta_T(P, Q)$
1.1 Generate a random integer $r \in \mathbb{F}_{2^m}$
1.2 $x'_P \leftarrow x_P + r, y'_P \leftarrow y_P + r, x'_Q \leftarrow x'_Q + r, y'_Q \leftarrow y_Q + r, b' \leftarrow b + r, s' \leftarrow s + r, t' \leftarrow t + r$
1.3 $u' \leftarrow x'_p + 1$
2.1 $f \leftarrow u'(x'_P + x'_Q + 1) + y'_P + y'_Q + b' + 1 + (u' + x'_Q)s' + t'$
3. for $i = 1$ to $(m+1)/2$ do
4. Generate a random integer $\alpha \in \mathbb{F}_{2^m}$ and set $\beta \leftarrow 0$
5. $v \leftarrow x_P + \alpha, x_P \leftarrow x_P + \alpha^2, y_P \leftarrow y_P + \alpha^2, x_Q \leftarrow x_Q + \alpha, y_Q \leftarrow y_Q + \alpha, b \leftarrow b + \alpha, s \leftarrow s + \alpha, t \leftarrow t + \alpha$
6. $u \leftarrow v, x_P \leftarrow \sqrt{x_P}, y_P \leftarrow \sqrt{y_P}, w \leftarrow \alpha.(x_P + u)$
7. $g \leftarrow u(x_P + x_Q) + y_P + y_Q + x_P + (u + x_Q)s + t + w$
8. $f \leftarrow f.g$
9. $\beta \leftarrow \beta + \alpha + \alpha^2, x_Q \leftarrow {x_Q}^2 + \beta, y_Q \leftarrow {x_Q}^2 + \beta, v \leftarrow x_P, x_P \leftarrow x_P + \beta, y_P \leftarrow y_P + \beta$
10. end for
11. return $f^{(2^{2m}-1)(2^m-2^{(m+1)/2}+1)(2^{(m+1)/2}+1)}$
```

Proof. By the same as for Algorithm 2, we will prove that the effect of the randomized values will be removed at the end of each loop and then the outputted result will be the correct value of $\eta_T(P, Q)$.

It is easy to see that that the result computed by Algorithm 3 outside the main loop is the same as in Algorithm 1, since the performed operations are the same as for Algorithm 2. On the other hand, the computation of g by Algorithm 3 gives

$$\begin{aligned} g &= (x_P+\alpha)(\sqrt{x_P}+\alpha+x_Q+\alpha)+\sqrt{y_P}+\alpha+y_Q \\ &\quad +\alpha+\sqrt{x_P}+\alpha+(x_P+\alpha+x_Q+\alpha)(s+\alpha) \\ &\quad +t+\alpha+\alpha.(\sqrt{x_P}+\alpha+x_P+\alpha \\ &= x_P(\sqrt{x_P}+x_Q)+\sqrt{y_P}+y_q+\sqrt{x_P} \\ &\quad +(x_P+x_Q)s+t \end{aligned}$$

Which prove that Algorithms 1 and 3 output the same result at the first step.

By induction hypothesis, we suppose that Algorithms 1 and 3 output the same result at the i-th loop. AS mentioned before, at the end of the $i+1$-th loop of Algorithm 1 we have

$$\begin{aligned} g &= {x_P}^{-(i)}({x_P}^{-(i+1)}+{x_Q}^{(i)})+{y_P}^{-(i+1)}+{y_Q}^{(i)} \\ &\quad +{x_P}^{-(i+1)}+({x_P}^{-(i)}+{x_Q}^{(i)})s+t. \end{aligned}$$

At the $i+1$-th loop Algorithm 3 performs

$$\begin{aligned} g &= ({x_P}^{-(i)}+\beta')({x_P}^{-(i+1)}+\beta'+{x_Q}^{(i)}+\beta') \\ &\quad +{y_P}^{-(i+1)}+\beta'+{y_Q}^{(i)}+\beta'+{x_P}^{-(i+1)}+\beta' \\ &\quad +({x_P}^{-(i)}+\beta'+{x_Q}^{(i)}+\beta')(s+\beta') \\ &\quad +\beta'.({x_P}^{-(i+1)}+\beta'+{x_P}^{-(i)}+\beta')+t+\beta' \\ &= {x_P}^{-(i)}({x_P}^{-(i+1)}+{x_Q}^{(i)})+{y_P}^{-(i+1)}+{y_Q}^{(i)} \\ &\quad +{x_P}^{-(i+1)}+({x_P}^{-(i)}+{x_Q}^{(i)})s+t, \end{aligned}$$

where $\beta' = \alpha_i + \alpha_{i+1}$. The elements α_i and α_{i+1} are respectively the random integers generated at i-th and $i+1$-th step. Hence Algorithms 1 and 3 output the same result at the $i+1$-th loop, which complete the proof. $\square$

Mustapha Hedabou received his M. Sc degree in Mathematics from the university of Paul Sabatier, Toulouse, France. In 2006, he received his PhD degree in computer science from INSA de Toulouse, France. He is currently an Associate professor at ENSA de Safi in Morroco. His area interest covers Information Security, Public Key Cryptography based on Elliptic Curves and Identity based cryptography.