Security of Prime Field Pairing Cryptoprocessor Against Differential Power Attack

Santosh Ghosh, Debdeep Mukhopadhyay, and Dipanwita Roy Chowdhury

Department of Computer Science and Engineering,
Indian Institute of Technology,
Kharagpur, India
{santosh, debdeep, drc}@cse.iitkgp.ernet.in

Abstract. This paper deals with the differential power attack on a pairing cryptoprocessor. The cryptoprocessor is designed for pairing computations on elliptic curves defined over finite fields with large prime characteristic. The work pinpoints the vulnerabilities of such pairing computations against side-channel attacks. By exploiting the power consumptions, the paper experimentally demonstrates such vulnerability on FPGA platform. A suitable counteracting technique is also suggested to overcome such vulnerability.

Key words: Pairing Based Cryptography, Side-channel Analysis, Power Analysis Attack, DPA Attack, Prime Fields.

1 Introduction

Bilinear pairing or pairing is a new and increasingly popular way of constructing cryptographic protocols. This has resulted in the development of pairing based schemes such as identity based encryption (IBE) which are ideally used in identity aware devices. The security of such devices leads to the security of pairing computations. In the last decade, an increasingly popular form of attack known as side-channel attack (SCA) [3, 4], which exploits the weakness in implementations, have developed. SCA breaks a cryptosystems by analyzing the information that could be measured through some covert channel of a cryptoprocessor like : power consumption, time, electromagnetic radiation, fault, etc.

Pairing can be computed on different characteristic fields like binary ($\mathbb{F}_{2^m}$), trinary ($\mathbb{F}_{3^m}$), and large prime ($\mathbb{F}_p$). The security of pairing computations over first two fields against differential power analysis (DPA) attack have been described in [9] and [5], respectively. However, security analysis of pairing computations on prime fields against side-channel attack has not been considered before.

This paper explores the side-channel vulnerability of pairing computations on FPGA platform. One of the popular pairing friendly elliptic curves defined over $\mathbb{F}_p$ is the Barreto-Naehrig curve (BN curve) [11]. A dual-core pairing cryptoprocessor for BN curves has been developed on FPGA platform. The paper proposes an optimized parallel scheduling of underlying finite field operations for Tate pairing computations by the cryptoprocessor. It further observes the mathematical formula of different steps of the pairing computation and pinpoints the vulnerability against side-channel attacks. The paper then describes a differential power analysis (DPA) technique based on such vulnerability. The actual DPA attack has been mounted on FPGA platforms which ascertains the secret parameter of pairing computation. The paper then proposes a suitable computation technique for counteracting the above vulnerability.

The paper is organized as follows: section 2 provides a mathematical background of pairing computation technique. The description of pairing cryptoprocessor over prime field is given in section 3. The vulnerability of pairing computation over prime fields is pointed out in section 4. The proposed DPA attack and its counteracting technique is described in section 5. The paper is concluded in section 6.

2 Mathematical Background

Pairing is a bilinear map which is performed on a pair of elements of a group (say $\mathbb{G}_1$) to an element of another group (say $\mathbb{G}_3$). Pairings for cryptographic applications use an additive group defined over elliptic or hyperelliptic curves as $\mathbb{G}_1$ and a multiplicative group defined over an integer field as $\mathbb{G}_2$ [7]. The mapping also follow two important properties called *bilinearity* and *non-degeneracy*. Sometimes the pairing is computed on two elements from two different additive groups (say $\mathbb{G}_1$ and $\mathbb{G}_2$) and it maps to an element of a multiplicative group $\mathbb{G}_3$. The groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are in general formed by an elliptic curve over $\mathbb{F}_q$ and $\mathbb{F}_{q^k}$, where k is also known as embedding degree of the elliptic curve. The security of a pairing is based on the difficulty to solve the discrete logarithm problem in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$.

The computation efficiency of such bilinear map is also an important factor for cryptographic applications. Cryptographic pairings are efficiently computed by Miller's algorithm [1, 2] which is shown in Alg. 1. More specifically this algorithm shows the computation of Tate pairing. Several optimizations of this algorithm have been presented in [8]. The resulting algorithm proposed in [8] is called BKLS algorithm for Tate pairing computation. Other pairings like ate, R-ate are computed by similar way using different parameters other than r and by interchanging the input points [13].

The underlying elliptic curve plays an important role for achieving computation efficiency and security of a pairing computation. Active research is going on for finding out such a *pairing-friendly* elliptic curves. One of the most popular pairing-friendly elliptic curves is known as Barreto-Naehrig curves (BN curves) [11]. The BN curve is defined over a large prime field with embedding degree 12. Thus $\mathbb{G}_1$ and $\mathbb{G}_2$ in Alg. 1 are additive elliptic curve groups defined over $\mathbb{F}_p$ and $\mathbb{F}_{p^{12}}$, respectively. The pairing value $t_r(P, Q) = f \in \mathbb{G}_3$, where $\mathbb{G}_3$ is a multiplicative integer group defined over $\mathbb{F}_{p^{12}}$. For achieving 128-bit security the BN curve is defined over a 256-bit prime field.

Input: $P \in G_1$ and $Q \in G_2$.
Output: $t_r(P, Q)$.

Write r in binary : $r = \sum_{i=0}^{L-1} r_i 2^i$.
$T \leftarrow P$, $f \leftarrow 1$.
for i *from* $L-2$ *downto* 0 **do**
  $T \leftarrow 2T$.
  $f \leftarrow f^2 \cdot l_{T,T}(Q)$.
  if $r_i = 1$ *and* $i \neq 0$ **then**
    $T \leftarrow T + P$.
    $f \leftarrow f \cdot l_{T,P}(Q)$.
  end
end
return $f^{(q^k-1)/r}$.

Algorithm 1: Computing the Tate pairing.

The BN curves also admit a sextic twist [13], which means that the point Q in Alg. 1) is mapped on a point Q' defined over $\mathbb{F}_{p^2}$. Thus, the line functions $l_{T,T}(Q)$ and $l_{T,P}(Q)$ is computed over $\mathbb{F}_{p^2}$ instead of $\mathbb{F}_{p^{12}}$. Value of the line functions are represented as : $l_0 + l_1W^2 + l_2W^3$, with $l_0 \in \mathbb{F}_p$, $l_1, l_2 \in \mathbb{F}_{p^2}$, and a quadratic non-residue W over $\mathbb{F}_{p^2}$. The Miller function f is computed over $\mathbb{F}_{p^{12}}$, which is represented as : $f_0 + f_1W + f_2W^2 + f_3W^3 + f_4W^4 + f_5W^5$, with $f_i \in \mathbb{F}_{p^2}$. So in the Tate pairing computation f^2, $f \cdot l_{T,T}(Q)$, and $f \cdot l_{T,P}(Q)$ are performed on $\mathbb{F}_{p^{12}}$. Whereas all other computations are performed on $\mathbb{F}_p$ and $\mathbb{F}_{p^2}$.

The detailed procedure of pairing computation including the final exponentiation on BN curve is described in [13] and [14]. Another efficient way of computing final exponentiation is described in [15]. We use Jacobian coordinate systems for performing elliptic curve operations, where a point (X, Y, Z) corresponds to the point (x, y) in affine coordinates with $x = X/Z^2$ and $y = Y/Z^3$.

3 Pairing Crytoprocessor (PCP)

The major operations for a pairing computation are point doubling (PD), point addition (PA), line computation ($l(Q)$), f^2, and $f \cdot l(Q)$. In case of Tate pairing on BN curve, the PA and PD are performed in $\mathbb{F}_p$. Similarly, the operation $l(Q)$ is performed in $\mathbb{F}_{p^2}$ while the other two operations are performed in $\mathbb{F}_{p^{12}}$. However, the operations in these extension fields consist of a set of operations in underlying $\mathbb{F}_p$.

The current work explores the side-channel vulnerabilities of a pairing cryptoprocessor (PCP). Therefore, instead of designing a new architecture from the scratch, we implement the pairing cryptoprocessor that was proposed in [16] on FPGA platform. The work first implement a programmable core for computing all necessary $\mathbb{F}_p$ operations. Based on this programmable core we design a cryptoprocessor for pairing computation on FPGA platform. The proposed design consists two programmable cores which exploit the parallelism of Miller's algorithm. Each of the programmable cores can perform operations on $\mathbb{F}_p$ and $\mathbb{F}_{p^2}$.

We follow the formula and algorithms for the computation of asymmetric Tate pairing that are given in [13]. The major steps in pairing algorithm (Alg. 1) are *the Miller function* and *the final exponentiation*. The Miller function consists of two major steps, namely : *doubling step* and *addition step*. Here, we discuss the computation of above steps for Tate pairing over BN curve on our dual-core PCP.

The Tate pairing (t_r) over BN curve takes input points P and Q over $\mathbb{F}_p$ and $\mathbb{F}_{p^2}$, respectively. The parameter r is a 256-bit prime of Hamming weight 91. Thus, the Miller algorithm runs for 255 iterations having 255 doubling steps and 90 addition steps. There are sufficient independent operations within the doubling and addition steps which can be performed in parallel. Our dual-core PCP consists of a fixed number of functional units. Therefore, an optimization can be done based on the available functional units and the operations. In the following subsections, we describe an optimized scheduling of above steps on proposed PCP.

3.1 Computation of Doubling Step

The doubling step consists of the following computations.

- The point doubling ($2T$) operation.
- The computation of tangent line at point T ($l_{T,T}(Q)$).
- The squaring of Miller function (f^2).
- The multiplication of Miller function with line function ($f^2 \cdot l_{T,T}(Q)$).

The computation of $2T$, $l_{T,T}(Q)$, and f^2 are performed in parallel on our PCP. In Jacobian coordinates the formulae for doubling a point $T = (X, Y, Z)$ are $2T = (X_3, Y_3, Z_3)$ where $X_3 = 9X^4 - 8XY^2$, $Y_3 = (3X^2)(4XY^2 - X_3) - 8Y^4$ and $Z_3 = 2YZ$. The tangent line at T, after clearing denominators, is $l(x, y) = 3X^3 - 2Y^2 - 3X^2Z^2x + Z_3Z^2y$ [12].

In case of Tate pairing computation on BN curves, the parameters $\{x, y\} \in \mathbb{F}_{p^2}$ and $\{X, Y, Z, X_3, Y_3, Z_3\} \in \mathbb{F}_p$. Let us assume that x and y are represented as $x_0 + x_1\mathcal{U}$ and $y_0 + y_1\mathcal{U}$, where $\{x_0, x_1, y_0, y_1\} \in \mathbb{F}_p$ and $\mathcal{U}$ is an indeterminant. The above operations are performed by one of the programmable cores in the dual-core PCP by following way.

1. $t_0 \leftarrow X^2, \quad t_1 \leftarrow Y^2, \quad t_2 \leftarrow Y \cdot Z$
2. $t_3 \leftarrow (t_0)^2, \quad t_4 \leftarrow X \cdot t_1, \quad t_5 \leftarrow (t_1)^2$
3. $t_4 \leftarrow 2t_4, \quad t_6 \leftarrow 2t_3, \quad Z_3 \leftarrow 2t_2$
4. $t_4 \leftarrow 2t_4, \quad t_6 \leftarrow 2t_6, \quad t_5 \leftarrow 2t_5$
5. $t_3 \leftarrow t_3 + t_6, \quad t_5 \leftarrow 2t_5$
6. $X_3 \leftarrow t_3 - t_2, \quad t_5 \leftarrow 2t_5$
7. $t_3 \leftarrow t_4 - X_3, \quad t_7 \leftarrow t_7 + t_0$

8. $t_7 \leftarrow t_7 \cdot t_3, \quad t_4 \leftarrow Z^2, \quad t_2 \leftarrow X \cdot t_0$
9. $Y_3 \leftarrow t_7 - t_5, \quad t_1 \leftarrow 2t_1, \quad t_5 \leftarrow 2t_2$
10. $t_4 \leftarrow t_4 \cdot t_0, \quad t_0 \leftarrow t_4 \cdot Z_3$
11. $t_2 \leftarrow 2t_4, \quad t_5 \leftarrow t_2 + t_5$
12. $t_4 \leftarrow t_4 + t_2, \quad l_0 \leftarrow t_5 - t_1$
13. $l_{10} \leftarrow t_4 \cdot x_0, \quad l_{11} \leftarrow t_4 \cdot x_1$
14. $l_{20} \leftarrow t_0 \cdot y_0, \quad l_{21} \leftarrow t_0 \cdot y_1$

In the above scheduling nonlinear $\mathbb{F}_p$ operations are performed in the instructions 1, 2, 8, 10, 13, and 14. If we assume that $\mathbb{F}_p$ squaring $(s) \approx \mathbb{F}_p$ multiplication (m) then the cost of above operations is $6m$ on a programmable core in our dual-core PCP. At the same time other core starts the computation of f^2. We represent the Miller function $f \in \mathbb{F}_{((p^2)^3)^2}$ as : $(f_{0,0} + f_{0,1}\mathcal{V} + f_{0,2}\mathcal{V}^2) + (f_{0,0} + f_{0,1}\mathcal{V} + f_{0,2}\mathcal{V}^2)\mathcal{W}$, where $f_{i,j} \in \mathbb{F}_{p^2}$. The equivalent representations of f are :

$$\begin{aligned}
f &= f_0 + f_1\mathcal{W}, \text{ where} f_0, f_1 \in \mathbb{F}_{p^6}; f \in \mathbb{F}_{(p^6)^2}.\\
&= (f_{0,0} + f_{0,1}\mathcal{V} + f_{0,2}\mathcal{V}^2) + (f_{1,0} + f_{1,1}\mathcal{V} + f_{1,2}\mathcal{V}^2)\mathcal{W},\\
&\quad \text{where} f_{i,j} \in \mathbb{F}_{p^2}; f \in \mathbb{F}_{((p^2)^3)^2}.\\
&= f_{0,0} + f_{1,0}\mathcal{W} + f_{0,1}\mathcal{W}^2 + f_{1,1}\mathcal{W}^3 + f_{0,2}\mathcal{W}^4 + f_{1,2}\mathcal{W}^5\\
&\quad \text{where} f_{i,j} \in \mathbb{F}_{p^2}; f \in \mathbb{F}_{(p^2)^6}.
\end{aligned}$$

The computation of $c = f^2$ is performed in $\mathbb{F}_{p^{12}}$ using complex method by following way.

$$\begin{aligned}
v &= f_0 \cdot f_1,\\
c_0 &= (f_0 + f_1)(f_0 + \beta f_1) - v - \beta v,\\
c_1 &= 2v,
\end{aligned}$$

where v, c_0, c_1 are in $\mathbb{F}_{p^6}$ and β is a quadratic non-residue in $\mathbb{F}_{p^6}$. It requires two $\mathbb{F}_{p^6}$ multiplications. Now, one $\mathbb{F}_{p^6}$ multiplication is performed in the tower field $\mathbb{F}_{(p^2)^3}$ using Karatsuba technique by six multiplications in $\mathbb{F}_{p^2}$. Let us consider that an element $a_i \in \mathbb{F}_{p^2}$ is represented as : $a_{i0} + a_{i1}\mathcal{U}$, $a_{ij} \in \mathbb{F}_p$. The computation of $v = f_0 \cdot f_1$ on a programmable core is as follows:

1. $\tilde{v}_0 \leftarrow f_{00} \cdot f_{10}$, where $f_{00}, f_{10} \in \mathbb{F}_{p^2}$
2. $\tilde{v}_1 \leftarrow f_{01} \cdot f_{11}$, where $f_{01}, f_{11} \in \mathbb{F}_{p^2}$
3. $\tilde{v}_2 \leftarrow f_{02} \cdot f_{12}$, where $f_{02}, f_{12} \in \mathbb{F}_{p^2}$
4. $t_{10} \leftarrow f_{010} + f_{020}, \quad t_{11} \leftarrow f_{011} + f_{021}$
5. $t_{20} \leftarrow f_{110} + f_{120}, \quad t_{21} \leftarrow f_{111} + f_{121}$
6. $t_3 \leftarrow t_1 \cdot t_2$, where $t_1, t_2 \in \mathbb{F}_{p^2}$
7. $t_{10} \leftarrow \tilde{v}_{10} + \tilde{v}_{20}, \quad t_{11} \leftarrow \tilde{v}_{11} - \tilde{v}_{21}$
8. $t_{30} \leftarrow t_{30} - t_{10}, \quad t_{31} \leftarrow t_{31} - t_{11}$
9. $t_{31} \leftarrow t_{31} + t_{31}$
10. $v_{00} \leftarrow \tilde{v}_{00} - t_{31}, \quad v_{01} \leftarrow \tilde{v}_{01} + t_{30}$

11. $t_{10} \leftarrow f_{000} + f_{010}, \quad t_{11} \leftarrow f_{001} + f_{011}$
12. $t_{20} \leftarrow f_{100} + f_{110}, \quad t_{21} \leftarrow f_{101} + f_{111}.$
13. $t_3 \leftarrow t_1 \cdot t_2,$ where $t_1, t_2 \in \mathbb{F}_{p^2}$
14. $t_{10} \leftarrow \tilde{v}_{00} + \tilde{v}_{10}, \quad t_{11} \leftarrow \tilde{v}_{01} - \tilde{v}_{11}$
15. $t_{20} \leftarrow \tilde{v}_{21} + \tilde{v}_{21}$
16. $t_{10} \leftarrow t_{10} + t_{20}, \quad t_{11} \leftarrow \tilde{v}_{20} - t_{11}$
17. $v_{10} \leftarrow t_{30} - t_{10}, \quad v_{11} \leftarrow t_{31} + t_{11}$
18. $t_{10} \leftarrow f_{000} + f_{020}, \quad t_{11} \leftarrow f_{001} + f_{021}$
19. $t_{20} \leftarrow f_{100} + f_{120}, \quad t_{21} \leftarrow f_{101} + f_{121}$
20. $t_3 \leftarrow t_1 \cdot t_2,$ where $t_1, t_2 \in \mathbb{F}_{p^2}$
21. $t_{10} \leftarrow \tilde{v}_{00} + \tilde{v}_{20}, \quad t_{11} \leftarrow \tilde{v}_{01} + \tilde{v}_{21}$
22. $t_{10} \leftarrow \tilde{v}_{10} - t_{10}, \quad t_{11} \leftarrow \tilde{v}_{11} - t_{11}$
23. $v_{20} \leftarrow t_{30} + t_{10}, \quad v_{21} \leftarrow t_{31} + t_{11}$

The result $v \in \mathbb{F}_{p^6}$ is represented as : $(v_{00} + v_{01}\mathcal{U}) + (v_{10} + v_{11}\mathcal{U})\mathcal{V} + (v_{20} + v_{21}\mathcal{U})\mathcal{V}^2$, where $v_{ij} \in \mathbb{F}_p$. In the above computation, steps 1, 2, 3, 6, 13, 20 perform multiplications in $\mathbb{F}_{p^2}$. Thus the cost of $v = f_0 \cdot f_1$ is $6m$, which is computed in parallel with $2T$, $l_{T,T}(Q)$ by the proposed PCP.

The second $\mathbb{F}_{p^6}$ multiplication, *i.e.*, the computation of $(f_0 + f_1)(f_0 + \beta f_1)$ is performed by both the programmable cores, which costs only $3m$ in the PCP. Therefore, the total cost of computing $2T$, $l_{T,T}(Q)$, and f^2 by the PCP is $9m$.

The $l(Q)$ is represented as : $(l_0 + l_1\mathcal{V}) + (l_2\mathcal{V})\mathcal{W}$, where $l_0 \in \mathbb{F}_p$, $l_1, l_2 \in \mathbb{F}_{p^2}$, which is equivalent to $l_0 + l_1\mathcal{W}^2 + l_2\mathcal{W}^3$. The computation of $f \cdot l(Q)$ is performed in the tower field $\mathbb{F}_{((p^2)^3)^2}$ by following way.

$$
\begin{aligned}
f' &= f \cdot l(Q) \\
&= ((f_{0,0} + f_{0,1}\mathcal{V} + f_{0,2}\mathcal{V}^2) + (f_{1,0} + f_{1,1}\mathcal{V} + f_{1,2}\mathcal{V}^2)\mathcal{W}) \cdot \\
&\quad ((l_0 + l_1\mathcal{V}) + (l_2\mathcal{V})\mathcal{W})
\end{aligned}
$$

The top most extension is quadratic. Thus the computation of $f \cdot l(Q)$ is done by three $\mathbb{F}_{p^6}$ multiplications, which are identified as :

$$
\begin{aligned}
t_1^1 &= (l_0 + l_1\mathcal{V}) \cdot (f_{0,0} + f_{0,1}\mathcal{V} + f_{0,2}\mathcal{V}^2) \\
t_2^1 &= (l_2\mathcal{V}) \cdot (f_{1,0} + f_{1,1}\mathcal{V} + f_{1,2}\mathcal{V}^2) \\
t_3^1 &= (l_0 + (l_1 + l_2)\mathcal{V}) \cdot (((f_{0,0} + f_{1,0}) + (f_{0,1} + f_{1,1})\mathcal{V} + \\
&\quad (f_{0,2} + f_{1,2})\mathcal{V}^2)
\end{aligned}
$$

One multiplication in $\mathbb{F}_{p^6}$ using Karatsuba method requires 18 $\mathbb{F}_p$ multiplications. However, due to the sparse representation of $l(Q)$ the cost of computing t_i^1, $1 \le i \le 3$ is lesser than the actual costs of three $\mathbb{F}_{p^6}$ multiplications. Both the equations for t_1^1 and t_3^1 require only 14 $\mathbb{F}_p$ multiplications. In our parallel cryptoprocessor the above two equations are computed in parallel on two programmable cores, which costs $5m$. The computation of t_2^1 requires only nine $\mathbb{F}_p$ multiplications, which is performed on both the cores and it costs only $2m$.

Therefore, the computation of $f \cdot l(Q)$ requires 37 $\mathbb{F}_p$ multiplications, which costs only $7m$ in our PCP. Therefore, the total cost for computing the doubling step (the computation of $2T, l_{T,T}(Q), f^2$, and $f \cdot l(Q)$) of the Miller algorithm for Tate pairing on BN curve is $9m + 7m = 16m$.

3.2 Computation of Addition Step

The addition step consists of the computations of $T + P$, $l_{T,P}(Q)$, and $f \cdot l_{T,P}(Q)$. The formulae for mixed Jacobian-affine addition are the following: if $T = (X_1, Y_1, Z_1)$ is in Jacobian coordinates and $P = (X_2, Y_2)$ is in affine coordinates, then $T + P = (X_3, Y_3, Z_3)$ where $X_3 = (Y_2 Z_1^3 - Y_1)^2 - (X_2 Z_1^2 - X_1)^2 (X_1 + X_2 Z_1^2)$, $Y_3 = (Y_2 Z_1^3 - Y_1)(X_1 (X_2 Z_1^2 - X_1)^2 - X_3) - Y_1 (X_2 Z_1^2 - X_1)^3$, $Z_3 = Z_1 (X_2 Z_1^2 - X_1)$. The line through T and P is $l(x, y) = (X_2 (Y_2 Z_1^3 - Y_1) - Y_2 Z_3) - (Y_2 Z_1^3 - Y_1) x + Z_3 \cdot y$. During the addition step of Miller algorithm we compute the above operations in parallel on both cores. There are limited independent operations in this step. Therefore, there are scopes for optimizing the scheduling of operations on $\mathbb{F}_p$ arithmetic units for reducing the additional registers and related wiring. The respective scheduling is shown here.

1. $t_0 \leftarrow Y_2 \cdot Z_1,\ t_0 \leftarrow (Z_1)^2$
2. $t_0 \leftarrow t_1 \cdot t_0,\ t_1 \leftarrow t_1 \cdot X_2$
3. $t_4 \leftarrow t_1 + X_1,\ t_0 \leftarrow t_0 - Y_1,\ t_5 \leftarrow t_1 - X_1$
4. $t_3 \leftarrow (t_0)^2,\ Z_3 \leftarrow t_5 \cdot Z_1,\ t_7 \leftarrow (t_5)^2; l_{10} \leftarrow t_0 \cdot x_0,$ $l_{11} \leftarrow t_0 \cdot x_1$
5. $t_2 \leftarrow t_7 \cdot X_1,\ t_4 \leftarrow t_4 \cdot t_7,\ t_5 \leftarrow t_5 \cdot t_7; t_{10} \leftarrow t_0 \cdot X_2$
6. $X_3 \leftarrow t_3 - t_4$
7. $t_2 \leftarrow t_2 - X_3$
8. $t_2 \leftarrow t_2 \cdot t_0,\ t_4 \leftarrow Y_2 \cdot Z_3,\ t_5 \leftarrow t_5 \cdot Y_1; l_{20} \leftarrow Z_3 \cdot y_0,$ $l_{21} \leftarrow Z_3 \cdot y_1$
9. $Y_3 \leftarrow t_2 - t_5; l_0 \leftarrow t_{10} - t_4$

In the above scheduling, the nonlinear operations (multiplication and squaring) in $\mathbb{F}_p$ are performed in steps 1, 2, 4, 5, and 8. Thus, the cost of computing $T + P$, $l_{T,P}(Q)$ is $5m$ in the PCP. This computation is followed by $f \cdot l(Q)$, which costs $7m$. Therefore, the cost for evaluating the addition step is $5m + 7m = 12m$ in the PCP.

3.3 Computation of Final Exponentiation

The final exponentiation is computed by following way. It follows the optimization to factor $(p^{12} - 1)/r$ into three parts [14] and compute $f^{(p^{12}-1)/r}$ as :

$$\begin{aligned} f^{\frac{p^{12}-1}{r}} &= f^{(p^6-1) \times \frac{p^6+1}{p^4-p^2+1} \times \frac{p^4-p^2+1}{r}} \\ &= ((f^{p^6-1})^{p^2+1})^{\frac{p^4-p^2+1}{r}}. \end{aligned}$$

The computation is done by following way:

1. $f \leftarrow f^{p^6-1}$.
2. $f \leftarrow f^{p^2+1}$.
3. $a \leftarrow f^{-(6z+5)},\ b \leftarrow a^p,\ b \leftarrow a \cdot b$.
4. Compute $f^p,\ f^{p^2},\ f^{p^3}$.
5. $f \leftarrow f^{p^3} \cdot \left[b \cdot (f^p)^2 \cdot f^{p^2}\right]^{6z^2+1} \cdot b \cdot (f^p \cdot f)^9 \cdot a \cdot f^4$,

where z is a BN parameter and we choose z = 6000000000001F2D (in hexadecimal). Table 1 lists the operation costs of final exponentiation on the PCP. The power of $(p^6 - 1)$ in $\mathbb{F}_(p^6)^2$ is an easy exponentiation, which is performed by a conjugation (Frobenius) and a division [15, 10]. The operation $f^{p^6} = f_0 - f_1\mathcal{W}$. Thus, f^{p^6-1} is performed by one inversion and one multiplication in $\mathbb{F}_{p^{12}}$, which costs $29m$ on our dual-core PCP.

Table 1. Operation costs for the final exponentiation on our PCP.

Operation	cost on PCP
f^{p^6-1}	$29m$
f^{p^2+1}	$12m$
$f^{-(6z+5)}$	$480m$
$a^p,\ a \cdot b,\ f^p,\ f^{p^2},\ f^{p^3}$	$21m$
$T \leftarrow b \cdot (f^p)^2 \cdot f^{p^2}$	$24m$
$T \leftarrow T^{6z^2+1}$	$951m$
$f^{p^3} \cdot T \cdot b \cdot (f^p \cdot f)^9 \cdot f^4$	$93m$

The exponentiations f^{6z+5}, T^z and $(T^z)^{6z}$ are performed by repeated square-and-multiply. Note that $6z + 5$ and $6z$ have bitlength 66 and Hamming weight 11, while z has bitlength 63 and Hamming weight 11.

3.4 Cost for Computing Tate Pairing

In case of BN curve, r has bitlength 256 and Hamming weight 91. Thus the total cost for evaluating iterative Miller function of the Tate pairing computation is $5176m$ on our PCP. The cost for computing the final exponentiation is $1610m$. Hence, the total cost for computing a Tate pairing over BN curves by our cryptoprocessor is $6786m$, which takes $1,764,360$ cycles. The cryptoprocessor finishes one Tate pairing computation over BN curve in $35.3ms$ on a Virtex-4 FPGA platform. It consumes $52k$ slices and runs at 50 MHz clock frequency.

4 Side-channel Vulnerability

Page and Vercauteren [5] presented SPA and DPA attacks on the pairing computations performed by the Duursma-Lee algorithm [6] and the BLKS algorithm [8] over $\mathbb{F}_{3^m}$. The power consumption attack on η_T pairing computation over $\mathbb{F}_{2^m}$ is described by Kim *et al.* in [9]. However, the same in case of $\mathbb{F}_p$ has not been studied so far. This section investigates the security of pairing computations over $\mathbb{F}_p$ against power consumption attacks.

4.1 Weakness of Pairing Computations in $\mathbb{F}_p$

In the decryption step of identity-based encryption schemes, a dominant operation is $e(U, S_{ID})$, where S_{ID} is the fixed secret key, and U is a part of a ciphertext [17]. In this case side-channel attacks may try to extract the secret key from the pairing computation by repeatedly manipulating U. The Tate pairing over $\mathbb{F}_p$ consists of elliptic curve group operations (ECD and ECA), the line functions, and the Miller function [13]. The line functions as per the definition provided by Chatterjee *et al.* [12] use both the public point U and private point S_{ID}. The formula of line functions are based on the underlying $\mathbb{F}_p$ primitives.

During the addition step of Tate pairing computation the formula of the line function is $l(x, y) = (y - Y_2)Z_3 - (x - X_2)(Y_2 Z_1^3 - Y_1)$ [12]. In pairing based cryptographic schemes, the point $T = (X_1, Y_1, Z_1)$ is an intermediate resultant point of current point doubling operation, the point $U = (X_2, Y_2)$ is used as a public parameter (it could be the plain texts or messages), and $S_{ID} = (x, y)$ is used as the private key. The resultant point $(T+U)$ is represented by (X_3, Y_3, Z_3). Therefore, in such a scheme the operations $(x - X_2)$ and $(y - Y_2)$ could be exploited through side-channel attacks for finding out the x and y-coordinates of the secret point.

5 Proposed DPA Attack

In this section, we investigate differential power analysis (or DPA) attack against the subtraction $(x - X_2)$ used in the Tate pairing on elliptic curves in $\mathbb{F}_p$, where x is secret and X_2 is public and known to, or even chosen by, the attacker. The subtraction $(x - X_2)$ in $\mathbb{F}_p$ is computed by first computing $S = x - X_2$ and then the result is reduced (if required) by adding p with S. Let us assume that all operations are performed on 2's complement numbers. Therefore, the subtraction $S = x - X_2$ could be performed as: $S = \sum_{i=0}^{k} 2^i s_i = \sum_{i=0}^{k-1} 2^i x_i + \sum_{i=0}^{k-1} 2^i \bar{X}_{2_i} + 1$, where k represents the bit length of operands (x, X_2) and $\bar{X}_{2_i}$ corresponds to the 1's complement of X_{2_i}. The subtraction is started from the least significant bit (or LSB) by computing *sum* and *carry* bits iteratively. The formula for i-th carry bit is: $c_i = x_i \bar{X}_{2_i} \oplus x_i c_{i-1} \oplus \bar{X}_{2_i} c_{i-1}$. Similarly, the i-th sum bit is computed as: $s_i = x_i \oplus \bar{X}_{2_i} \oplus c_{i-1}$ for $k - 1 \leq i \leq 0$ with $c_{-1} = 1$.

The proposed DPA attack works by following way. The attacker first collects the power consumption traces of n number of randomly chosen public point U.

We consider the simplified Hamming weight model for power leakage [18]. In this model, power consumption depends on the Hamming weight of the data being processed. Thus, we can express the power consumption W as:

$$W = \varepsilon H + \eta \tag{1}$$

where H, ε, and η represent the Hamming weight of the intermediate data, the incremental amount of power for each extra 1 in the Hamming weight, and the noise, respectively. We assume that the average of noise η is zero.

Let W be the power consumption associated with the subtraction operation $(x - X_2)$. We start from the LSB and iteratively find all bits of the x-coordinate of the secret point $S_{ID} = (x, y)$. To recover the i-th bit of x, we guess that $x_i = 0$ and divide power consumptions into two sets by $\bar{X}_{2_i} \oplus c_{i-1}$.

$$P_k = \{\ W \ \mid\ \bar{X}_{2_i}\ \oplus\ c_{i-1}\ =\ k\} \quad \text{with } k = \{0, 1\}$$

Thus, the differential power consumption is:

$$\Delta = < P_1 - P_0 > .$$

If the guess is correct, then the averages of P_1 and P_0 are, $\varepsilon(M+1)/2$ and $\varepsilon(M-1)/2$, where M corresponds to the bit length of S. Thus, if $\Delta > 0$, we know that $x_i = 0$; otherwise, the averages of P_1 and P_0 is $\varepsilon(M-1)/2$ and $\varepsilon(M+1)/2$. Thus, if $\Delta < 0$ then $x_i = 1$. There should be a positive peak when $x_i = 0$ and a negative peak when $x_i = 1$.

In summary, since the subtraction operation $(x - X_2)$ of line function in pairing computation is vulnerable to the proposed attack, we can recover x. Next, we can obtain the value of y-coordinate of the secret point S_{ID} by solving the curve equation.

5.1 Mounting the DPA on FPGA Platform

We perform the actual DPA attack on aforementioned pairing cryptoprocessor (or PCP). The PCP is implemented on a customized FPGA board for power analysis. We put an one ohm resistor between the VCCint pin of the FPGA chip and the on board voltage regulator. We measure the current drawn through that resistor during pairing computation by a current probe. The specification of the probe is Tektronix current probe (serial number B014316). We use the probe with a TCPA300 power amplifier in standby mode. The measured power is displayed and stored in a Tektronix TDS5032B Digital Phosphor Oscilloscope. We develop software tools to automate the whole process for varying inputs. The power consumptions are measured in terms of mV which is varying around $\pm 5mV$. The power signal is sampled at $12.5MS/s$.

We choose an x with $x_0 = 0$ and perform $(x - X_2)$ for 2000 times with 2000 different randomly chosen X_2. The respective power consumptions are stored in 2000 one dimensional vectors. Now we differentiate the the power vectors in two sets namely P_1 and P_0. A vector will be in set P_1 if $\bar{X}_{2_0} \oplus c_{-1} = 1$; *i.e.*, $X_{2_0} = 1$.

Otherwise, the vector will be in set P_0. For computing the differential power consumption we subtract the average of P_0 vectors (means) from the average of P_1 vectors. We say this differential power consumption vector as difference-of-means which is represented by Δ. Then we accumulate the samples of Δ and plot it. The respective difference-of-means is depicted in Fig. 1(a), which shows a positive peak as expected for $x_0 = 0$.

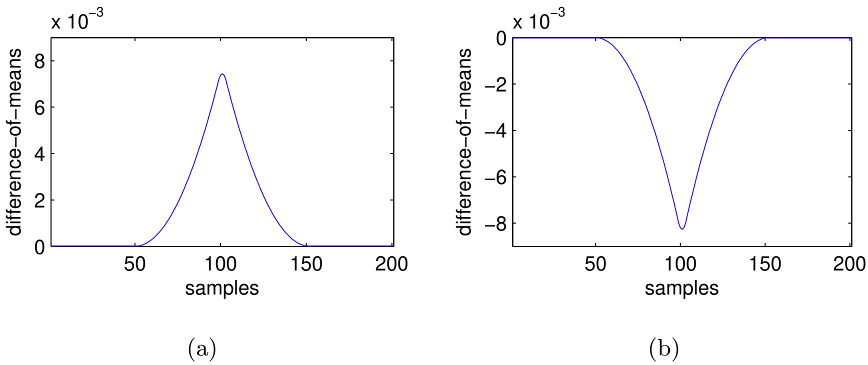

Fig. 1. The correlation between LSB and corresponding average power differences of an addition in $\mathbb{F}_p$. (a) for $x_0 = 0$ and (b) for $x_0 = 1$.

The same experiment has been repeated for another x with $x_0 = 1$. The difference-of-means in this case is plotted in Fig. 1(b). In this case the expectation of $< P_1 - P_0 >$ is negative and we got the result as expected with 2000 random X_2.

Above experimental result ensures that an attacker can easily mount the DPA attack on pairing computation over $\mathbb{F}_p$. After finding out the LSB, DPA can be performed for second LSB, and so on. The same power traces could be utilized for finding out all secret bits. The differentiation of power vectors into two sets depending on the current value of $(\bar{X}_{2_i} \oplus c_{i-1})$ upto the generation of the difference-of-means will be repeated for finding out each of the secret bits. Thus, above DPA attack iteratively finds out all bits of the x-coordinate of secret S_{ID}. After obtaining the x-coordinate, the value of y-coordinate could be obtained easily by solving the underlying elliptic curve equation.

5.2 Proposed Counteracting Technique

In the pairing computation, the secret point is only used for computing the line functions. The formula of the line function during doubling step of the Miller algorithm over $\mathbb{F}_p$ is as follows:

$$l_{T,T}(x, y) = Z_3Z^2y - 2Y^2 - 3X^2(Z^2x - X),$$

where $T = (X, Y, Z)$ be the intermediate resultant point of Miller algorithm while $2T = (X_3, Y_3, Z_3)$ [12].

The formula of $l_{T,T}(x, y)$ is using the secret point $S_{ID} = (x, y)$ of identity based encryption (IBE) [17]. But, it does not use the public point $U = (X_2, Y_2)$. Therefore, this function could not be exploited by any side-channel attacks.

The second line function $l_{T,P}(x, y)$ is computed during the addition step of the Miller algorithm. In IBE scheme P is replaced by U. The formula of $l_{T,P}(x, y)$ is:

$$l_{T,U}(x, y) = (y - Y_2)Z_3 - (x - X_2)(Y_2 Z_1^3 - Y_1),$$

where $T(X_1, Y_1, Z_1)$ is the intermediate result of doubling step and (X_3, Y_3, Z_3) represents the addition result of $T + U$. In this line computation formula both public point $U = (X_2, Y_2)$ and private point $S_{ID} = (x, y)$ are used. The computation of $l_{T,U}(x, y)$ is the main weakness of pairing computation over $\mathbb{F}_p$ against side-channel attacks. The DPA attack described above can easily find out the x and y-coordinates of private point S_{ID} by exploiting the above formula.

The main drawback of the above formula is that the public and private parameters are directly involved to perform an $\mathbb{F}_p$ operation. The side-channel attack thus exploit the respective $\mathbb{F}_p$ operation for finding out the secret bits by manipulating public parameter U. To counter act on such computation against side-channel attacks it could be computed by following way.

$$l_{T,P}(x, y) = (X_2(Y_2 Z_1^3 - Y_1) - Y_2 Z_3) - (Y_2 Z_1^3 - Y_1)x + Z_3 \cdot y.$$

The above computation technique does not have any $\mathbb{F}_p$ primitive which is performed on one public parameter and one private parameter. The attacker may try to exploit the power consumption of the cryptoprocessor during the computation of $l_{T,P}(x, y)$. The private parameter x in the above formula is multiplied with an unknown parameter $(Y_2 Z_1^3 - Y_1)$. Therefore, no difference-of-mean can be computed for identifying the secret bits of x.

The second secret parameter y is multiplied with Z_3 in the modified computation of $l_{T,P}(x, y)$. The parameter Z_3 is computed by executing the formula $Z_3 = Z_1(X_2 Z_1^2 - X_1)$ which ensures Z_3 is unknown due to the unknown temporary point $T(X_1, Y_1, Z_1)$. Therefore, no difference-of-mean value can be computed based on the specific bits of Z_3 for identifying the secret bits of y. Thus, the proposed counteracting technique protects both x and y coordinates of secret point S_{ID}, which ensures the security of pairing computation against DPA attack.

6 Conclusion

This paper has demonstrated an optimized scheduling of Tate pairing computation over BN curve on a dualcore pairing cryptoprocessor. The computation cost for One Tate pairing achieving 128-bit security on FPGA platform is $35.3ms$. The paper further analyzes the effect of covert power channel of the pairing cryptoprocessor against physical security. The paper has pinpointed the vulnerability of such pairing computation against DPA attack. The actual DPA has been performed on FPGA platform and respective vulnerability has been demonstrated. Finally, the paper has proposed a suitable counteract to protect secret point of pairing computation against DPA attack.

References

1. V.S. Miller. Short Programs for Functions on Curves. unpublished manuscript, 1986.
2. V.S. Miller. The Weil pairing, and its efficient calculation. Journal of Cryptology, Vol. 17, pp. 235–261, 2004.
3. P.C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems. Advances in Cryptology - CRYPTO'96, USA, LNCS 1109, pp. 104–113, Springer, 1996.
4. P.C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. Advances in Cryptology - CRYPTO'99, USA, LNCS 1666, pp. 388–397, 1999.
5. D. Page and F. Vercauteren. Fault and side-channel attacks on pairing based cryptography. Cryptology ePrint Archive, Report 2004/283. http://eprint.iacr.org/.
6. I. Duursma and H. Lee. Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$. ASIACRYPT 2003, LNCS 2894, pp. 111–123, Springer, 2003.
7. J. Hoffstein, J. Pipher, and J.H. Silverman. An introduction to mathmatical cryptography. Springer, 2008.
8. P.S.L.M. Barreto, H. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. CRYPTO 2002, LNCS 2442, pp. 354–368, 2002.
9. T.H. Kim, T. Takagi, D.G. Han, H. Kim, and J. Lim. Power analysis attacks and countermeasures on ηT pairing over binary fields. ETRI Journal, Vol. 30, No. 1, pp. 68–80, 2008.
10. M. Naehrig, P.S.L.M. Barreto, and P. Schwabe. On compressible pairings and their computation. AFRICACRYPT'08, LNCS 5023, pp. 371–388, 2008.
11. P.S.L.M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. SAC '05. LNCS 3897, pp. 319–331, Springer, Heidelberg, 2006.
12. S. Chatterjee, P. Sarkar, and R. Barua. Efficient computation of Tate pairing in projective coordinate over general characteristic fields. Information Security and CryptologyICISC 2004, LNCS 3506, pp. 168–181, 2005.
13. D. Hankerson, A. Menezes, and M. Scott. Software implementation of pairings. In: Joye, M., Neven, G. (eds.) Identity-Based Cryptography, 2008.
14. A.J. Devegili, M. Scott, and R. Dahab. Implementing cryptographic pairings over Barreto-Naehrig curves. Pairing '07. LNCS 4575, pp. 197–207, 2007.
15. M. Scott, N. Benger, M. Charlemagne, L.J.D. Perez, and E.J. Kachisa. On the final exponentiation for calculating pairings on ordinary elliptic curves. Pairing '09, LNCS 5671, pp. 78–88, 2009.
16. S. Ghosh, D. Mukhopadhyay, and D. Roychowdhury. High speed flexible pairing cryptoprocessor on FPGA Platform. Pairing'10, LNCS 6487, Japan, 2010.
17. D. Boneh and M.K. Franklin. Identity-based encryption from the Weil pairing. CRYPTO 2001, LNCS 2139, pp. 213–229, Springer, Heidelberg, 2001.
18. T.S. Messerges. Using second-order power analysis to attack DPA resistant software. CHES 2000, LNCS 1965, pp. 238–251, Springer, Berlin, Germany, 2000.