Practical validation of several fault attacks against the Miller algorithm

Nadia El Mrabet[1], Jacques Fournier[2],
Louis Goubin[3], Ronan Lashermes[2,3], *Marie Paindavoine*[4,5].

1 - LIASD, Paris 8, France. 2 - CEA Tech, DPACA/LSAS, Gardanne, France.
3 - UVSQ-PRiSM, Versailles, France. 4 - Orange Labs, Applied Crypto Group, France.
5 - LIP, Lyon, France.

FDTC 2014
September 23, 2014

Content

A brief description of pairings

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ be three finite groups of elements having the same prime order r.
A pairing is a map e:

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$$

which is:

- Bilinear - $e([a]P, [b]Q) = e(P, Q)^{a,b}$,
- Non degenerate -
  $\forall P \in \mathbb{G}_1 \,(\text{resp. } \forall Q \in \mathbb{G}_2)\,,\ e(P, Q) = 1 \Rightarrow Q = \mathcal{O} \,(\text{resp. } P = \mathcal{O})$,
- Efficiently computable.

Very useful in cryptography: identity-based encryption, short signatures, tripartite Diffie-Hellman.

Construction of pairings

$e(P, Q)$: maps two subgroups of $\mathcal{E}(\overline{\mathbb{F}_p})$ of points of order r in μ_r (r-th roots of unity).

$[r]P = [r]Q = \mathcal{O}$

Smallest k such as $\mu_r \subset \mathbb{F}_{p^k}$: embedding degree.

Two steps:
▷ The Miller Algorithm ▷ The Final Exponentiation

This talk focuses on the Miller algorithm: it outputs a function $f_{r,Q}$ which admits Q as a zero of order r and $[r]Q = \mathcal{O}$ as a pole.

Computation of $f_{r,Q}$

Recursive construction with a double-and-add structure.

Recurrence relations

- Initialization :

$$f_{1,Q} = 1.$$

- Doubling step :

$f_{2i,Q} = f_{i,Q}^2 \cdot h_1$ with h_1 the equation of the tangent at the point $[2i]Q$.

- Addition step :

$$f_{i+1,Q} = f_{i,Q} \cdot h_2 \text{ with } h_2 \text{ the line equation } ([i]Q, Q).$$

Algorithm

Algorithm 1 The Miller algorithm for the Ate Pairing

```
Input : $r = \sum_{i=0}^{t} r_i 2^i$, $P \in \mathbb{G}_1$ and $Q \in \mathbb{G}_2$.
Output : $f = f_{r,Q}(P) \in \mathbb{G}_3$.
 1: $T \leftarrow Q$
 2: $f \leftarrow 1$
 3: for $i \leftarrow t-1$ to $0$ do
 4:     $f \leftarrow f^2 \cdot h_1(P)$ ($h_1$ is the tangent equation at the point $T$)
 5:     $T \leftarrow [2]T$
 6:     if $r_i = 1$ then
 7:         $f \leftarrow f \cdot h_2(P)$ ($h_2$ is the line $(Q, T)$ equation)
 8:         $T \leftarrow T + Q$
 9:     end if
10: end for
11: return $f$
```

Content

Implementation parameters

A wide variety of pairings and curves.
We choose to attack the Ate pairing, on Barreto-Naehrig curves.

Our implementation

$y^2 = x^3 + 5$
Embedding degree: $k = 12$
254 bits p and r.
$\mathbb{G}_1$ subgroup of $\mathcal{E}(\mathbb{F}_p)$ and $\mathbb{G}_2$ subgroup of $\mathcal{E}(\mathbb{F}_{p^{12}})$

Our goal

In most protocols, $e(P, Q)$ takes one public argument and one secret.
We want to recover the secret point (either P or Q).

Field extensions and twisted curves

$\mathbb{F}_{p^{12}}$: $\mathbb{F}_{p^2}$-vector space. We have a unique decomposition:

$$\forall R \in \mathbb{F}_{p^{12}},\ R = \sum_{i=0}^{5} R_i w^i,\ R_i \in \mathbb{F}_{p^2}, w \in \mathbb{F}_{p^{12}} \setminus \mathbb{F}_{p^6}.$$

When the points are in $\mathcal{E}(\mathbb{F}_{p^{12}})$: heavy representation & computation!

We use the twisted curve (degree 6) $\mathcal{E}'$: there exists a bijection from the points of $\mathbb{G}_2$ to the points of $\mathcal{E}'(\mathbb{F}_{p^2})$.

$\rightsquigarrow$ The coordinates of Q now lie in $\mathbb{F}_{p^2}$

Recovering the secret point

From the tangent equation

$$h_1(P) = \left(3X_T^3 - 2Y_T^2\right) \cdot w^6 + \left(2Y_T Z_T^3 y_P\right) \cdot w^3 - \left(3X_T^2 Z_T^2 x_P\right) \cdot w^4,$$

we obtain the following system in $\mathbb{F}_{p^2}$:

$$\begin{cases} R_0 = \left(3X_T^3 - 2Y_T^2\right) \cdot u & (1) \\ R_3 = 2Y_T Z_T^3 y_P & (2) \\ R_4 = -3X_T^2 Z_T^2 x_P & (3) \end{cases}$$

Curve equation: $X_T^3 = Y^2 + 5$

Equation 2 : Y_T as Z_T polynomial.

By substition, we obtain an univariate polynomial in Z_T : we recover the coordinates of $T = [j]Q$, j known, then Q.

Fault models

Loop skip [PV06,EM09]

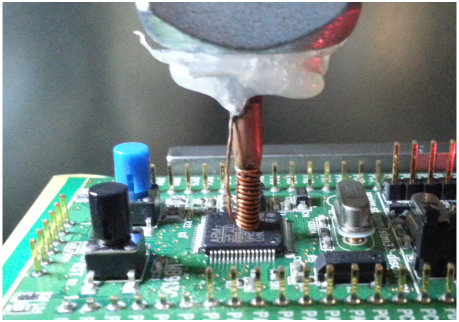
- We target the loop counter
- Obtain two algorithm executions with successive iterations numbers
- If the second iteration is double only
- The quotient of results is $h_1(P)$

Controlled add [WS07]

- Targets the last iteration
- Fault a modular addition while computing $h_1(P)$
- If we know the fault value
- The correct/faulty result ratio allows us to recover $h_1(P)$

Experimental validation

Are we able to experimentally achieve these fault models?



Experimentations

Targeting a Cortex-M3 microcontroller computing an (home-made) Ate pairing with an Electromagnetic fault injection bench.

Experimental results

It is possible to induce an instruction skip in the microcontroller

The two fault models were implemented

When removing the final exponentiation to get the output of the Miller loop...

...we recovered the secret point!

Content

Countermeasures

Blinding countermeasures :

1. Coordinates blinding: Replace the jacobian coordinates $Q = (X_Q : Y_Q : Z_Q)$ by $(\lambda^2 X_Q : \lambda^3 Y_Q : \lambda Z_Q) (\lambda \neq 0, 1)$ with a random $\lambda \in \mathbb{F}_{p^2}$ before computation.
2. Miller variable blinding: At each iteration, multiply f by a random element of $\mathbb{F}_{p^d}, d < k, d | k$. The final exponentiation maps the masks onto one.
3. Additive Blinding: For a random $M \in \mathbb{G}_2$, one computes $e(P, Q) = e(P, Q + M) \cdot e(P, -M)$. It does not affect the result as pairings are bilinear.
4. Multiplicative blinding: We have $e(\alpha P, \beta Q) = e(P, Q)^{\alpha\beta}$. One chooses α, β with $\alpha \cdot \beta = 1 \pmod{r}$.

Efficiency of the countermeasures

First two originally designed against side-channel analysis. Also proposed against fault attacks.

Last two designed specifically for fault attacks. But their overhead is more important.

Can we use the first two in order to circumvent fault attacks?

A relation between the blinded execution and the correct one

$$Q = (\lambda^2 X_Q : \lambda^3 Y_Q : \lambda Z_Q) \in \mathcal{E}(\mathbb{F}_p).$$

For the doubling step we have the following relation:

$$\begin{cases} T = (\lambda^{2i} X_T, \lambda^{3i} Y_T, \lambda^i Z_T) \\ h_1^{(\lambda)} = \lambda^{24i} h_1. \end{cases}$$

And for the addition step:

$$\begin{cases} P = (\lambda^2 X_P, \lambda^3 Y_P, \lambda Z_P) \\ T = (\lambda^{2i} X_T, \lambda^{3i} Y_T, \lambda^i Z_T) \\ h_2^{(\lambda)} = \lambda^{9i+12} h_2. \end{cases}$$

Hence, for some integer a:

$$f_{r,P}^{(\lambda)} = \lambda^a \cdot f_{r,P}$$

Loop counter fault model

Two executions, hence two masks, but adds only one more unknown:

$$h_1(P)^{(\lambda)} = \frac{\lambda_1^a}{\lambda_2^b} \cdot h_1(P).$$

We denote $L = \frac{\lambda_1^a}{\lambda_2^b}$ the new unknown:

$$h_1(P)^{(\lambda)} = L \cdot (R_0 \cdot w^6 + R_3 \cdot w^3 + R_4 \cdot w^4),$$

By identification we have:

$$\begin{cases} R_0^{(\lambda)} & = LR_0 \\ R_3^{(\lambda)} & = L \cdot R_3 \\ R_4^{(\lambda)} & = L \cdot R_4. \end{cases}$$

Solving the polynomial system

The system can be solved with a Gröbner basis computation.

We first recover the coordinates of $T = [j]Q$, j known.

It allows us to recover the coordinates of Q.

We are able to bypass the Miller variable blinding with the same method.

Conclusion

It is not (yet) an attack on a whole pairing computation

But realistic fault models that we can realize in implementations

The efficient countermeasures imply important overhead.

Thank you!

Any questions?