Inverting the final exponentiation of Tate pairings on ordinary elliptic curves using faults

Ronan Lashermes[1,2], Jacques Fournier[1], Louis Goubin[2]

August 23rd 2013

CHES 2013

CEATech[1], Gardanne, France, UVSQ PRiSM[2], Versailles, France

Teaser

We will see:

- How to recover 1536 bits of the secret with one fault!
- The full secret (3072 bits) recovery in 3 faults.
- How to revert a surjection by finding the unique preimage used in the computation.

Pairing Based Cryptography

Pairing :

Bilinear maps for cryptography

Why?

Allow new cryptographic schemes.
Ex: Identity Based Encryption.

Elliptic curves

Curve E

A point (X, Y) on the curve satisfies:

$$E : Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$$

Fields for X and Y

- p a big prime.
- $\mathbb{F}_p$ a finite field.
- r a prime divisor of $card(E(\mathbb{F}_p))$.
- k the smallest integer such that $r|p^k - 1$ (k is the embedding degree).
- $\mathbb{F}_{p^k}$ an extension field.
- μ_r the group formed by the r^{th} roots of unity in $\mathbb{F}_{p^k}$ ($\mu_r \subset \mathbb{F}_{p^k}$).

Tate pairing

Definition

Reduced Tate pairing:

$$\begin{cases} e_T : E(\mathbb{F}_p)[r] \times E(\mathbb{F}_{p^k})/rE(\mathbb{F}_{p^k}) & \rightarrow \quad \mu_r \\ \qquad\qquad\qquad\qquad\qquad (P, Q) & \rightarrow \quad f_{r,P}(Q)^{\frac{p^k-1}{r}} \end{cases}$$

Tate pairing

Definition

Reduced Tate pairing:

$$\begin{cases} e_T : E(\mathbb{F}_p)[r] \times E(\mathbb{F}_{p^k})/rE(\mathbb{F}_{p^k}) & \rightarrow & \mu_r \\ (P, Q) & \rightarrow & f_{r,P}(Q)^{\frac{p^k-1}{r}} \end{cases}$$

Computed with two main steps:

Miller Algorithm

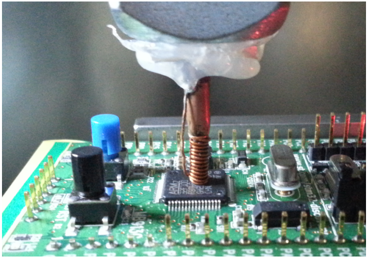
$f_{r,P}(Q)$

Final Exponentiation

$\cdot^{\frac{p^k-1}{r}}$

Fault attacks

- Fault attack = circuit perturbation to alter the cryptographic algorithm.



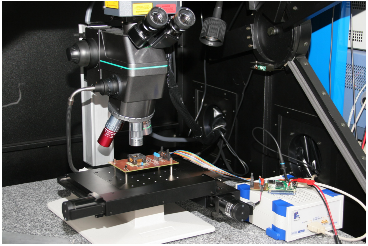


Figure : EM and laser benches

Fault model: instruction skip.

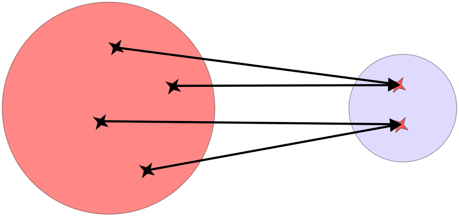
Introduction to the final exponentiation

Definition

Let f be in $\mathbb{F}_{p^k}$, $FE(f) = f^{\frac{p^k-1}{r}}$.

Properties

It is a surjective application. To invert the final exponentiation is to find the correct, unique, preimage of a surjection.



Are pairings resistant wrt fault attacks?

Fault attacks on the Miller algorithm (big prime characteristic)

There are several attacks possible on the Miller algorithm. But they all require that the attacker know the result of the Miller algorithm prior to the final exponentiation.

Are pairings resistant wrt fault attacks?

Fault attacks on the Miller algorithm (big prime characteristic)

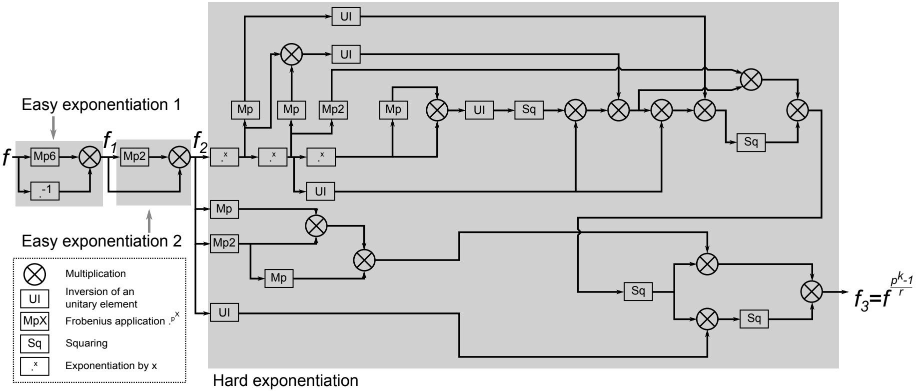
There are several attacks possible on the Miller algorithm. But they all require that the attacker know the result of the Miller algorithm prior to the final exponentiation.

Impossible?

The reason why pairings are considered resistant wrt fault attacks: the attacker cannot access to $f_{r,P}(Q)$, the result of the Miller algorithm.

(e.g. with $k = 12$ and $\log_2(r) \approx 256$, each element of μ_r has $\approx 2^{2816}$ preimages!)

Implementation



From "On the final exponentiation for calculating pairings on ordinary elliptic curves" by Scott et al. in Pairing 2009

Computation of the final exponentiation

Decomposition

$k = 2 \cdot d$

$$\frac{p^k - 1}{r} = (p^d - 1) \cdot \frac{p^d + 1}{r}$$

Notations

$$f_1 = f^{p^d - 1}$$

$$f_3 = f_1^{\frac{p^d+1}{r}} = f^{\frac{p^k-1}{r}}$$

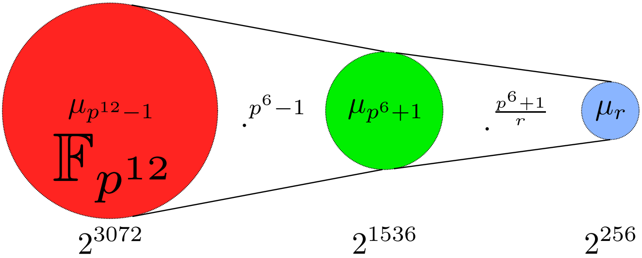


Figure : Final exponentiation groups, security level 128 bits

Roots of unity

$$f^{p^k-1} = 1$$

$$f_1^{p^d+1} = 1$$

$$f_3^r = 1$$

$f \in \mu_{p^k-1}, f_1 \in \mu_{p^d+1}, f_3 \in \mu_r$.

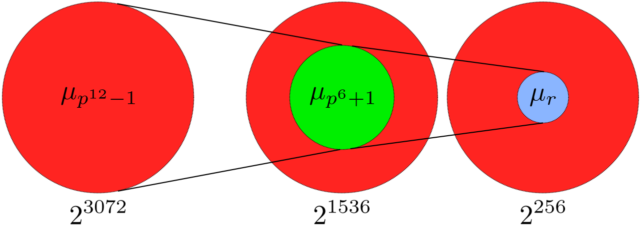


Figure : Final exponentiation, element representation (representation size $\neq$ entropy!)

Extension construction

$$\mathbb{F}_{p^k} = \mathbb{F}_{p^d}[w]/(w^2 - v)$$

So $f = g + h \cdot w$ et $w^2 = v, g, h \in \mathbb{F}_{p^d}$.

Redundancy relations

$$f^{p^d+1} = g^2 - v \cdot h^2 \in \mathbb{F}_{p^d}$$

$$f_1^{p^d+1} = g_1^2 - v \cdot h_1^2 = 1$$

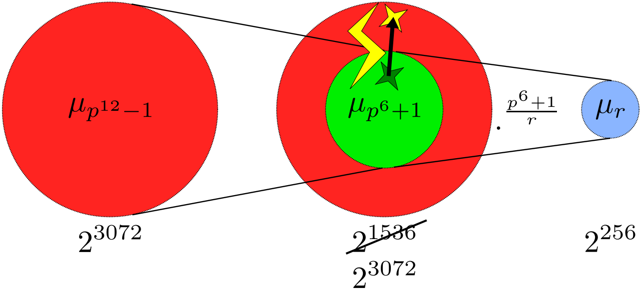


Figure : First fault location

e_1 value known to the attacker.

Notations

$$f_1^* = f_1 + e_1 \notin \mu_{p^d+1}$$

$$f_1^* = (g_1 + e_1) + h_1 \cdot w$$

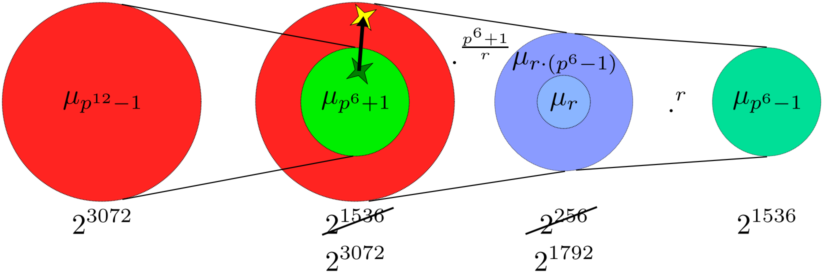


Figure : Fault effect

Result

$$
\begin{aligned}
(f_1^*)^{p^d+1} &= (f_3^*)^r \neq 1 \\
&= (g_1 + e_1)^2 - v \cdot h_1^2 \\
&= g_1^2 - v \cdot h_1^2 + 2 \cdot e_1 \cdot g_1 + e_1^2 \\
&= 1 + 2 \cdot e_1 \cdot g_1 + e_1^2
\end{aligned}
$$

We have found f_1

g_1

$$g_1 = \frac{(f_1^*)^{p^d+1} - 1 - e_1^2}{2 \cdot e_1}$$

h_1

Two possible values

$$h_1^+ = \sqrt{\frac{g_1^2 - 1}{v}} \; ; \; h_1^- = -\sqrt{\frac{g_1^2 - 1}{v}}$$

easy to check.

Guessing e_1

If you do not know e_1...

... then you have to guess it.

A guess on e_1 gives two f_1 candidates. The attacker then compare $f_1^{\frac{p^d+1}{r}}$ against f_3 and $(f_1 + e_1)^{\frac{p^d+1}{r}}$ against f_3^*.
In this case, there is a low false positive rate: $1/r^2$.

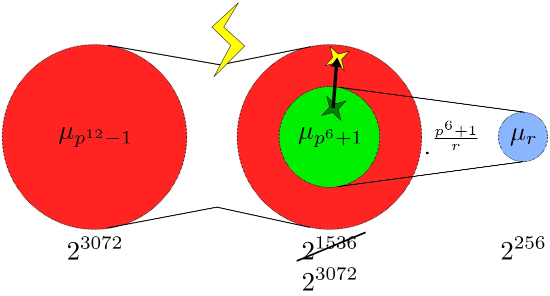


Figure : Second fault location $e_2 \in \mathbb{F}_{p^d}$

e_2 value known to the attacker.

New redundancy relation & fault

$$f_1 = f^{p^d-1} = \bar{f} \cdot f^{-1}$$

$$\frac{g_1 - 1}{v \cdot h_1} = -\frac{h}{g} = K$$

$$f_1^* = \bar{f} \cdot (f^{-1} + e_2)$$

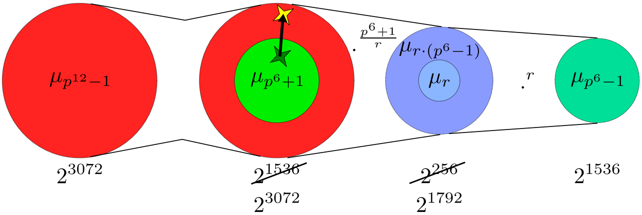


Figure : Second fault effect

Fault propagation

$$f_1^* = f1 + \Delta_{f_1}$$
$$\Delta_{f_1} = \bar{f} \cdot e_2$$
$$\Delta_{f_1} = e_2 \cdot g - e_2 \cdot h \cdot w$$

We have found f

Quadratic equation

$$\begin{cases} (f_1^*)^{p^d+1} & = (g_1 + e_2 \cdot g)^2 - v \cdot (h_1 - e_2 \cdot h)^2 \\ h & = -g \cdot K \end{cases}$$

Which gives

$$g^2 \cdot e_2^2 \cdot (1 - v \cdot K^2) + g \cdot 2 \cdot e_2 \cdot (g_1 - v \cdot K \cdot h_1) + 1 - (f_1^*)^{p^d+1} = 0$$

Guessing e_2

Problem

Each supposition about e_2 gives a f which is valid wrt all our observations.

Guessing e_2

Problem

Each supposition about e_2 gives a f which is valid wrt all our observations.
$\Rightarrow$ We have to repeat the second step with at least another fault ($\neq e_2$).

Guessing e_2

Problem

Each supposition about e_2 gives a f which is valid wrt all our observations.
$\Rightarrow$ We have to repeat the second step with at least another fault ($\neq e_2$).

Sets of candidates

$$\begin{aligned} e_2 &\in \{1, 2, \ldots, 10\} & \rightarrow f &\in \{fc_1, fc_2, \ldots, fc_{10}\} \\ e_2' &\in \{1, 2, \ldots, 10\} & \rightarrow f &\in \{fc_1', fc_2', \ldots, fc_{10}'\} \end{aligned}$$

f is in the intersection of the two sets of candidates.
For $e_2, e_2' \in [\![1, m]\!]$,

$$\#intersection = \left\lfloor m \cdot \frac{\gcd(e_2, e_2')}{\max(e_2, e_2')} \right\rfloor$$

There is a trick to accelerate the computation of the intersection (in paper).

Summary of the attack

How to invert the final exponentiation?

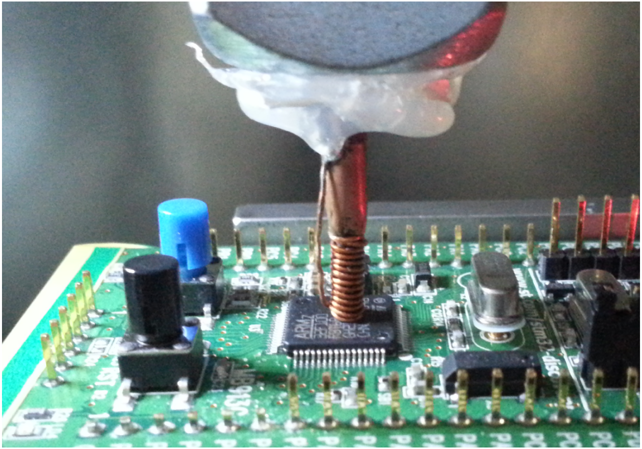
1. Correct execution.
2. Fault injection, f_1 is recovered.
3. Fault injection, f candidates are recovered.
4. Repeat step 3 until f is recovered.

f can be found with only 3 faults!

The attack has been validated using simulations.

Perspectives - what next?

- Practical attack on the final exponentiation.
- Achieve a complete fault attack on a pairing.



Thank you! Questions?

"La montagne Sainte Victoire vue de Gardanne" by Cezanne