Inverting the Final Exponentiation of Tate Pairings on Ordinary Elliptic Curves Using Faults

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

[1] CEA-TechReg, Gardanne, France
ronan.lashermes@cea.fr, jacques.fournier@cea.fr
[2] UVSQ-PRiSM, Versailles, France
louis.goubin@prism.uvsq.fr

Abstract. The calculation of the Tate pairing on ordinary curves involves two major steps: the Miller Loop (ML) followed by the Final Exponentiation (FE). The first step for achieving a full pairing inversion would be to invert this FE, which in itself is a mathematically difficult problem. To our best knowledge, most fault attack schemes proposed against pairing algorithms have mainly focussed on the ML. They solved, if at all, the inversion of the FE in some special 'easy' cases or even showed that the complexity of the FE is an intrinsic countermeasure against a successful full fault attack on the Tate pairing. In this paper, we present a fault attack on the FE whereby the inversion of the final exponentiation becomes feasible using 3 independent faults.

Keywords: Tate pairing, Ate pairing, final exponentiation, fault attacks.

1 Introduction

Pairing-Based Cryptography (PBC) uses bilinear mappings (or pairings) to construct cryptographic schemes. Identity-Based Encryption (IBE) [1], anonymous IBE, one round Diffie-Hellman key exchanges or searchable encryption [2] constitute the scope of promising applications of PBC, accentuating the need for secure implementations. An exhaustive literature is currently available on the choice of curves and associated parameters for secure efficient PBC implementations as well as analyses covering the issues linked to the resistance of such implementations against side channel and fault attacks [3, 4]. A pairing calculation consists of two major steps namely the *Miller Loop* (ML) and the *Final Exponentiation* (FE). Most of the existing work covering fault attacks against pairing calculations focuses on the ML [5–7], even stating in some cases that in practice the presence of the complex FE after the ML reduces the practical significance of such fault attacks [6, 7].

In this paper, we propose a scheme where a fault attack, using only three faulty outputs and a correct one, is used to calculate the input to the "complex" final exponentiation despite the fact that the FE inversion has been defined as a mathematical hard problem [8]. To our best knowledge, this is the first published fault attack on the FE which allows to "un-nest" the complex calculations

G. Bertoni and J.-S. Coron (Eds.): CHES 2013, LNCS 8086, pp. 365–382, 2013.

involved in this second part of a pairing calculation, thus opening the way to the future building of complete fault attack schemes against Tate-like pairings over ordinary curves.

We first begin by laying some of the basic notations and concepts used to describe PBC. We then detail the structure of the Tate pairing before reviewing existing fault attack schemes in order to understand how our scheme complements them. After that we explain our attack, review some of the limitations that we have identified (up to now), discuss its practical feasibility before proposing countermeasures and concluding the paper.

2 Pairing Based Cryptography

Detailed descriptions of the ins and outs of a pairing implementation can be found in [9]. Below we shall introduce the notations and concepts required to understand the proposed fault attack scheme against the Tate pairing on ordinary curves.

Let p be a big prime number and $E(\mathbb{F}_p)$ an ordinary elliptic curve over $\mathbb{F}_p$. Let r be a prime divisor of $card\,(E(\mathbb{F}_p))$. We define the embedding degree k of E with respect to r as the smallest integer such that $r|p^k-1$. Additionally, $r|\Phi_k(p)$, with Φ_k the k-th cyclotomic polynomial [10, 11].

A pairing maps two points over subgroups of order r of an elliptic curve $E\left(\mathbb{F}_{p^k}\right)$ to the multiplicative field $\mathbb{F}^*_{p^k}$. As an example, the Tate pairing is defined as

$$\langle .,. \rangle_r : E(\mathbb{F}_p)[r] \times E(\mathbb{F}_{p^k})/([r]E(\mathbb{F}_{p^k})) \rightarrow \mathbb{F}^*_{p^k}/\left(\mathbb{F}^*_{p^k}\right)^r$$

In order to work with actual values rather than equivalence classes (i.e. guarantee the uniqueness of the pairing result), the output of the Tate pairing is mapped to μ_r with a final exponentiation to the power of $\frac{p^k-1}{r}$. The group μ_r is formed by the r-th roots of unity in $\mathbb{F}_{p^k}$: $\mu_r = \{x \in \mathbb{F}^*_{p^k} | x^r = 1\}$. All Tate pairing outputs in the same equivalence class are mapped to a unique value in μ_r. The reduced Tate pairing is then defined as

$$\begin{aligned} t_r : E(\mathbb{F}_p)[r] \times E(\mathbb{F}_{p^k})/([r]E(\mathbb{F}_{p^k})) &\rightarrow \mu_r \\ (P,Q) &\mapsto \langle P,Q \rangle_r^{\frac{p^k-1}{r}} \end{aligned}$$

The evaluation of $\langle P,Q \rangle_r$ is called the Miller Loop (ML) and the exponentiation to the power $\frac{p^k-1}{r}$ is the Final Exponentiation (FE). Several other pairings on ordinary curves derived from the Tate pairing, such as the Ate pairing [12] or the Optimal Ate pairing [13], have this final exponentiation step, meaning that our attack also works on such alternative implementations.

3 The Security of PBC from a Fault Attack Perspective

In a practical case, like in Boneh & Franklin's IBE [1], the decryption scheme involves the calculation of a pairing between a 'public' point and a 'secret' one. The attacker's aim in this case is to recover the secret point in order to impersonate the legitimate owner of the secret key. The security of a pairing implementation is usually measured by the ability for an attacker to recover one of the two input points, knowing the second input point and the pairing result. This problem is called *Fixed Argument Pairing Inversion (FAPI)* which can in-turn be subdivided into two problems: first the *Exponentiation Inversion* (EI) problem which consists in recovering the output of the ML; then the *Miller Inversion* (MI) problem which aims at recovering the target point knowing the result of the ML. These problems have been recently studied in [14] and [15] based on the previous works of [16] and [17].

The *EI* problem can be stated as finding the **unique** correct preimage of the reduced Tate pairing under the FE knowing one input point and the reduced final result. Indeed, one may find the correct preimage knowing the final reduced result with the additional information brought by the Miller Loop and the knowledge of one input point.

Here we will not discuss about the Miller Loop and we will consider only the final exponentiation on a random element f of $\mathbb{F}^*_{p^k}$. In this context, knowing the result of the exponentiation does not allow an attacker to recover f purely mathematically since he cannot distinguish the correct preimage f from all other preimages in this many-to-one relationship (with as many as $\frac{p^{12}-1}{r}$ preimages, e.g. $\approx 2^{2816}$ preimages for $k = 12$).

To find the result of the Miller Loop is not enough to solve the *FAPI* problem since the *MI* problem still needs to be solved. But our approach brings us a step closer to achieving the full pairing inversion by showing that it is possible to invert the final exponentiation with fault attacks.

3.1 Fault Attacks against PBC

Our attack exploits the information brought by faults injected during the execution of the FE on a computing device.

A fault attack aims at disrupting the expected behaviour of an algorithm. Such an attack may alter the data flow (corrupting a data) or the control flow (e.g. modifying the number of iterations in a loop). Fault injection techniques range from clock glitches, voltage glitches to more advanced techniques such as the use of a laser beam or an electromagnetic pulse. A fault injection is not an easy task as several parameters (intensity, spatial localisation, time of injection...) have to be monitored in order to achieve the desired faulty behaviour without damaging the target [18, 19].

Fault attacks on pairing have already been discussed in various contexts [5–7]. Schemes have been proposed in order to reverse the Miller Loop by altering the number of iterations in the loop [5, 7] or by altering the value at the last iteration [6]. In these papers, to complete their attacks, the authors propose

strategies to invert the FE: they consider pairings with either a simple FE [5] or without any FE [6] at all, which are not relevant to Tate-like pairings on ordinary curves. For the latter situation, the authors in [6] even conclude that the complex FE is an inherent deterrent to the use of fault attacks on the entire pairing scheme over ordinary curves since the exponentiation could not be reversed. For such a situation, in [7], the authors propose to "short-circuit" the entire exponentiation routine but this approach is tricky as it means that the attack must not only bypass an entire routine but must at the same time have access to the result of the Miller Loop. In this paper we propose what is in our opinion a more "realistic" approach where, by using 3 independent faults (on 3 executions of the same pairing calculation), the FE itself can be reversed.

3.2 Fault Model

In the binary representation, a fault effect can be represented with a bit-XOR operation (bit-flip faults), bit-AND (stuck at zero faults) or bit-OR (stuck at one faults) on the data (or control) value. One has then to translate the fault effect as a valid mathematical operation in our field. As a consequence a fault value is intrinsically dependent on the binary representation of an element in that field.

A fault must have a manageable limited effect. Typically, a simple fault model is to consider random faults on a machine word-size data. An example would be a random single-byte fault on an 8-bit microcontroller. Such a simple model is compatible with some of the latest fault injection techniques proposed in the literature: for example in [19], the authors illustrate how an electromagnetic pulse might corrupt the execution of an instruction, modelled by an "instruction skip". With this method we can adopt a fault model where a data corruption, of the size of a machine word, can be achieved by the "skipping" of an instruction.

To accommodate the diversity of existing platforms, we chose to consider a random fault value on one word of an l-bit architecture. It can be modelled as the addition with e where $-2^l < e < 2^l$ if the fault occurs on the least significant word of the binary representation of the field element. If a fault occurs on another word (e.g. on the i-th word), the fault value e should be multiplied by $2^{i \cdot l}$ to model the fault effect correctly (it may be necessary if the attacker wants to inject two different fault values on the same intermediate result using "instruction skips").

For clarity, from now on we shall consider the fault model to be such that $0 < e < 2^l$ (it is a valid model for random stuck at 1 faults on one word). The extension of our fault attack to negative error values is straight forward since we guess the value of e in our equations.

3.3 Motivations for Fault Attacks against the FE

Several elements hint at the potential efficiency of a fault attack on the FE. First the result of the reduced Tate pairing is in μ_r which contains r elements. But this result is represented as an element of the full $\mathbb{F}_{p^k}$ field. To give an example, on a Barreto-Naehrig (BN) curve over $\mathbb{F}_{p^{12}}$ with $\log_2(p) \approx 256$, an element in μ_r has

$\log_2(r) \approx 256$ bits of information but it is represented over $12 \cdot \log_2(p) \approx 3072$ bits! This means that $3072 - 256 = 2816$ bits are redundant.

A tempting approach for the attacker would be to use these bits to learn information about the targeted preimage by inducing a fault that diverts intermediate values from their subgroup.

4 Inverting the FE Using Fault Attacks

As mentioned in [6], the FE in Tate-like pairings is a complex calculation. We show how precisely chosen faults can help in finding the critical intermediate values to finally reverse the entire exponentiation.

Our work is based on the algorithms proposed by Scott *et al.* in [10]. It focuses on FE in fields with an even embedding degree. We shall write $d = k/2$. The optimisation technique described in [10], still widely used in pairing implementations, is based on the decomposition of the FE into three stages. As $\frac{p^k-1}{r}$ can be re-written as $\frac{p^k-1}{r} = \left(p^d - 1\right) \cdot \frac{p^d+1}{\Phi_k(p)} \cdot \frac{\Phi_k(p)}{r}$, the FE can be performed as a succession of three exponentiations. Two are "easy" (with $\left(p^d - 1\right)$ and $\frac{p^d+1}{\Phi_k(p)}$) since they rely on exponentiations to the power p^n for some n and can hence be computed with the help of the Frobenius endomorphism which has a low computational cost. The last step is the so-called "hard exponentiation" (because it cannot rely on the use of the Frobenius) and is the exponentiation to the power $\frac{\Phi_k(p)}{r}$. For example, with $k = 12$, we have

$$\frac{p^{12} - 1}{r} = \left(p^6 - 1\right) \cdot \left(p^2 + 1\right) \cdot \frac{p^4 - p^2 + 1}{r} \tag{1}$$

Let f, the result of a Miller Loop, be a random value in $\mathbb{F}^*_{p^k}$. We name these intermediate results of each exponentiation

$$f_1 = f^{p^d-1} \; ; \; f_2 = f_1^{\frac{p^d+1}{\Phi_k(p)}} \text{ and } f_3 = f_2^{\frac{\Phi_k(p)}{r}} \tag{2}$$

The attacker knows the result f_3 and wants to recover f. Note that f_1, f_2 and f_3 belong to different subgroups of $\mathbb{F}^*_{p^k}$. Since $f \in \mathbb{F}^*_{p^k}$, the following equations hold

$$f^{p^k-1} = 1 \; ; \; f_1^{p^d+1} = 1 \; ; \; f_2^{\Phi_k(p)} = 1 \text{ and } f_3^r = 1 \tag{3}$$

Thus $f_1 \in \mu_{p^d+1}$, $f_2 \in \mu_{\Phi_k(p)}$ and $f_3 \in \mu_r$. These subgroups have sizes $p^d + 1$, $\Phi_k(p)$ and r respectively. As an example for $k = 12$, f_1 contains ≈ 1536 bits of information, f_2 contains ≈ 1024 bits of information and f_3 contains ≈ 256 bits of information.

4.1 Recovering f_1

In this section we shall show how a fault on the intermediate value f_1 can help to retrieve its value.

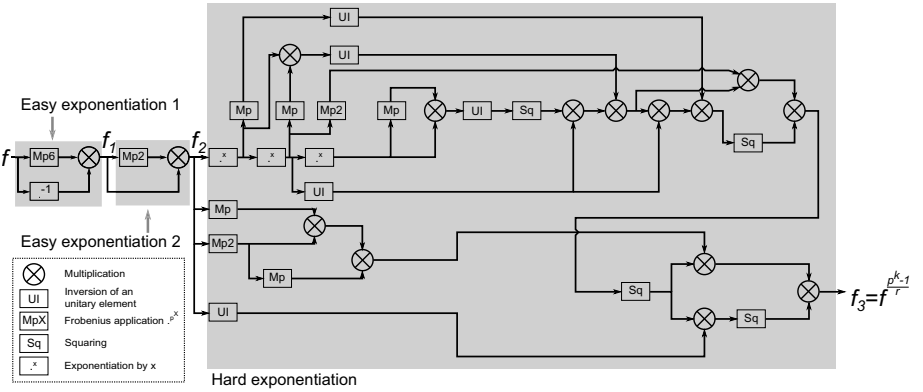

Fig. 1. Algorithm for the FE in $\mathbb{F}_{p^{12}}$. x is a public parameter of the curve

Extracting a Candidate. We first have the following lemma.

Lemma 1. *Let $\mathbb{F}_{p^k} = \mathbb{F}_{p^d}[w]/(w^2 - v)$ be the construction rule for the $\mathbb{F}_{p^k}$ extension field. v is a quadratic nonresidue in $\mathbb{F}_{p^d}$ and is a public parameter. Let $x \in \mathbb{F}_{p^k}$ be such that $x = g + h \cdot w$ with $g, h \in \mathbb{F}_{p^d}$. Then $x^{p^d+1} = g^2 - v \cdot h^2 \in \mathbb{F}_{p^d}$.*

Proof. We have $x^{p^d} = g - h \cdot w$ since $x^{p^d} = (g + h \cdot w)^{p^d} = g^{p^d} + h^{p^d} \cdot w^{p^d} = g + h \cdot (-w)$. As a result $x^{p^d+1} = x^{p^d} \cdot x = (g - h \cdot w) \cdot (g + h \cdot w) = g^2 - w^2 \cdot h^2 = g^2 - v \cdot h^2$ since $w^2 = v$ □

Let $f_1 = g_1 + h_1 \cdot w$ with $g_1, h_1 \in \mathbb{F}_{p^d}$. We have

$$f_1^{p^d+1} = f_3^r = 1 \tag{4}$$

Thus by Lemma 1

$$g_1^2 - v \cdot h_1^2 = 1 \tag{5}$$

But equation (4) holds only because $f_1 \in \mu_{p^d+1}$. Let $e \in \mathbb{F}_{p^d}$ be a fault injected on f_1 (say during the multiplication producing f_1 or during the loading of f_1 for the second "easy" exponentiation - see Fig. 1.) such that the faulty value f_1^* equals

$$f_1^* = f_1 + e \notin \mu_{p^d+1} \tag{6}$$

We consider that the fault e occurs only on the g_1 component[1] (which is compatible with our fault model if $2^l < p^6$), i.e

$$f_1^* = (g_1 + e) + h_1 \cdot w \tag{7}$$

[1] If on h_1, the same argumentation can be done.

$(f_1^*)^{p^d+1}$ can be computed by the attacker using the measured faulty result f_3^* since r is public knowledge

$$(f_1^*)^{p^d+1} = (f_3^*)^r \in \mathbb{F}_{p^d} \tag{8}$$

Using Lemma 1 and equations (5) and (7) we have

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

Finally, g_1 can be written as:

$$g_1 = \frac{(f_1^*)^{p^d+1} - 1 - e^2}{2 \cdot e} \tag{9}$$

Two possible values for h_1 can hence be calculated using equation (5):

$$h_1^+ = \sqrt{\frac{g_1^2 - 1}{v}} \; ; \; h_1^- = -\sqrt{\frac{g_1^2 - 1}{v}} \tag{10}$$

Verifying the Candidates. The two candidates $f_1^+ = g_1 + h_1^+ \cdot w$ and $f_1^- = g_1 + h_1^- \cdot w$ can thus be verified by checking if $(f_1^+)^{\frac{p^d+1}{r}} = f_3$ or $(f_1^-)^{\frac{p^d+1}{r}} = f_3$. If the value of e is unknown, the attacker must guess the injected fault. For each guess, two candidates are computed and checked. A candidate is equal to the correct f_1 only when the correct e is guessed.

In our fault model, $0 < e < 2^l$ thus $2^l - 1$ attempts have to be made to find f_1 with 100% certainty. At this stage one may wonder what is the chance that the attacker finds a valid f_1 candidate (and an error value) which fits all his observations but is not equal to f_1 (i.e. a false positive). The f_1 candidate is noted f_{1c} and the corresponding error guessed is e_c.

$$f_{1c}^{p^d+1} = 1 \tag{11}$$

$$(f_{1c} + e_c)^{p^d+1} = (f_3^*)^r \tag{12}$$

But, the attacker observes $f_3 = f_1^{\frac{p^d+1}{r}}$ and $f_3^* = (f_1 + e)^{\frac{p^d+1}{r}}$. The question is what is the probability that $f_{1c} \neq f_1$ but that

$$f_3 = f_{1c}^{\frac{p^d+1}{r}} \tag{13}$$

$$f_3^* = (f_{1c} + e_c)^{\frac{p^d+1}{r}} \tag{14}$$

Using equation (11), the probability that equation (14) is verified can be inferred as being equal to $1/r$ for a random f_{1c} in μ_{p^d+1}. Indeed we already know that $f_{1c}^{\frac{p^d+1}{r}}$ is in μ_r and $1/r$ is the probability that one random element in μ_{p^d+1} maps to a fixed value f_3 in μ_r. Similarly, from equation (12), we can deduce that the probability for equation (13) to be verified is equal to $1/r$ for a random f_{1c} in $\mathbb{F}_{p^k}^*$ since $(f_3^*)^r = (f_{1c} + e_c)^{p^d+1} \in \mu_{p^d-1}$. Thus $f_3^* \in \mu_{r\cdot(p^d-1)}$ and $(f_3^*)^r$ has r preimages in $\mu_{r\cdot(p^d-1)}$. As a consequence, the probability that we obtain the correct preimage is $1/r$.

We can combine these two probabilities and evaluate the probability of having an incorrect candidate for f_1 that matches the attacker's observations. The probability that a random candidate satisfying equations (11) and (12) also satisfies equations (13) and (14), corresponding to the observations of the attacker, is equal to $1/r^2$. In the case where $k = 12$, typically $r \approx 2^{256}$, the probability of finding a valid candidate which is not equal to f_1 is $1/2^{512}$.

Hence we have shown how a fault injected on f_1 can be used to recover the latter's value, with a high probability, using the correct output f_3 and the faulty one f_3^* of the FE.

4.2 Recovering f

Knowing the value of f_1, we shall now see how to recover f.

Extracting a Candidate. The strategy is to use similar equations to the ones used previously and to include the new information about f_1 obtained by the attacker. Proof of the lemma is in Appendix A.

Lemma 2. *Let $f = g + h \cdot w$, $f^{-1} = g' + h' \cdot w$ and $f_1 = g_1 + h_1 \cdot w$.*
Then $\frac{g_1-1}{v\cdot h_1} = \frac{h'}{g'} = -\frac{h}{g} \Leftrightarrow f_1 = f^{p^d-1}$.

In the following, let K be the known value (known because we know g_1 and h_1 from f_1 found previously) $K = \frac{g_1-1}{v\cdot h_1} = -\frac{h}{g}$.

As a consequence, the knowledge of f_1 allows to find random preimages by taking a random $g \in \mathbb{F}_{p^d}$ and choosing $h = -K \cdot g$.

To recover f, the attacker creates a new fault $e_2 \in \mathbb{F}_{p^d}$ during the inversion in the first easy exponentiation (see Fig. 1.). Then

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

Let Δ_{f_1} be the difference: $\Delta_{f_1} = f_1^* - f_1 = \bar{f} \cdot e_2$. Since $e_2 \in \mathbb{F}_{p^d}$, we can write $\Delta_{f_1} = \Delta_{g_1} + \Delta_{h_1} \cdot w$ with

$$\Delta_{g_1} = e_2 \cdot g \text{ and } \Delta_{h_1} = -e_2 \cdot h$$

As f_1^* is not in μ_{p^d+1} with a high probability equal to $(1 - \frac{1}{2^{p^d-1}})$, the attacker can compute $(f_1^*)^{p^d+1} = (f_3^*)^r \in \mathbb{F}_{p^d}$.

In this case

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

which gives the quadratic equation (using the relation $h = -g \cdot K$)

$$g^2 \cdot e_2^2 \cdot (1 - v \cdot K^2) + g \cdot 2 \cdot e_2 \cdot (g_1 - v \cdot K \cdot h_1) + 1 - (f_1^*)^{p^d+1} = 0 \qquad (15)$$

We then solve this equation to obtain two solutions for g:

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

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

h can be computed with g and K: $h = -g \cdot K$. Thus we have two potential candidates for f.

Verifying the Candidates. Even if e_2 is unknown, this procedure gives two candidates by guessing e_2. Now, whether this guess is correct or wrong, every potential candidate f_c has the following property: $f_c^{p^d-1} = f_1$ and therefore $f_c^{\frac{p^k-1}{r}} = f_3$. The attacker has found several valid preimages of f_3 and has to decide which is the correct one.

By checking whether $(\bar{f}_c \cdot (f_c^{-1} + e_2))^{\frac{p^d+1}{r}}$ is equal to the faulty result f_3^* allows to eliminate one of the two candidates for this guess of e_2. We finally obtain one candidate for each e_2 guessed and this candidate satisfies all observations made by the attacker. Finally we obtain a set of candidates of the same size as the set of possible error values.

The attacker has then to generate a third fault e_3, different from e_2, at the same location as the last one and intersect the two sets of candidates to find the correct one. Unfortunately, this intersection does not necessarily contain only one element. We can evaluate the size of this intersection set.

First we can neglect the probability that a random element of $\mathbb{F}_{p^k}^*$ maps to f_1 (the probability is $1/(p^d+1)$). Equation (15) outputs one f candidate f_{c1} by guessing $e_2 = 1$. Then the set of candidates for this error is $\{f_{c1}, f_{c2}, \ldots, f_{c(2^l-1)}\}$ with f_{ci} corresponding to the guess $e_2 = i$. If we replace the product $g \cdot e_2$ by $\frac{g}{i} \cdot (i \cdot e_2)$ in equation (15), we can see that the previous set can be rewritten as $\{f_{c1}, \frac{f_{c1}}{2}, \ldots, \frac{f_{c1}}{2^l-1}\}$.

Similarly with e_3, equation (15) outputs one f candidate f'_{c1} by guessing $e_3 = 1$ and then $f'_{ci} = f'_{c1}/i$. The second set of candidates is $\{f'_{c1}, \frac{f'_{c1}}{2}, \ldots, \frac{f'_{c1}}{2^l-1}\}$.

Let e_{2t} and e_{3t} be the two faults truly injected. Since the correct value f is in the two sets of candidates, first equal to f_{c1}/e_{2t} then equal to f'_{c1}/e_{3t}, we have

$$f = \frac{f'_{c1}}{e_{3t}} = \frac{f_{c1}}{e_{2t}} \tag{16}$$

Writing $a = \frac{e_{2t}}{e_{3t}}$, equation (16) can be transformed into $f'_{c1} = f_{c1}/a$. The second set of candidates can be rewritten as $\{\frac{f_{c1}}{a}, \frac{f_{c1}}{2a}, \ldots, \frac{f_{c1}}{(2^l-1)a}\}$. Thus a same candidate is in the two sets each time the equation

$$a \cdot i = j \tag{17}$$

is satisfied with $i, j \in [\![1, 2^l - 1]\!]$. In our fault model, we can take e_{2t} and e_{3t} as elements in $\mathbb{N}$ and the number of solutions to this equation becomes $\left\lfloor (2^l - 1) \cdot \frac{\gcd(e_{2t}, e_{3t})}{\max(e_{2t}, e_{3t})} \right\rfloor$ as shown in Appendix B.

Finally the size of the intersection, which also contains the correct candidate, is

$$\#intersection = \left\lfloor (2^l - 1) \cdot \frac{\gcd(e_{2t}, e_{3t})}{\max(e_{2t}, e_{3t})} \right\rfloor \tag{18}$$

and the number of wrong candidates is

$$\#intersection - 1 = \left\lfloor (2^l - 1) \cdot \frac{\gcd(e_{2t}, e_{3t})}{\max(e_{2t}, e_{3t})} \right\rfloor - 1 \tag{19}$$

The intersection of the sets of candidates obtained with e_2 and with e_3 contains at least one element if we get the two guesses correct once.

The computational cost of f recovery is low since the attacker has to use the procedure to recover a candidate through equation (15) only once per fault injected with guesses $e_2 = 1$ and $e_3 = 1$.

Then he stores the corresponding candidates and computes the ratio $a = f_{c1}/f'_{c1}$. Finally he solves equation (17), trying all $i \in [\![1, 2^l - 1]\!]$ and checking that $a \cdot i \in [\![1, 2^l - 1]\!]$, which provides e_{2t} and e_{3t} (only solutions if there is no wrong candidate). With e_{2t}, he computes $f = f_{c1}/e_{2t}$. The memory used in the recovery of f is just one element of $\mathbb{F}_{p^k}$ per fault injected.

We cannot avoid the occurrence of wrong candidates. In order to conclude our attack we must have a unique candidate which satisfies all our observations.

If more than one candidate is contained in the intersection of the two sets then other faults must be generated at the same location until one candidate only matches all the observations of the attacker.

4.3 Summary of Our Fault Attack on the Tate Pairing's FE

At least four executions of the **same** pairing on the computing device are required to perform our attack.

1. The computation is executed normally. The attacker stores f_3 the correct result of the exponentiation.

2. A first fault is created on f_1 according to Section 4.1. The attacker memorizes f_3^*, a first faulted result. f_1 is found using equations (9) and (5).
3. A second fault e_2 is created during the inversion in the first easy exponentiation according to Section 4.2. The attacker stores f_3^*, the faulted result and extracts a candidate f_{c1} for f guessing $e_2 = 1$ with equation (15) and Lemma 2.
4. Similarly to the previous step, a third fault $e_3 \neq e_2$ is created. With the faulted result f_3^*, the attacker extracts a new candidate f'_{c1} for f guessing $e_3 = 1$. The value $a = f_{c1}/f'_{c1}$ is then computed. A pair (i, j) solution to the equation $a * i = j$ with $i, j \in [\![1, 2^l - 1]\!]$ allows him to compute $f = f_{c1}/j$.

If several pairs (i, j) are found, more faults may be needed to ensure the uniqueness of the candidate for f. The important feature of this scheme is that only one fault per execution is needed to recover f, no double or triple faults.

4.4 Practical Feasibility of Our Attack

This attack scheme has been experimentally checked with Sagemath [20] in $\mathbb{F}_{p^{12}}$ with parameters identical to [9]. Our fault model was the injection of a random e with $0 < e < 2^l$.

For a random $f \in \mathbb{F}_{p^k}^*$, we simulated 1000 fault injections for "f_1 recovery" with a random fault $e \in [\![1, 2^{10} - 1]\!]$ and we made $2^{10} - 1$ guesses on the fault value per injection. As a result, f_1 was correctly found for every fault injection and no wrong candidate was observed.

Similarly, we simulated "f recovery" knowing f_1. Two different errors in $[\![1, 2^l - 1]\!]$ were injected for 100 fault injections, first for $l = 7$ and then for $l = 10$. The number of wrong candidates reached, in average, 4.87 for $l = 7$ and 5.66 for $l = 10$. These examples show that even when we "loosen" the constraints on the possible errors (from 2^7 to 2^{10}) the number of wrong candidates, on average, does not increase dramatically. But of course, the computational cost of the attack increases with 2^l. A detailed example of an implementation of the attack is presented in Appendix C.

5 Countermeasures

So far in the literature, most countermeasures proposed against fault attacks on pairings focus on protecting the Miller Loop for the good reason that it has been the main target of the fault attacks [5, 21, 22]. With our attack on the FE, we hope that other efficient countermeasures shall be proposed by the community in addition to the suggestions made below.

Inversion of Unitary Elements: In some implementations, an efficient countermeasure is already present. Indeed since normally $f_1 \in \mu_{p^d+1}$, this element is called "unitary" and has the following property: $f_1^{-1} = \bar{f}_1$. As a consequence, all inversions besides the first one (necessary to compute f_1) are replaced by a

simple conjugation which has a far lower computational cost. As a consequence a fault injected on f_1 cannot be exploited since the resulting output is not equal to the expected value $(f_1^*)^{\frac{p^d+1}{r}}$. The conclusion is that implementations should ensure that the inversions of unitary elements are always replaced with conjugations. Additionally, the use of a Boolean variable stating if the element is unitary and deciding which code (inversion or conjugation) is used for the inversion of an element should be avoided since this could then become a target in order to allow our fault injection. As an example, this latter Boolean variable is implemented in the classic Miracl library [23].

Compressed Representation: A generalization of the previous countermeasure is to use a compressed representation of the elements during the exponentiation as shown in [24, 25]. The effect is similar to the previous countermeasure. A fault attack on an implementation with the compressed representation would have to be specifically designed in order to work.

Checking Subgroup Membership: It is possible to deter this attack by checking the subgroup membership of intermediate values. As an example, f_1 should be in μ_{p^d+1}. To check this membership (checking $f_1^{p^d+1} = 1$), one has to compute $f_1^{p^d+1}$ at the price of a conjugation and a multiplication in $\mathbb{F}^*_{p^k}$. Similarly it should be possible to check that $f_2^{\Phi_k(p)} = 1$ and $f_3^r = 1$.

6 Conclusion and Perspectives

The possibility to invert the final exponentiation with a fault attack has been shown. Even if we don't have any strong restriction on the errors injected, recovering the input of the FE with a high probability is feasible. Our experimentations with Sagemath [20] allowed us to propose bounds on the number of wrong candidates obtained with this attack.

To settle the feasibility of inverting the FE with a fault attack, we must now demonstrate that our attack scheme can be implemented in practice.

The next step from an attacker's perspective would be to perform a full attack on pairing which would definitely settle pairings vulnerability to fault attacks. One possibility to achieve this is to consider double faults - two faults during one execution of the pairing: one to invert the Miller Loop according to [5] or [7] and another in the FE to access the faulted value of the Miller Loop. The possibility of this attack scheme is yet to be proven but does not seem out of reach [26].

Acknowledgements. This work was partially founded by the French Agence Nationale de la Recherche (ANR) through the ECLIPSE project. We thank N. El Mrabet, H. Le Bouder and G. Reymond for their helpful comments and discussions. We would also like to thank the anonymous reviewers for their constructive comments.

References

1. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil pairing. SIAM J. of Computing 32(3), 586–615 (2003)
2. Dutta, R., Barua, R., Sarkar, P.: Pairing-Based Cryptographic Protocols: A Survey. Cryptology ePrint Archive, Report 2004/064 (2004), `http://eprint.iacr.org/`
3. El Mrabet, N., Di Natale, G., Flottes, M.L., Rouzeyre, B., Bajard, J.C.: Differential Power Analysis against the Miller Algorithm. Technical report, Published in Prime 2009, IEEE Xplore (August 2008)
4. Whelan, C., Scott, M.: Side channel analysis of practical pairing implementations: Which path is more secure? In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 99–114. Springer, Heidelberg (2006)
5. Page, D., Vercauteren, F.: A Fault Attack on Pairing-Based Cryptography. IEEE Transactions on Computers 55(9), 1075–1080 (2006)
6. Whelan, C., Scott, M.: The Importance of the Final Exponentiation in Pairings when considering Fault Attacks. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 225–246. Springer, Heidelberg (2007)
7. El Mrabet, N.: What about Vulnerability to a Fault Attack of the Miller's algorithm During an Identity Based Protocol? In: Park, J.H., Chen, H.-H., Atiquzzaman, M., Lee, C., Kim, T.-h., Yeo, S.-S. (eds.) ISA 2009. LNCS, vol. 5576, pp. 122–134. Springer, Heidelberg (2009)
8. Vercauteren, F.: The Hidden Root Problem. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 89–99. Springer, Heidelberg (2008)
9. Beuchat, J.-L., González-Díaz, J.E., Mitsunari, S., Okamoto, E., Rodríguez-Henríquez, F., Teruya, T.: High-Speed Software Implementation of the Optimal Ate Pairing over Barreto–Naehrig Curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 21–39. Springer, Heidelberg (2010)
10. Scott, M., Benger, N., Charlemagne, M., Dominguez Perez, L.J., Kachisa, E.J.: On the Final Exponentiation for Calculating Pairings on Ordinary Elliptic Curves. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 78–88. Springer, Heidelberg (2009)
11. Barreto, P.S.L.M., Kim, H.Y., Lynn, B., Scott, M.: Efficient algorithms for pairing-based cryptosystems. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 354–369. Springer, Heidelberg (2002)
12. Hess, F., Smart, N., Vercauteren, F.: The Eta Pairing Revisited. IEEE Transactions on Information Theory 52(10), 4595–4602 (2006)
13. Vercauteren, F.: Optimal Pairings. IEEE Transactions on Information Theory 56(1), 455–461 (2010)
14. Kim, S., Cheon, J.H.: Fixed Argument Pairing Inversion on Elliptic Curves. Cryptology ePrint Archive, Report 2012/657 (2012), `http://eprint.iacr.org/`
15. Kanayama, N., Okamoto, E.: Approach to Pairing Inversions Without Solving Miller Inversion. IEEE Transactions on Information Theory 58(2), 1248–1253 (2012)
16. Galbraith, S., Hess, F., Vercauteren, F.: Aspects of Pairing Inversion. IEEE Transactions on Information Theory 54(12), 5719–5728 (2008)
17. Satoh, T.: On Pairing Inversion Problems. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 317–328. Springer, Heidelberg (2007)

18. Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., Whelan, C.: The Sorcerer's Apprentice Guide to Fault Attacks. Proceedings of the IEEE 94(2), 370–382 (2006)
19. Dehbaoui, A., Dutertre, J.M., Robisson, B., Tria, A.: Electromagnetic Transient Faults Injection on a Hardware and a Software Implementations of AES. In: FDTC, pp. 7–15. IEEE (2012)
20. Stein, W., et al.: Sage Mathematics Software (Version 5.5). The Sage Development Team (2012), `http://www.sagemath.org`
21. Ozturk, E., Gaubatz, G., Sunar, B.: Tate Pairing with Strong Fault Resiliency. In: Proceedings of FDTC 2007, pp. 103–111. IEEE Computer Society (2007)
22. Ghosh, S., Mukhopadhyay, D., Chowdhury, D.: Fault Attack and Countermeasures on Pairing Based Cryptography. International Journal Network Security 12, 21–28 (2011)
23. Certivox: Miracl library, v 5.6.1 (2012), `https://certivox.com/solutions/miracl-crypto-sdk/`
24. Naehrig, M., Barreto, P.S.L.M., Schwabe, P.: On compressible pairings and their computation. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 371–388. Springer, Heidelberg (2008)
25. Aranha, D.F., Karabina, K., Longa, P., Gebotys, C.H., López, J.: Faster explicit formulas for computing pairings over ordinary curves. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 48–68. Springer, Heidelberg (2011)
26. Van Woudenberg, J., Witteman, M., Menarini, F.: Practical Optical Fault Injection on Secure Microcontrollers. In: 2011 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pp. 91–99 (September 2011)

A Proof of Lemma 2

$f_1 = f^{p^d-1} \Rightarrow \frac{g_1-1}{v \cdot h_1} = \frac{h'}{g'} = -\frac{h}{g}$

Proof.

$$f_1 = \bar{f} \cdot f^{-1} = (f - 2 \cdot h \cdot w) \cdot f^{-1} = f \cdot f^{-1} - 2 \cdot h \cdot w \cdot f^{-1} = 1 - 2 \cdot h \cdot w \cdot (g' + h' \cdot w)$$

Thus

$$g_1 = 1 - 2 \cdot h \cdot h' \cdot w^2 = 1 - 2 \cdot h \cdot h' \cdot v$$
$$h_1 = -2 \cdot h \cdot g'$$

Finally

$$\frac{g_1 - 1}{v \cdot h_1} = \frac{-2 \cdot h \cdot h' \cdot v}{-2 \cdot h \cdot v \cdot g'} = \frac{h'}{g'}$$

Moreover

$$g' = \frac{g}{g^2 - v \cdot h^2}$$
$$h' = \frac{-h}{g^2 - v \cdot h^2}$$

So

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

□

$\frac{g_1-1}{v \cdot h_1} = \frac{h'}{g'} = -\frac{h}{g} \Rightarrow f_1 = f^{p^d-1}$

Proof. We write

$$\bar{f} \cdot f^{-1} = (g - h \cdot w) \cdot (g' + h' \cdot w) \tag{20}$$
$$= g \cdot g' - v \cdot h \cdot h' + (g \cdot h' + h \cdot g') \cdot w \tag{21}$$

with

$$g' = \frac{g}{g^2 - v \cdot h^2} = \frac{1}{g\,(1 - v \cdot K^2)} \text{ and } h' = \frac{-h}{g^2 - v \cdot h^2} = \frac{1}{h\,(v - 1/K^2)} \tag{22}$$

As a consequence:

$$g \cdot g' - v \cdot h \cdot h' = \frac{1 + v \cdot K^2}{1 - v \cdot K^2} = \frac{v \cdot h_1^2 + g_1^2 - 2 \cdot g_1}{v \cdot h_1^2 - g_1^2 + 2 \cdot g_1 - 1} = \frac{2 \cdot g_1 \cdot (g_1 - 1)}{2 \cdot (g_1 - 1)} = g_1$$

And

$$g \cdot h' + h \cdot g' = \frac{K}{1 - v \cdot K^2} - \frac{1}{K \cdot (v - 1/K)} = \frac{2 \cdot K}{1 - v \cdot K^2}$$
$$= \frac{2 \cdot (g_1 - 1) \cdot h_1}{v \cdot h_1^2 - g_1^2 + 2 \cdot g_1 - 1} = \frac{2 \cdot (g_1 - 1) \cdot h_1}{2 \cdot (g_1 - 1)} = h_1$$

□

B Size of the Intersection Set for the Candidates in f Recovery

Let e_{2t} and e_{3t} be in our fault model: $0 < e_{2t}, e_{3t} < 2^l - 1$ and $p >> 2^l$. Let $a = \frac{e_{2t}}{e_{3t}} \in \mathbb{F}_p$, we want to find the number of pairs (i, j) solutions to equation (17): $a \cdot i = j$ with $i, j \in [\![1, 2^l - 1]\!]$.

We can write

$$\frac{e_{2t}}{e_{3t}} = \frac{j}{i}$$

This fraction can be rewritten as $\frac{u}{v}$, reducing it to lowest terms:

$$u = \frac{e_{2t}}{gcd(e_{2t}, e_{3t})}$$
$$v = \frac{e_{3t}}{gcd(e_{2t}, e_{3t})}$$

All pairs solutions to equation (17) can be written as $(k \cdot u, k \cdot v)$, $k \in \mathbb{N}^+$. The conditions $i, j \in [\![1, 2^l - 1]\!]$ are equivalent to $k \leq \frac{2^l - 1}{u}$ and $k \leq \frac{2^l - 1}{v}$ which combined give $k \leq \frac{2^l - 1}{max(u,v)}$.

By the definition of u and v, we have: $max(u, v) = \frac{max(e_{2t}, e_{3t})}{gcd(e_{2t}, e_{3t})}$.
Finally, we have a solution for each integer k in the range

$$[\![1, (2^l - 1) \cdot \frac{gcd(e_{2t}, e_{3t})}{max(e_{2t}, e_{3t})}]\!]$$

The upper bound gives us the number of possible solutions to our equation (17).

C Attack Example

In this section, we provide the numerical values for an attack that was successfully simulated based on the methodology proposed in this paper. We used the same pairing parameters as in [9]. Our simulated fault injection creates error values in $[\![1, 2^4 - 1]\!]$ ($l = 4$).

Let the secret value be

$$\begin{aligned} f = (\ & \texttt{15E4F6523E7C5649E05B9FB24E3C212274A268F39E5034331ED5071CFBDF3A05}\cdot v^5 \\ & + \texttt{1672D105A344B97BFBB195D6AAAAB2E1912272E000432FD0866F789DB489165B}\cdot v^4 \\ & + \texttt{21D0D8EFDE1A9DDC227267B13D7EE703699B5E3293BCE339DF0CB70AC4D0D099}\cdot v^3 \\ & + \texttt{13A0D208C4134E0012166F8E7813A8D1FFB69CEBE0AD873426C181A95A5087C8}\cdot v^2 \\ & + \texttt{037A116F6C8A9CC97A775F672E751B3999D246DA5B056D417DE18891ED95EAE6}\cdot v \\ & + \texttt{05A9CC966050A3477C3510DAD85A6A31253203446D8907E228602D0E2AC27060})\cdot w \\ & + \texttt{06C9FA931438FD7122C35411049BE0D95CB2A1955AA51A653547560D8D01CD72}\cdot v^5 \\ & + \texttt{174072170F5121FE3658BE0CC4449CC7BBDA2298E5A3077885424861A9FD3DC2}\cdot v^4 \\ & + \texttt{13DED9A829FAD5568B466E7DFC42ECA52D8F6BCE25C635CE8A6E79155C56347F}\cdot v^3 \\ & + \texttt{20ECF9E9ED0A46FE32A4B5481C5D54A15C879B88B4A81C0AAE1254EEEAA4F226}\cdot v^2 \\ & + \texttt{147B7E0F2849E818D758194E503F0F691CC76207BF27065FDB18030E469F6533}\cdot v \\ & + \texttt{164C79AEC143A16DC5276597A89DFBF4D893B5D09D4A325301ACB45863A52AC0} \end{aligned}$$

C.1 Step 1: Normal Execution

First the attacker runs a normal execution, giving

$$\begin{aligned} f_3 = (\ & \texttt{14FF0ED863C56B2CF6790E35919CF0A8D33877A282EDC87C8574597257487813}\cdot v^5 \\ & + \texttt{1A59AE711E38EEA5D384214718CE68315AD9996B2CBFD7ACEDA5F1958E9C7CF8}\cdot v^4 \\ & + \texttt{04147EDBE3C5643AC6028BC597E9665D7B07C948DF7BB6CC3E367ACA223B29E4}\cdot v^3 \\ & + \texttt{1F23F4F893B297ED3EB321AF4AD3F17AA580B4D5D80CE54AA42E826738271689}\cdot v^2 \\ & + \texttt{100B00759CADCB5221D4B7CCC5C68B7980A53CD947452FB94D1B969F40624AC9}\cdot v \\ & + \texttt{0CBEE77D4398468DC63D8A13175B2E4FCCA9E4790A471B3F86D835C25E0D1FC0})\cdot w \\ & + \texttt{18C04751B8DFEA8F9CD7C813F15B5B37FB09738B04389D9CFBCBA4EABA9BB10E}\cdot v^5 \\ & + \texttt{21B35F3CA37C92BD73F88FA0249D736CF909208C12C32B5C22E42586E11B518E}\cdot v^4 \\ & + \texttt{09D3C014FE5AAA1F7F74AB0CC51793BFDA2551AC15B5040AF19586B22B6BA360}\cdot v^3 \\ & + \texttt{11A4EAB896B1C6D0F4D701D48C5C6F0D9D1148DE267A4A90A9258E0D112FDA23}\cdot v^2 \\ & + \texttt{13D2014FCE1AE043A88A108C969F9D658246962132901BAE75872DE5736ECF7D}\cdot v \\ & + \texttt{17C81BD9014A90D8964B3B864ABB83DF1225F513E49DD432D9459F22D4EBF7ED} \end{aligned}$$

C.2 Step 2: f_1 Recovery

A first fault injection is performed according to Subsection 4.1. The observed faulty value is

$$
\begin{aligned}
f_3^* = (\ & \texttt{14878AB9DA8D626472C222486B6BEAFCBB9D552E42C4A95F57CBE5DE0EB58A2B}\cdot v^5 \\
+\ & \texttt{1869B2D29B7B9DF5F28DC92904EE751125E223938C87C836102954D49D1BDDB1}\cdot v^4 \\
+\ & \texttt{073C8E8ECA143AC26ACC2B4738414098EEADA9DD198390C6FD49567873224085}\cdot v^3 \\
+\ & \texttt{1F3FE27B71407EF9DBD68E5AD408F94941A11DE9B27B20DF3894E7711E2C4572}\cdot v^2 \\
+\ & \texttt{018CA3F1F35B050D25191996940189F351942EE6DD0D10F0FE63B7DBC8C2417A}\cdot v \\
+\ & \texttt{0DE59F30BBE780A0D738E3B707C0A48F8C600E63857D31DDB78D0852476DB845})\cdot w \\
+\ & \texttt{18AD1088312DE86A6668FDA07CEEEE01137D06FF6F5402DD820B471FF42E2CDC}\cdot v^5 \\
+\ & \texttt{173D1A8CC7143964B7C6B3B17A5B14ABE25F22FBB74F779749FBE0DAE044D29B}\cdot v^4 \\
+\ & \texttt{023DACB18FBCAD8484A8FA8F35DDD57B124F48DF3B5676995821880FD6DD6485}\cdot v^3 \\
+\ & \texttt{08774A2A16C9CC6CA30D8BE07717B1234D075307097FC34F47DF6CB32CFF8B22}\cdot v^2 \\
+\ & \texttt{12AA1927DF8D8AD9DDD59A883D5918F685AAFB9ED2B196A16F0F3F8B8312F9AE}\cdot v \\
+\ & \texttt{0414DD150D1CA399A3AF8E5FD647423F9AD4A05624D74966835FE27ECAC42C9D}
\end{aligned}
$$

For each error guess $e_1 \in [\![1, 2^4 - 1]\!]$, g_1 is computed using equation (9) and h_1^+ and h_1^- with equation (5). Then $f_1^+ = g_1 + h_1^+ \cdot w$ and $f_1^- = g_1 + h_1^- \cdot w$ are constructed and checked against the observed $f_3 = f_1^{\frac{p^d+1}{r}}$ and $f_3^* = (f_1^*)^{\frac{p^d+1}{r}}$. A f_1 is found which satisfies these criteria for error $e_1 = 7$:

$$
\begin{aligned}
f_1 = (\ & \texttt{1E6BC8B6919346B74846AF6D4303D1A79D229A442435EA28865BD478D31AB1A2}\cdot v^5 \\
+\ & \texttt{221EA2429ED6254894C99D32D5BBCA06F5018B9C64F9A62051C4919EA815B097}\cdot v^4 \\
+\ & \texttt{200C76138D0DACBE0C6BC874CB0548D84A5C367C7665A7EFA14309F52B955502}\cdot v^3 \\
+\ & \texttt{1446D8D4F4D3892C42B72799B17AF78E4570319545EA24A19B96B8E937E14E0F}\cdot v^2 \\
+\ & \texttt{02FFF74B0C285EF8CC82010A422E0ADD0300E6C67C362E220ABA9CECEC20E051}\cdot v \\
+\ & \texttt{0F1CEA1EF6E3CA90D3FEBB5B2954A90A3F96F036138766370C1CD161D83F1768})\cdot w \\
+\ & \texttt{1DA86D419A0A0D17F20F0A96A2022160A35EAC0AC80B962A009908805CC5C8FE}\cdot v^5 \\
+\ & \texttt{02A30BA4FBE1821C659E5235C3375C55A5F715F521F6E32549A7314CE3C774AC}\cdot v^4 \\
+\ & \texttt{14297ECE1671FD16C3E57EB95F8DB69A53EEADBA16859E5EBC2184707BFBA1C2}\cdot v^3 \\
+\ & \texttt{17E7030B5FD4558F002D1F387B4180B9B989C813AF6B75FA5C4468297BF251A1}\cdot v^2 \\
+\ & \texttt{0270B45A029B9326291540F57B19A4093D197AA17BE66939EC67569EDE0168A3}\cdot v \\
+\ & \texttt{12065E0EFCFF4E4E25C594BCDC23F5D076FDC8003CB3F27618B523A6163D097A}
\end{aligned}
$$

C.3 Step 3: f Recovery

Two faults are injected according to Subsection 4.2. The observed faulty values are

$$
\begin{aligned}
f_3^* = (\ & \texttt{16F28C152154059E9DE6E9195258B8FC99E356EB1D9AEF299AC8FA826B33BBC6}\cdot v^5 \\
+\ & \texttt{08351B505C701E6E76CFFFB9877BE4B514A8138C1E0823860CF48777C359F5C5}\cdot v^4 \\
+\ & \texttt{20F5B35DD04E60CC85CB1AB1707C4045C19774512303F07BA4C259E545D2F9A1}\cdot v^3 \\
+\ & \texttt{22F43AF3353F93A45AF088D788D6EC32D0ADCF32CDB43B3C50378097B4665D46}\cdot v^2 \\
+\ & \texttt{2100931B1712BE28ECA6F35DF909828627C41AFB2352EA38E5D690526464B54A}\cdot v \\
+\ & \texttt{0AF16AD93F1FB968CC2C59FB00199823959 85A10E8DEFAA7C11C18DF841ABB9E})\cdot w \\
+\ & \texttt{223879E599390FD4DC285C9BA14BC1BAE64227C196B22CA2CF02DFA95AFC8E9B}\cdot v^5 \\
+\ & \texttt{0C78B0BB70A87D8BBCC72E84BA382FD4EC60AA11869D37BDAC82B639F9869B7B}\cdot v^4 \\
+\ & \texttt{1AE1AEA4A7B18D01340EB6017B5F7D0FD6134B07D764E819B64F529F07D6F980}\cdot v^3 \\
+\ & \texttt{1954E832F272C86EACA35DECC0A3F5CDA59E9D7A5F9C9EC7EF0FF51BC15DC125}\cdot v^2 \\
+\ & \texttt{151EF27A88585E1A229E81877B895642580D0623ED0BA264EF9DEA90E7FAECB1}\cdot v \\
+\ & \texttt{0A0BC9599DF18B044DE6522EE18E036DD76E875AC4E2C9007885C9F009F1E716}
\end{aligned}
$$

and

$$
\begin{aligned}
f_3'^{*} = (\ & \texttt{23571DDFC0C6B8509B84F49A969AB7F7BA38A5D071BFA339AF5078303D7F92BB}\cdot v^5 \\
+\ & \texttt{152F585FE7767B3A185C3BFE5BFC9A69C9AB0089BE6CAD2BA4A2382AC1E5CCE6}\cdot v^4 \\
+\ & \texttt{09C432E52552CF26B4484ED21B37B5C73E389299673EF9490ED5C63DCD1936DC}\cdot v^3 \\
+\ & \texttt{1DD38AA3691BD907A78DDFC4FDB1270E1D192E97DF6ECFD49BC63EC156BBBFB8}\cdot v^2 \\
+\ & \texttt{1B7D2A41682147DCA380B21CFBED319F3AEEF3C01F1E986E22E50E9167858663}\cdot v \\
+\ & \texttt{003B7A90812730447FDF12CE78075BE98399209D5AFB602FDD5A5E84DBA98979})\cdot w \\
+\ & \texttt{1377A70C46F2A429C0FD87941DEA17C3CCB29E84187D0952DCD9684651EC62B3}\cdot v^5 \\
+\ & \texttt{0F686B68CF92E4677259166B8D4C7F67E0DBAF18358826CFDF8462CF3E5BB747}\cdot v^4 \\
+\ & \texttt{015CCDF3776A4F4FDA9E02DF07C9F90E3D765C12DB3D25D49BC2CFF9401B105A}\cdot v^3 \\
+\ & \texttt{0FEB0A0E9229D1111C8BF20AE3A2638EB6FA4313020D2B341102CC6CC8F91560}\cdot v^2 \\
+\ & \texttt{0F72717DF131B16A8C69EC07A2EAE763DA688086C528EE7A9C09443B1BC0E4E0}\cdot v \\
+\ & \texttt{181A35AE9376E2DF2AA9BE6EA9807D24CEC537E834C9E80DDF5E810C84CD3AF6}
\end{aligned}
$$

By guessing that $e_2 = e_3 = 1$ and checking against observed f_3^* and $f_3'^*$, two candidates for f are saved using equation (15) and Lemma 2:

$$
\begin{aligned}
f_{c1} = (\ & \texttt{13D7C1CD2019B9E15AEA184A1DA41EEEA8AA745018D1D5C49CFB6004DAD90A28}\cdot v^5 \\
+\ & \texttt{18D4721BAB2536A450EFBF915D873EA6A92ACD9F8A5CAE4D41695D8B58D1C92E}\cdot v^4 \\
+\ & \texttt{14CFC849E4E70ECAC254D076F24E0D12CE6455C671A3FCFCE36F6F60EB575559}\cdot v^3 \\
+\ & \texttt{22E3763C6FA97F0768F181A01D1CE65C0962874A8619C0CF62CC0EF42CD4C604}\cdot v^2 \\
+\ & \texttt{1F4A9CEAD0DF83154E325AA0A21DF50668647DAD3330D74D6CEDCD215A454216}\cdot v \\
+\ & \texttt{0F873644C594ADC50F677D191448B978D1BFDE27C1D146F1ECE1357F80D5F35F})\cdot w \\
+\ & \texttt{19A8D42718BFD93BEA67DA00A295E562C5456F0017CDED8D6101A679F5103901}\cdot v^5 \\
+\ & \texttt{200F1BB87794E33860D0297843F077BE299FFB1F9ABB433536D2AB6EF9E72BCD}\cdot v^4 \\
+\ & \texttt{01A0BFD2678C31535D2B5D733BE1468728FF8DFEDBF7E43B656061C03F07D872}\cdot v^3 \\
+\ & \texttt{0CCCF1146B5400FD54198AC4C81FE7A058B27DCD99E8FC542AA1FC663FCC834E}\cdot v^2 \\
+\ & \texttt{072287715853DA2809CA5EC62FBE7F6A91F73605405F39573B563B807B9A8EC6}\cdot v \\
+\ & \texttt{177B600DB91B5E2466140D5A4B14D0542C2628150F9BC4E39690771B80CE80BB}
\end{aligned}
$$

and

$$
\begin{aligned}
f'_{c1} = (\ & \texttt{2163C67F7EDE7355C9049330564D000DB10A4A9C9281A192FCA6B8E7DEF9D024}\cdot v^5 \\
+\ & \texttt{0261A11607E07D28553E29BAB2DC8BC518085ABF8A197E7CBAF9E4EDA448B2D3}\cdot v^4 \\
+\ & \texttt{166FEA5EA40D80ACEE58835E3BB42850E1CB36D3F5E719C482E31856268684C1}\cdot v^3 \\
+\ & \texttt{0F42A433AB96310756DB1211A5093D8A09ABEA5EA56C399B3C0A8D4AD2843E3C}\cdot v^2 \\
+\ & \texttt{1BD08B7B6454E64BD3BAFB3973A8D9CCCE9236D2D82B6A0BEF0C448F6CAF5730}\cdot v \\
+\ & \texttt{09DD69AE65440A7D93326C3E3BEE4F47AC8DDAE354483F0FC4810871561382FF})\cdot w \\
+\ & \texttt{12DED9940486DBCAC7A485EF9DFA04896892CD6ABD28D3282BBA506C680E6B8F}\cdot v^5 \\
+\ & \texttt{08CEA9A16843C13A2A776B6B7FABDAF66DC5D886B5183BBCB190630D4FE9EE0B}\cdot v^4 \\
+\ & \texttt{1132E12EDAD26BBB205B098DC6835C2378726170CE31AE6E597248AAE2B1A3F4}\cdot v^3 \\
+\ & \texttt{0F50F22F1B8AC9BD6FEAF01532A79540792D2184FD40E04AFB10077755279129}\cdot v^2 \\
+\ & \texttt{16180466CD4B01CD80E8601066647242F232133D993832F8DEBE987234FB2994}\cdot v \\
+\ & \texttt{012EE65EF7D7BCB6A0ECA7C2A276D45F539272447251929094E3C2C31D2955FB}
\end{aligned}
$$

The ratio $a = \frac{f_{c1}}{f'_{c1}}$ is computed:

$$a = \texttt{1F02DBA40998EDC684A75745760861F94D61F758150000014EB0540000000002}$$

And one solution is found which satisfies equation (17): $8 \cdot a = 9$.

Finally we find that $f = f_{c1}/9 = f'_{c1}/8$ and it is the correct answer!