Fault and Side-Channel Attacks on Pairing Based Cryptography *

D. Page[1] and F. Vercauteren[2]

[1] Department of Computer Science,
University of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB,
United Kingdom.
page@cs.bris.ac.uk

[2] Department of Electrical Engineering,
Katholieke Universiteit Leuven,
Kasteelpark Arenberg 10,
B-3001 Leuven-Heverlee,
Belgium.
fvercaut@esat.kuleuven.ac.be

Abstract. Current side-channel analytic attacks against public key cryptography focus on traditional schemes such as RSA and ECC, and to a lesser extent primitives such as XTR. However, bilinear maps, or pairings, have presented theorists with a new and increasingly popular way of constructing cryptographic protocols. Most notably, this has resulted in efficient methods for Identity Based Encryption (IBE). Since identity based cryptography seems an ideal partner for identity aware devices such as smart-cards, in this paper we examine the security of concrete pairing instantiations in terms of side-channel analysis.

1 Introduction

The increasing ubiquity of computing devices is continuing to offer exciting new applications to consumers, but also multiplies the number of security issues a system designer must consider. Since such devices will be carried into and used in hostile environments and often house sensitive information, for example identity related tokens or financial information, the threat of attack is significant. This threat is magnified by both the potential pay-off and level of anonymity that side-channel attacks allow.

* The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT. The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability

Side-channel analysis, formally proposed by Kocher et al. [17, 18], is the art of externally monitoring a device while it executes some algorithm that includes secret information. By careful profiling of, for example, power or EM emissions and correlation of said profiles with the target algorithm, an attacker can often uncover the secret information. This sort of attack is generally described in the context of smart-cards where an attacker could have easy physical access to the card and details of the power consumption while it is connected to a malign terminal. However, the fact that one can attack a device somewhat remotely via timing [8] and EM emission [2, 3] means that most ubiquitous computing devices need to be aware of similar problems in their operational environments. We can extend this passive definition of side-channel attack to include active attacks such as fault injection whereby an attacker can physically manipulate the target device, tamper with internal state and perform erroneous operations. Although a closer proximity of access is required by the attacker, active attacks can often be applied non-destructively and do therefore provide a feasible means of recovering the same secret information as passive attacks.

Although side-channel attack and defence techniques are becoming increasingly well understood, the current emphasis in terms of public key systems is mainly on traditional cryptographic schemes such as RSA and ECC. For example, in ECC based systems, one generally derives security from a discrete logarithm problem posed over the curve group. That is, one constructs some private value d and performs the operation

$$Q = d \cdot P$$

for a public point P. For large enough curve groups, reversal of this operation is intractable and hence one can transmit Q without revealing d. A multitude of side-channel attacks against implementations of this primitive have been proposed. An equally large range of innovative countermeasures mean that when correctly implemented, the threats can generally be at least managed and possibly nullified.

Newer primitives such as XTR have received some investigation [22, 15], but there has been no previous work on the side-channel security of pairing based cryptography [11]. Since pairings, formally described as bilinear maps, underpin cryptographic protocols such as Identity Based Encryption (IBE) [9], one might see them as an ideal application for the same identity aware, ubiquitous computing devices that are vulnerable to side-channel attack. One reason for this lack of analysis is the fact that early implementation techniques for computing the Tate pairing such as the BKLS algorithm [7] are effectively realised as a point multiplication with a fixed multiplier and some auxiliary operations to compute the pairing value. From such a description, the fact that the algorithm performs a fixed sequence of operations and has no secret in a conventional sense has led people to believe that the pairing is secure against current side-channel attack methods. In part, this is true. However, the role of the pairing in associated protocols is somewhat different and much more diverse than point multiplication in ECC: it isn't enough to consider the new primitive in the context of traditional attack methods.

Field	Field Polynomial	Curve	Order	MOV security
$\mathbb{F}_{3^{79}}$	$t^{79}+t^{26}+2$	$Y^2=X^3-X-1$	$3^{79}+3^{40}+1$	750
$\mathbb{F}_{3^{97}}$	$t^{97}+t^{12}+2$	$Y^2=X^3-X+1$	$(3^{97}+3^{49}+1)/7$	906
$\mathbb{F}_{3^{163}}$	$t^{163}+t^{80}+2$	$Y^2=X^3-X-1$	$3^{163}+3^{82}+1$	1548
$\mathbb{F}_{3^{193}}$	$t^{193}+t^{12}+2$	$Y^2=X^3-X-1$	$3^{193}-3^{97}+1$	1830
$\mathbb{F}_{3^{239}}$	$t^{239}+t^{24}+2$	$Y^2=X^3-X-1$	$3^{239}-3^{120}+1$	2268
$\mathbb{F}_{3^{353}}$	$t^{353}+t^{142}+2$	$Y^2=X^3-X-1$	$3^{353}+3^{177}+1$	3354

Table 1. A table of field definitions and curve equations used in pairing based cryptography.

To bridge the resulting gap, in this paper we present an investigation of pairings and associated protocols in terms of their security against fault attacks and side-channel analysis. Although one can parameterise pairing based protocols according to a large number of options we try to consider general cases, offering concrete examples when discussing specific issues. Furthermore, we consider both software and hardware implementation given that different methods may be used to realise each.

We organise our work by first presenting a brief introduction to pairings and algorithms for their evaluation in Section 2. We then investigate applications of active, fault injection attacks against the Duursma-Lee algorithm and the Baek-Zheng (t, n)-threshold decryption scheme in Section 3. In Section 4 we then present a method of passive side-channel attack and associated defence techniques, focusing on Boneh-Franklin [9] encryption. Finally, we present some concluding remarks in Section 5.

2 An Introduction to Pairings

To ease description in the context of side-channel analysis, we use the concrete example of pairings where the base field is of characteristic three, i.e. $\mathbb{F}_q$ where $q = 3^m$. Although most of our results translate easily to situations where other fields are used, selecting characteristic three allows us to present and consider a unified description of all known methods of pairing evaluation.

Let E be an elliptic curve over a finite field $\mathbb{F}_q$, and let $\mathcal{O}$ denote the identity element of the associated group of rational points $E(\mathbb{F}_q)$. We include a table of suitable curve parameterisations in Table 1. For a positive integer $l|\#E(\mathbb{F}_q)$ coprime to q, let $\mathbb{F}_{q^k}$ be the smallest extension field of $\mathbb{F}_q$ which contains the l-th roots of unity in $\overline{\mathbb{F}}_q$. Also, let $E(\mathbb{F}_q)[l]$ denote the subgroup of $E(\mathbb{F}_q)$ of all points of order dividing l, and similarly for the degree k extension of $\mathbb{F}_q$. To unify the notation used in most protocols, we use three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_3$. The groups $\mathbb{G}_1$ and $\mathbb{G}_2$ will always be subgroups of elliptic curve groups, whereas the group $\mathbb{G}_3$ is a subgroup of the multiplicative group of a finite field. In all the schemes we consider here, we set $\mathbb{G}_1 = \mathbb{G}_2$ and assume that the pairing internally distorts the inputs provided into the required groups.

The Reduced Tate Pairing From an efficiency perspective, k is usually chosen to be even [7]. For a thorough treatment of the following, we refer the reader to [7] and also [12], and to [25] for an introduction to divisors. The reduced Tate pairing of order l is the map

$$e_l : E(\mathbb{F}_q)[l] \times E(\mathbb{F}_{q^k})[l] \to \mathbb{F}_{q^k}^* / (\mathbb{F}_{q^k}^*)^l,$$

given by $e_l(P, Q) = f_{P,l}(\mathcal{D})$. Here $f_{P,l}$ is a function on E whose divisor is equivalent to $l(P) - l(\mathcal{O})$, $\mathcal{D}$ is a divisor equivalent to $(Q) - (\mathcal{O})$, whose support is disjoint from the support of $f_{P,l}$, and $f_{P,l}(\mathcal{D}) = \prod_i f_{P,l}(P_i)^{a_i}$, where $\mathcal{D} = \sum_i a_i P_i$. It satisfies the following properties:

- For each $P \neq \mathcal{O}$ there exists $Q \in E(\mathbb{F}_{q^k})[l]$ such that $e_l(P, Q) \neq 1 \in \mathbb{F}_{q^k}^* / (\mathbb{F}_{q^k}^*)^l$ *(non-degeneracy)*.
- For any integer n, $e_l([n]P, Q) = e_l(P, [n]Q) = e_l(P, Q)^n$ for all $P \in E(\mathbb{F}_q)[l]$ and $Q \in E(\mathbb{F}_{q^k})[l]$ *(bilinearity)*.
- Let $L = hl$. Then $e_l(P, Q)^{(q^k-1)/l} = e_L(P, Q)^{(q^k-1)/L}$.
- It is efficiently computable.

The non-degeneracy condition requires that Q is not a multiple of P, i.e. that Q is in some order l subgroup of $E(\mathbb{F}_{q^k})$ disjoint from $E(\mathbb{F}_q)[l]$. When one computes $f_{P,l}(\mathcal{D})$, the value obtained belongs to the quotient group $\mathbb{F}_{q^k}^* / (\mathbb{F}_{q^k}^*)^l$, and not $\mathbb{F}_{q^k}^*$. In this quotient, for a and b in $\mathbb{F}_{q^k}^*$, $a \sim b$ if and only if there exists $c \in \mathbb{F}_{q^k}^*$ such that $a = bc^l$. Clearly, this is equivalent to

$$a \sim b \;\text{ if and only if }\; a^{(q^k-1)/l} = b^{(q^k-1)/l},$$

and hence one ordinarily uses this value as the canonical representative of each coset. The isomorphism between $\mathbb{F}_{q^k}^* / (\mathbb{F}_{q^k}^*)^l$ and the elements of order l in $\mathbb{F}_{q^k}^*$ given by this exponentiation makes it possible to compute $f_{P,l}(Q)$ rather than $f_{P,l}(\mathcal{D})$. In fact we can consider the Tate pairing as a pairing of the groups $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$, with $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{E}(\mathbb{F}_{q^k})[l]$ and $\mathbb{G}_3 = \mu_l \subset \mathbb{F}_{q^k}$, the group of l-th roots of unity.

The BKLS [7] algorithm takes advantage of this fact to evaluate the pairing using Miller's algorithm but without the costly denominators. Using this algorithm, we operate on tuples we call T-points such that $(P, \alpha) \in \mathbb{G}_1 \times \mathbb{G}_3$. We then define addition of these objects

$$(P_3, \gamma) = (P_1, \alpha) + (P_2, \beta)$$

using Algorithm 1. Note that this is essentially normal point addition with some auxiliary operations to compute γ and that a method for tripling T-points follows directly from the above. We use tripling rather than doubling because the efficiency of cubing in $\mathbb{F}_q$ allows a particularly concise point tripling formula; in other characteristics doubling would obviously be more appropriate. Also note as mentioned above, we can ignore the division by $V(x_{P_3}, y_{P_3})$ which vastly accelerates evaluation.

Algorithm 1: An algorithm for adding T-points within BKLS evaluation.

Input : T-point $\mathcal{P}_1 = (P_1, \alpha)$, point $\mathcal{P}_2 = (P_2, \beta)$

Output : T-point $\mathcal{P}_3 = (P_3, \gamma) = \mathcal{P}_1 + \mathcal{P}_2$

$P_3 \leftarrow P_1 + P_2$
let $L(X, Y) = 0$ denote the line through P_1 and P_2
if $P_3 \neq \mathcal{O}$ **then**
    let $V(X, Y) = 0$ denote the line through P_3 and $\mathcal{O}$
else
    let $V(X, Y) = 1$ denote the line through P_3 and $\mathcal{O}$
$\gamma \leftarrow \frac{\alpha \cdot \beta \cdot L(x_{P_3}, y_{P_3})}{V(x_{P_3}, y_{P_3})}$
return (P_3, γ)

Algorithm 2: The Duursma-Lee algorithm.

Input : point $P = (x_1, y_1)$, point $Q = (x_2, y_2)$

Output : $f_P(\phi(Q)) \in \mathbb{F}^*_{q^6}/\mathbb{F}^*_{q^3}$

$f \leftarrow 1$
for $i = 1$ **to** m **do**
    $x_1 \leftarrow x_1^3$
    $y_1 \leftarrow y_1^3$
    $\mu \leftarrow x_1 + x_2 + b$
    $\lambda \leftarrow -y_1 y_2 \sigma - \mu^2$
    $g \leftarrow \lambda - \mu\rho - \rho^2$
    $f \leftarrow f \cdot g$
    $x_2 \leftarrow x_2^{1/3}$
    $y_2 \leftarrow y_2^{1/3}$
return f

Once the requisite tripling and addition methods as constructed for T-points, to compute the pairing we simply perform the equivalent of a point multiplication of P using these methods in place of conventional ECC point arithmetic and the curve order as the multiplier. At the end of this multiplication, we will have calculated $(\mathcal{O}, \gamma)$ such that when we power γ by $(q^k - 1)/l$ we have the result we want.

The Modified Tate Pairing Duursma and Lee introduced their algorithm in the context of pairings on a family of hyperelliptic curves. Restricting to the elliptic curve case, it applies to a family of supersingular curves in characteristic three, including those in Table 1.

Let $q = 3^m$ and $E(\mathbb{F}_q) : Y^2 = X^3 - X + b$, with $b = \pm 1$, and let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be points of order l. Let $\mathbb{F}_{q^3} = \mathbb{F}_q[\rho]/(\rho^3 - \rho - b)$, with $b = \pm 1$ depending on the curve equation, and let $\mathbb{F}_{q^6} = \mathbb{F}_{q^3}[\sigma]/(\sigma^2 + 1)$. Then the modified Tate pairing on E is the mapping $f_P(\phi(Q))$ where $\phi : E(\mathbb{F}_q) \to E(\mathbb{F}_{q^6})$ is the distortion map $\phi(x_2, y_2) = (\rho - x_2, \sigma y_2)$. The method for computing this is shown in Figure 2, noting that the final result is powered by $q^3 - 1$ to form a compatible result with the BLKS method.

3 Active Attacks

3.1 A Faulty Duursma-Lee Algorithm

Recovering a Secret Point Given the secret point $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, a point chosen by the attacker, assume for the moment that we can eliminate the final powering. Let $\overline{e}_\Delta$ denote pairing via the Duursma-Lee algorithm where through tampering we induce some transient fault and replace the loop bound m with Δ. This could be achieved by using glitch attack [4, 19] to tamper with the loop test and branch mechanism, or given that the Duursma-Lee algorithm is essentially parameterised by m given the right field arithmetic and that m is therefore likely to be held in memory somewhere, by selectively provoking errors in memory or registers [26, 5]. In this latter case, it may even by feasible to mount buffer-overflow type attacks to change the value of m. Given this ability, instead of producing a product of polynomials of the form

$$\prod_{i=1}^{m} \left[(-y_1^{3^i} \cdot y_2^{1/3^{i-1}} \sigma - (x_1^{3^i} + x_2^{1/3^{i-1}} + b)^2) - (x_1^{3^i} + x_2^{1/3^{i-1}} + b)\rho - \rho^2\right]$$

the algorithm instead produces

$$\prod_{i=1}^{\Delta} \left[(-y_1^{3^i} \cdot y_2^{1/3^{i-1}} \sigma - (x_1^{3^i} + x_2^{1/3^{i-1}} + b)^2) - (x_1^{3^i} + x_2^{1/3^{i-1}} + b)\rho - \rho^2\right]$$

for some value Δ and remembering that for now, we ignore the final powering.

If we were able to induce a fault so that $\Delta = m + 1$, for example by glitching the loop iteration so it executes once too often, then recovering the secret point it trivial: we compute two pairings, one correct and one tampered with

$$\begin{aligned} R_1 &= \overline{e}_m(P, Q) \\ R_2 &= \overline{e}_{m+1}(P, Q)\,. \end{aligned}$$

If we use $g_{(i)}$ to denote the i-th factor of a product produced by the Duursma-Lee algorithm, by dividing the two results above we are left with a single factor

$$g_{(m+1)} = (-y_1^{3^{m+1}} \cdot y_2^{1/3^m} \sigma - (x_1^{3^{m+1}} + x_2^{1/3^m} + b)^2) - (x_1^{3^{m+1}} + x_2^{1/3^m} + b)\rho - \rho^2.$$

Since we have that

$$z^{3^m} = z^{1/3^m} = z$$

for all elements $z \in \mathbb{F}_q$, we can extract x_1 or y_1, given that we know x_2 and y_2, and hence reconstruct the secret point.

However, the ability to selectively induce a fault so $\Delta = m+1$ seems tenuous. It is far more realistic to assume that we can induce $\Delta = m \pm r$ for random, unknown values of r. Using this ability, we calculate many erroneous pairing value with the aim of collecting a pair

$$\begin{aligned} R_1 &= \overline{e}_{m\pm r}(P,Q) \\ R_2 &= \overline{e}_{m\pm r+1}(P,Q)\,, \end{aligned}$$

so that we are again left with a single term of the product $g_{(m\pm r+1)}$ and can hence run a similar method as above although needing to compensate for the differing powers of x_1, y_1, x_2 and y_2 introduced by r.

The problem with this approach is detecting when we have induced faults with the correct Δ values for use. This is also quite an easy task. Since the algorithm is straight-line and takes a fixed number of operations, given the time taken to compute the pairing or a power profile, it seems simple to work out how many loop iterations were performed in the same way that one can observe and count the sixteen rounds of DES. Given the value of r, this approach will deterministically recover the correct value of P and requires reasonably few invocations of the pairing on the target device due to a similar argument as the birthday paradox: we simply keep provoking random faults until we recover a pair of results that satisfy our conditions.

Reversing the Final Powering The remaining problem then, is to reverse the final powering which takes the output of the pairing function and produces a unique representative for the coset in $\mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^l$. To reverse this operation given the result

$$R = e(P,Q)$$

we want to recover S, the value that was computed by the algorithm before the final powering so that $R = S^{(q^k-1)/l}$. For the Duursma-Lee method shown in Algorithm 2, this simplifies to $R = S^{q^3-1}$. It is clear that given R, the value of S is not uniquely determined, but only up to a non-zero factor in $\mathbb{F}_{q^3}$. Indeed, for non-zero $c \in \mathbb{F}_{q^3}$ we have $c^{q^3-1} = 1$. Furthermore, given one solution $S \in \mathbb{F}_{q^6}$ to $X^{q^3-1} - R = 0$, all others are of the form $c \cdot S$ for non-zero $c \in \mathbb{F}_{q^3}$. As an aside, note that this is similar to the T_2 compression and decompression mechanism of Rubin and Silverberg [23].

However, given the attack description above we are not trying to reverse the powering on the full product of factors computed by the Duursma-Lee algorithm, rather only one of these factors which has a fairly special form. That is, given

$$R = \frac{R_2}{R_1} = \frac{\overline{e}_{m\pm r+1}(P,Q)}{\overline{e}_{m\pm r}(P,Q)} = g^{q^3-1}_{(m\pm r+1)}$$

we want to recover $g_{(m\pm r+1)}$ which will, in turn, allow us to recover the correct x_1 and y_1 for the secret point.

We therefore need a method to compute one valid root of $R = g^{q^3-1}$ for some factor g, and then derive the correct value of g from amongst all possible solutions. The first problem can be solved very efficiently as follows: multiply the equation $X^{q^3-1} - R = 0$ by X to obtain

$$X^{q^3} - R \cdot X = 0\,,$$

and note that the operator $X^{q^3} - R \cdot X$ is a linear operator on the two dimensional vector space $\mathbb{F}_{q^6}/\mathbb{F}_{q^3}$. Since $\mathbb{F}_{q^6} \cong \mathbb{F}_{q^3}[\sigma]/(\sigma^2-1)$ we can write $X = x_0 + \sigma x_1$ and $R = r_0 + \sigma r_1$, with $x_0, x_1, r_0, r_1 \in \mathbb{F}_{q^3}$. Using this representation, we see that the above equation is equivalent to

$$M \cdot X = \begin{pmatrix} 1 - r_0 & r_1 \\ r_1 & 1 + r_0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = 0\,.$$

The kernel of the matrix M is a one-dimensional vector space over $\mathbb{F}_{q^3}$ and thus provides all the solutions to $X^{q^3-1} - R = 0$.

To choose the correct root amongst all $q^3 - 1$ possibilities, we use the specific form of the factors in the product computed in the Duursma-Lee algorithm. Indeed, each factor g is of the form

$$g = g_0 + g_1\rho - \rho^2 + g_2\sigma\,,$$

with $g_0, g_1, g_2 \in \mathbb{F}_q$. To recover g from $R = g^{q^3-1}$, we first obtain $g' = c \cdot g$ for some $c \in \mathbb{F}_{q^3}$ using the root finding algorithm above, and then compute c^{-1} and thus g itself. Again this boils down to a simple linear system of equations: by multiplying g' with an appropriate factor in $\mathbb{F}_{q^3}$, we can assume that g' is of the form $g' = 1 + (g'_0 + g'_1\rho + g'_2\rho^2)\sigma$. Let $d = c^{-1} = g/g' \in \mathbb{F}_{q^3}$, then d clearly is of the form $d = d_0 + d_1\rho - \rho^2$. To determine $d_0, d_1 \in \mathbb{F}_q$, we use the fact that the terms $\rho\sigma$ and $\rho^2\sigma$ do not appear in g. This finally gives the following linear system of equations:

$$\begin{pmatrix} g'_1 & g'_0 + g'_2 \\ g'_2 & g'_1 \end{pmatrix} \begin{pmatrix} d_0 \\ d_1 \end{pmatrix} = \begin{pmatrix} g'_1 + g'_2 \\ g'_0 + g'_2 \end{pmatrix}\,.$$

An Example To make this a little clearer, we present an example attack using the described method. Assume that we want to discover the secret point $P = (x_1, y_1)$, are granted control of $Q = (x_2, y_2)$, and receive results from the faulty pairing $R_i = \overline{e}_\Delta(P, Q)$. We perform many executions of the pairing, invoking different errors until we recover two values

$$\begin{aligned} R_j &= \overline{e}_{m \pm r}(P, Q) \\ R_k &= \overline{e}_{m \pm r+1}(P, Q). \end{aligned}$$

For the sake of concreteness but without loss of generality, let us assume that $m = 97$, i.e. $q = 3^{97}$, and $r = -7$ which gives

$$\begin{aligned} R_j &= \overline{e}_{90}(P, Q) \\ R_k &= \overline{e}_{91}(P, Q). \end{aligned}$$

Since we observed the calculation of R_j and R_k, either by timing the operation of monitoring some other side-channel, we can tell how many loop iterations were performed in the Duursma-Lee algorithm and hence the value of r. Given this information, we next compute

$$R = \frac{R_k}{R_j} = g_{(91)}^{q^3+1}$$

which is the 91-st term of the product produced by the Duursma-Lee algorithm including the final powering. This final powering is then removed and the correct g value recovered as described above so that we are left only with

$$g_{(91)} = (-y_1^{3^{91}} \cdot y_2^{1/3^{90}} \sigma - (x_1^{3^{91}} + x_2^{1/3^{90}} + b)^2) - (x_1^{3^{91}} + x_2^{1/3^{90}} + b)\rho - \rho^2.$$

From this polynomial we can clearly extract the coefficients

$$\begin{aligned} x_3 &= x_1^{3^{91}} + x_2^{1/3^{90}} \\ y_3 &= y_1^{3^{91}} \cdot y_2^{1/3^{90}} \end{aligned}$$

given that we know b. We also know x_2 and y_2 so by computing the values $x_2^{1/3^{90}}$ and $y_2^{1/3^{90}}$, possible since we also know r, we can eliminate these terms from x_3 and y_3 to leave

$$\begin{aligned} x_3 &= x_1^{3^{91}} \\ y_3 &= y_1^{3^{91}}. \end{aligned}$$

Finally, since for all $z \in \mathbb{F}_{3^{97}}$ we have $z = z^{3^{97}}$, if we power x_3 and y_3 by 3^6, we are produce

$$\begin{aligned} x_3 &= x_1 \\ y_3 &= y_1 \end{aligned}$$

which recovers the secret point. In the above example alterations to the process clearly need to be made for different situations but the attack clearly works against any randomly provoked value of r.

3.2 Baek-Zheng (t, n)-Threshold Decryption

Attack Description Baek and Zheng [6] proposed a method for (t, n)-threshold decryption that is based on extensive use of pairings; we provide an overview of this scheme in the Appendix. The basic idea is that a central entity, termed the authorised dealer, is granted a private key associated with his identity by a trusted authority. He then runs an algorithm to share this private key between n decryption servers with the servers storing their key share and the corresponding verification key: the private key shares are kept secret, the verification key shares are published.

Anyone can encrypt a message to the authorised dealer using his identity as the public key. When an authorised delegate of the trusted dealer wants to decrypt such a message, they give the ciphertext to each decryption server and get back a decryption share. By recombining at least t of these shares, the original message is recovered.

In short, an attacker can feed as many chosen ciphertexts as they want to each decryption server and get back decryption shares. This is the part of scheme that is vulnerable to attack. In a simplified form, a given decryption server computes the values

$$\begin{aligned} a &= e(S, U) \\ b &= e(R, U) \\ c &= e(R, P) \end{aligned}$$

where S is the key share the attacker would like to recover, R is a random unknown element in $\mathbb{G}_1$ and P is a public generator of $\mathbb{G}_2$. The value of U is parsed from the ciphertext $C = (U, V, W)$ and is hence under control of the attacker. Having computed these values, the server returns the tuple (a, b, c) as well as some other values to the attacker.

If we assume the attacker can provoke faults in the first of the pairings, then we are in the case where we control one input to the pairing, the other input is secret and we get the result back. Hence, we can run the attack described previously and recover S, the key share for each server. Even if the original secret key cannot be recovered, the attacker can decrypt any message since he knows everything that each decryption server does.

Defence via Improved Design This is hardly a devastating attack since it is somewhat unrealistic to assume that the attacker could provoke faults in the pairing given that the server is only remotely accessible. However, it does hint that the transmission or raw pairing values to an attacker who has provided the input could generally be seen as bad design practise. For example, Boneh-Franklin encryption is saved from the same attack because raw pairing values are never transmitted: they first pass through a hash function which masks their actual form. This seems the only strong defence against the fault attack that does not depend on taking the obvious route of preventing the fault using hardware assistance.

Using Point Blinding Techniques If raw pairing values really need to be transmitted, one can utilise the point blinding techniques described in Section 4 to construct a defence mechanism. Such techniques apply a blinding factor to one or both of the input points before the pairing and eliminate this factor afterwards. If we blind the point under control of the attacker, any tampering with the pairing will cause the un-blinding phase to essentially produce a random result.

4 Passive Attacks

4.1 Boneh-Franklin Encryption

Attack Description Considering Boneh-Franklin encryption [9], as outlined briefly in the Appendix, we see that the pairing is applied once during encryption and once during decryption. During encryption, the pairing operates only on public values and furthermore, the result may be pre-computed by the sender for each recipient of encrypted messages. This operation is therefore unsuitable for manipulation by the attacker.

However, the recipient of a ciphertext $C = (U, V)$ from the attacker must use their private key S_{ID} to decrypt the corresponding message. This information is fed into the pairing to compute

$$M = V \oplus H_2(e(S_{ID}, U)).$$

In this case an attacker can control one of the points supplied to the pairing, i.e. the value of U, while the other is both fixed and private. Furthermore, since the attacker controls the value of C he can prompt repeated executions of the pairing, adapting his input as required. For example, a smart-card implementing the Boneh-Franklin scheme might be repeatedly asked by the attacker to decrypt different messages that he has constructed so that monitoring the decryption yields information: assuming $S_{ID} = (x_1, y_1)$ is the secret point and $U = (x_2, y_2)$ is controlled by the attacker, manipulating y_2 to recover y_1 would allow the attacker to reconstruct S_{ID}.

To recover the private key S_{ID}, the attacker must monitor an operation that combines it in some way with the controlled point U. Thus, through understanding of said operation and manipulation of the controlled point, he can reveal the value of the secret point. Such an operation is clearly accessible in Step 6 of Algorithm 2, the Duursma-Lee algorithm, where one computes the product of y_1 and y_2. Similarly, in the BKLS algorithm, one needs to compute a product when updating the pairing value during a tripling operation

$$\gamma = \gamma^3 \cdot (a \cdot x_{P_1} + b + y_{P_1})$$

where a and b are derived from the coordinates of one input point and x_{P_1} and y_{P_1} from the other. The product $a \cdot x_{P_1}$ thus has one operand derived from S_{ID} and one from U. Careful analysis of how these values are derived followed by control of U allows a route of attack to be established.

SPA-like Attack In the most naive situation where the field multiplication is implemented using the shift-and-add method, one can mount an attack by monitoring execution in the same way as one would monitor binary exponentiation. Simply put, if a shift-and-add multiplication algorithm is employed and y_1 is used as the multiplier, the multiplication can be attacked using SPA analysis to spot where the conditional add operations take place.

Algorithm 3: A Messerges style DPA attack to reveal $S_{ID} = (x_1, y_1)$ by guessing y_1 one bit at a time.

set y_g to 0
set S_{hi} and S_{lo} to empty
for $i = 0$ **to** $n - 1$ **do**
  guess the i-th bit of y_g to one
  for $k = 0$ **to** $r - 1$ **do**
    select at random $U = (x_u, y_u)$ and $V = (x_v, y_v)$
    calculate $X = y_g \cdot y_u$
    use smart-card to decrypt $C = (U, V)$, collect power signal $S_k[j]$
    if the i-th bit of X is 1 **then**
      add $S_k[j]$ to S_{hi}
    else
      add $S_k[j]$ to S_{lo}
  average power signals to get DPA bias $D[j] = \overline{S}_{hi}[j] - \overline{S}_{lo}[j]$
  if DPA bias signal has a spike **then**
    the guess was right: set i-th bit of y_g to 1
  else
    the guess was wrong: set i-th bit of y_g to 0
reconstruct x_g from y_g and **return** $S_{ID} = (x_g, y_g)$

Although clearly viable given some assumptions about the implementation, there are a number of drawbacks to this approach. For example, in characteristic three the shift-and-add method performs an addition when the current multiplier coefficient is equal to one and a subtraction when it is equal to two. Since the attacker is unlikely to be able to distinguish between an addition and a subtraction, SPA analysis of the operation trace will only yield the fact that a given coefficient is either zero or non-zero rather than the actual magnitude.

DPA-like Attack A method for revealing one operand of an integer multiplication given knowledge of the other is presented by Messerges [20][Page 93] in the context of an attack against ECDSA. It is easy to see how an analogue would apply in the case of finite fields and where instead of one operand being produced by the algorithm, as in ECDSA where the r in $r \cdot d$ is an output, it is under control of the attacker as an input. Algorithm 3 outlines such an attack, guessing the value of y_1 one bit at a time and using a correlation with the power output of the smart-card as an indicator to tell if the guess is correct. In this description we use n to denote the length of y_1 and r to represent a limit on the number of runs of decryption on the smart-card. Note that unlike the SPA method where characteristic three seems troublesome, this attack is easily extended to cope by considering a field element as simply represented by a set of bits and guessing each one in turn.

Using Point Multiplication and Bilinearity A general method of defending against both the SPA and DPA attacks is to use the concept of point blinding to randomise the points fed into the pairing. By randomising the points on each invocation of the pairing, the attacker can no longer perform any correlation between successive runs in a DPA-like attack and any successful analysis of a single run using an SPA-like attack will simply yield the randomised rather than secret data. Since we know that the relationship

$$e(a \cdot P, b \cdot Q) = e(P, Q)^{a \cdot b}$$

holds, we can randomise the points P and Q by selecting random values for a and b. Although this results in an additional factor of $a \cdot b$ in the exponent of the result, we can eliminate this by careful selection of a and b such that

$$a \cdot b = 1 \pmod{l}.$$

If one cannot accept the cost of a GCD-like operation to compute arbitrary random a and b pairs of the right form, it is possible to employ a deterministic update procedure akin to traditional point blinding. One computes and stores two random pairs such that

$$\begin{aligned} a \cdot b &= 1 \pmod{l} \\ c \cdot d &= 1 \pmod{l}. \end{aligned}$$

This is only ever done once, perhaps when the device is initialised. Updating a and b can then be achieved by multiplication with c and d

$$\begin{aligned} a &\leftarrow a \cdot c \pmod{l} \\ b &\leftarrow b \cdot d \pmod{l} \end{aligned}$$

which only costs two field multiplications per update and retains the fact that $a \cdot b = 1 \bmod l$. In summary, the defence costs two additional point multiplications and two field multiplications to instrument.

Altering Traditional Point Blinding Traditional point blinding for ECC has the host device store two extra points that are retained across invocations of the point multiplication algorithm. Given a secret multiplier d, the card stores a random point R and the point $S = d \cdot R$. The point multiplication $d \cdot P$ is then computed as

$$d \cdot P = d \cdot (P + R) - S$$

so that the point fed into the multiplication routine is randomly blinded by first adding R with the result recovered by subtracting S. Considering the relationship

$$e(P, Q + R) = e(P, Q) \cdot e(P, R)$$

we can apply an augmented point blinding technique to the pairing so that the controlled point is again randomised before use. For the DPA-like attack described above, this seems sufficient defence since by randomising the control point, the attack can no longer correctly compute the multiplication oracle that drives DPA selection.

Assuming P is the secret point, Q is the point controlled by the attacker that we want to randomise, R is a random point and $S = e(P,R)^{-1}$, we apply the map as

$$e(P,Q) = e(P,Q+R) \cdot S.$$

By blinding Q, the point under control of the attacker, we effectively remove this control and hence defeat any attack based on it: the attacker can no longer reason about internal operation based on the point they sent into the pairing.

Since P and R are known to the device at initialisation, we can store the point R and field element S in order to implement the method. Using this technique, the overhead in performance is equal to one point addition and a field multiplication in $\mathbb{G}_3$. However, the issue of updating the blinding variables presents a drawback. Updating R and S is performed before or after the multiplication routine is executed in order to provide changing and hopefully non-deterministic blinding factors to the real point. In the traditional blinding defence, one might perform the following update operations

$$\begin{aligned} b &\leftarrow \{-2,+2\} \\ R &\leftarrow b \cdot R \\ S &\leftarrow b \cdot S. \end{aligned}$$

That is, one doubles the points and may randomly also invert them, although this is inexpensive. However, this simple form of update has been shown to be vulnerable to attack [21]. Furthermore, in our case we need to perform the operations

$$\begin{aligned} b &\leftarrow \{-2,+2\} \\ R &\leftarrow b \cdot R \\ S &\leftarrow S^b. \end{aligned}$$

Squaring a field element is reasonably inexpensive but in order to accommodate both potential values of b, we need to store some extra information so that inversion is equally inexpensive. To do this, we can use a technique in characteristic three where a fractional representation of $\mathbb{G}_3$ renders inversion inexpensive [13] or simple store and update the value S^{-1} along with S if there is enough space.

5 Conclusion

We have presented the first investigation into the security of pairing based cryptography against side-channel attack. Although the use of pairings in the sorts of environment where side-channel attack is most prevalent is still some way off, the coupling of identity based cryptography with identity aware devices seems very attractive. Naive early assumptions about the security of pairings in this setting were as a result of directly applying existing knowledge of RSA and ECC vulnerabilities. However, since the use of pairings represents a vastly different and more diverse problem from that in more conventional systems, it is important to also consider new methods of attack. Our results show that although vulnerabilities do inevitably exist, creative use of the bilinearity property of pairings and sensible implementation methods help to minimise such risk with low overhead.

There are clearly many other areas that could be followed up in light of this work. The diverse way in which pairings are used in comparison to conventional exponentiation in RSA or ECC means that vulnerabilities are equally as diverse in their manifestation:

Further passive attacks The signature scheme of Hess presents an additional avenue of attack related to conventional point multiplication: finding an unknown point given control of the multiplier. That is, can one discover P by monitoring the execution of the operation $d \cdot P$ where d is known and controlled by the attacker. This is clearly different from conventional attacks on point multiplication in the context of ECC since there, d is the secret and P is typically known and only potentially controllable. As well as the scheme of Hess, this problem also relates to several of our countermeasures so at least seems worth investigating further.

Further active attacks Ciet and Joye [10] described two different fault attacks on ECC which work by computing point multiplications with invalid points and erroneous field arithmetic. Translating these concepts to the context of pairings seems interesting but difficult. For example, feeding invalid points into the pairing generally causes it to become degenerate or produce garbage results. Additionally, we need to consider two points, one of which is unknown and fixed: it is not clear what would happen if the one under control of the attacker is not on the expected curve.

Special point attacks Goubin [14] proposed an attack on ECC point multiplication whereby special points were fed into the implementation. These points trigger errors in execution that, when carried through to the result, reveal information about the internal operation and hence private information. This method was later enhanced by Akishita and Takagi [1]. Although such attacks are easily countered [27] it seems interesting to consider what happens when a similar approach is used for points fed into the pairing. Clearly the attacks do not apply directly: there is no secret scalar multiplier to reveal by using so called special points. However, it is not clear what happens if either input to the pairing is a special point in some more pairing-specific sense.

Acknowledgements

The authors would like to thank Steven Galbraith, Rob Granger, Nigel Smart and Martijn Stam for invaluable help and discussion throughout the course of this work.

References

1. T.Akishita and T. Takagi. Zero-Value Point Attacks on Elliptic Curve Cryptosystem. In *Information Security Conference (ISC)*, Springer-Verlag LNCS 2851, 218–233, 2003.
2. D. Agrawal, B. Archambeault, J.R. Rao and P. Rohatgi. The EM Side-Channel(s). In *Cryptographic Hardware and Embedded Systems (CHES)*, Springer-Verlag LNCS 2523, 29–45, 2002.
3. D. Agrawal, J.R. Rao and P. Rohatgi. Multi-channel Attacks. In *Cryptographic Hardware and Embedded Systems (CHES)*, Springer-Verlag LNCS 2779, 2–16, 2003.
4. R.J. Anderson and M.G. Kuhn. Low Cost Attacks on Tamper Resistant Devices. In *International Security Protocols Workshop (IWSP)*, Springer-Verlag LNCS 1361, 125–136, 1997.
5. H. Bar-El, H. Choukri, D. Naccache, M. Tunstall and C. Whelan. The Sorcerer's Apprentice Guide to Fault Attacks, In *Cryptology ePrint Archive*, Report 2004/10, 2004.
6. J. Baek and Y. Zheng. Identity-Based Threshold Decryption. In *Public Key Cryptography (PKC)*, Springer-Verlag LNCS 2947, 262–276, 2004.
7. P.S.L.M. Barreto, H. Kim, B. Lynn and M. Scott. Efficient Algorithms for Pairing-Based Cryptosystems. In *Advances in Cryptology (CRYPTO)*, Springer-Verlag LNCS 2442, 354–368, 2002.
8. D. Boneh and D. Brumley. Remote Timing Attacks Are Practical. In *12th USENIX Security Symposium*, USENIX Press, 2003.
9. D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In *SIAM Journal on Computing*, Volume 32, no. 3, 586-615, 2003.
10. M. Ciet and M. Joye. Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults. To appear in *Designs, Codes and Cryptography*, 2004.
11. R. Dutta, R. Barua and P. Sarkar, Pairing-Based Cryptographic Protocols : A Survey. In *Cryptology ePrint Archive*, Report 2004/064, 2004.
12. S. Galbraith, K. Harrison and D. Soldera. Implementing the Tate pairing. In *Algorithmic Number Theory Symposium (ANTS-V)*, Springer LNCS 2369, 324–337, 2002.
13. R. Granger, D. Page and M. Stam. On Small Characteristic Algebraic Tori in Pairing-Based Cryptography. In *Cryptology ePrint Archive*, Report 2004/132, 2004.
14. L. Goubin. A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems. In *Public Key Cryptography (PKC)*, Springer-Verlag LNCS 2567, 199–210, 2003.
15. D-G. Han, J. Lim, and K. Sakurai. On Insecurity of the Side Channel Attack on XTR. In *Symposium on Cryptography and Information Security (SCIS)*, 2004.
16. F. Hess. Efficient Identity Based Signature Schemes Based on Pairings. In *Selected Areas in Cryptography (SAC)*, Springer-Verlag LNCS 2595, 310–324, 2003.
17. P.C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology (CRYPTO)*, Springer-Verlag LNCS 1109, 104–113, 1996.
18. P.C. Kocher, J. Jaffe and B. Jun. Differential Power Analysis. In *Advances in Cryptology (CRYPTO)*, Springer-Verlag LNCS 2139, 388–397, 1999.
19. O. Kömmerling and M.G. Kuhn. Design Principles for Tamper-Resistant Smartcard Processors. In *USENIX Workshop on Smart Card Technology*, 9–20, 1999.

20. T. Messerges. Power Analysis Attacks and Countermeasures for Cryptographic Algorithms. PhD Thesis, University of Illinois, 2000.
21. K. Okeya and K. Sakurai. Power Analysis Breaks Elliptic Curve Cryptosystems Even Secure Against the Timing Attack. In *Progress in Cryptology (INDOCRYPT)*, Springer-Verlag LNCS 1977, 178–190, 2002.
22. D. Page and M. Stam. On XTR and Side-Channel Analysis. To appear in *Selected Areas in Cryptography (SAC)*, 2004.
23. K. Rubin and A. Silverberg. Torus-Based Cryptography. In *Advances in Cryptology (CRYPTO)*, Springer-Verlag LNCS 2729, 349–365, 2003.
24. M. Scott and P. Barreto. Compressed Pairings. To appear in *Advances in Cryptology (CRYPTO)*, 2004.
25. J. Silverman. The Arithmetic of Elliptic Curves. Springer-Verlag GTM 106, 1986.
26. S.P. Skorobogatov and R.J. Anderson. Optical Fault Induction Attacks. In *Cryptographic Hardware and Embedded Systems (CHES)*, Springer-Verlag LNCS 2523, 2–12, 2002.
27. N.P. Smart. An Analysis of Goubin's Refined Power Analysis Attack. In *Cryptographic Hardware and Embedded Systems (CHES)*, Springer-Verlag LNCS 2779, 281–290, 2003.

Appendix: An Overview of Pairing Based Protocols

For convenience, we present an overview of the two main pairing based protocols that are referred to in the body of the paper. In order to achieve a unified notation for the schemes, and hence make our analysis easier, these descriptions may differ slightly from the original papers.

All the schemes rely on the existence of three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_3$ all of order l, that are additive and multiplicative respectively and are selected by the Trust Authority (TA). In all cases we can actually take $\mathbb{G}_1 = \mathbb{G}_2$ and define a bilinear map

$$e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_3$$

and select some generator $P \in \mathbb{G}_1$. Finally, we select a number of cryptographic hash functions with the following signatures

$$\begin{aligned} H_1 &: \{0,1\}^* \rightarrow \mathbb{G}_1^* \\ H_2 &: \mathbb{G}_3 \rightarrow \{0,1\}^N \\ H_3 &: \{0,1\}^* \times \mathbb{G}_3 \rightarrow \mathbb{Z}_l^*. \end{aligned}$$

for some value N and a few extra ones for the Baek-Zheng scheme

$$\begin{aligned} H_4 &: \mathbb{G}_3 \rightarrow \{0,1\}^N \\ H_5 &: \mathbb{G}_1^* \times \{0,1\}^N \rightarrow \mathbb{G}_1^* \\ H_6 &: \mathbb{G}_3 \times \mathbb{G}_3 \times \mathbb{G}_3 \rightarrow \mathbb{Z}_l^* \end{aligned}$$

where l is the length of a plaintext.

5.1 Boneh-Franklin Encryption [9]

Setup Generate a random master secret $s \in \mathbb{Z}_l^*$ and set P_{TA}, the public key of the TA, to be $s \cdot P$.

Extract Define the public key for ID as $P_{ID} = H_1(ID)$ and return the private key $S_{ID} = s \cdot P_{ID}$.

Encrypt Select some random $r \in \mathbb{Z}_l^*$ and construct the ciphertext for a message M as the pair

$$C = (r \cdot P, M \oplus H_2(e(P_{ID}, P_{TA})^r).$$

Decrypt Given the ciphertext parsed as $C = (U, V)$, compute

$$M = V \oplus H_2(e(S_{ID}, U)).$$

5.2 Baek-Zheng (t, n)-Threshold Decryption [6]

Setup Generate a random master secret $s \in \mathbb{Z}_l^*$ and set P_{TA}, the public key of the TA, to be $s \cdot P$.

Extract Given an identity ID, the algorithm computes the corresponding public key as $P_{ID} = H_1(ID)$ and then returns the private key $S_{ID} = s \cdot P_{ID}$.

Distribution Given a private key S_{ID}, n decryption shares and a threshold parameter $1 \leq t \leq n < q$, the algorithm select elements

$$R_1, R_2, \ldots, R_{t-1} \in_R \mathbb{G}_1^*$$

and computes

$$F(u) = S_{ID} + \sum_{j=0}^{t-1} u^j \cdot R_j$$

for $u \in \{0\} \cup \mathbb{N}$. It then computes a private key for server Γ_i as $S_i = F(i)$ and verification key $y_i = e(P, S_i)$ for $1 \leq i \leq n$. Finally, the algorithm distributes the private key S_i and verification key y_i to server Γ_i before making y_i public to all parties.

Encrypt Given a plaintext $M \in \{0,1\}^N$ and an identity ID, the algorithm selects some $r \in_R \mathbb{Z}_l^*$ and computes the public key $P_{ID} = H_1(ID)$ followed by the value $\kappa = e(P_{ID}, P_{TA})^r$. It then computes the values

$$\begin{aligned} U &= r \cdot P \\ V &= H_4(\kappa) \oplus M \\ W &= r \cdot H_5(U, V) \end{aligned}$$

and returns the ciphertext formed as the triple $C = (U, V, W)$.

Decryption Share Generation Given a ciphertext parsed as $C = (U, V, W)$ and a private key S_i from each decryption server, the algorithm computes $h = H_5(U, V)$ and checks if the equality $e(P, W) = e(U, h)$ holds. If the test holds

– Calculate the following values

$$\begin{aligned}
\kappa_i &= e(S_i, U) \\
\tilde{\kappa_i} &= e(T_i, U) \\
\tilde{y_i} &= e(T_i, P) \\
\lambda_i &= H_6(\kappa_i, \tilde{\kappa_i}, \tilde{y_i}) \\
L_i &= T_i + \lambda_i \cdot S_i
\end{aligned}$$

for previously selected $T_i \in_R \mathbb{G}_1$.

– Return $\delta_{i,C} = (i, \kappa_i, \tilde{\kappa_i}, \tilde{y_i}, \lambda_i, L_i)$.

Otherwise

– Return $\delta_{i,C} = (i,$ "invalid ciphertext"$)$.

Decryption Share Verification Given a ciphertext parsed as $C = (U, V, W)$, a set of verification keys $\{y_1, y_2, \ldots, y_n\}$, and a decryption share $\delta_{i,C}$, the algorithm computes $h = H_5(U, V)$ and checks if the equality $e(P, W) = e(U, h)$ holds. If the test holds

– If $\delta_{i,C}$ is of the form $(i,$ "invalid ciphertext"$)$, return "invalid share".
– Otherwise parse $\delta_{i,C}$ as $(i, \kappa_i, \tilde{\kappa_i}, \tilde{y_i}, \lambda_i, L_i)$ and compute $\lambda_i\prime = H_6(\kappa_i, \tilde{\kappa_i}, \tilde{y_i})$.
  - Check if $\lambda_i\prime = \lambda_i$, $e(L_i, U)/\kappa_i^{\lambda_i\prime} = \tilde{\kappa_i}$ and $e(L_i, P)/y_i^{\lambda_i\prime} = \tilde{y_i}$.
  - If the above tests hold, return "valid share"; otherwise return "invalid share".

Otherwise

– If $\delta_{i,C}$ is of the form $(i,$ "invalid ciphertext"$)$, return "valid share"; otherwise return "invalid share".

Share Combining Given a ciphertext C and a set of valid decryption shares $\{d_{j,C}\}_{j \in \Phi}$ where $|\Phi| \geq t$ for the threshold t, the algorithm $h = H_5(U, V)$ and checks if the equality $e(P, W) = e(U, h)$ holds. If the test holds

– Compute $\kappa = \prod_{j \in \Phi} \kappa_j^{c_{0,j}^{\Phi}}$ and return $M = H_4(\kappa) \oplus V$.

Otherwise

– Return "invalid ciphertext".