Exploiting the Order of Multiplier Operands: a Low-Cost Approach for HCCA Resistance

Poulami Das, Debapriya Basu Roy, and Debdeep Mukhopadhyay

***Abstract*—Horizontal collision correlation analysis (HCCA) imposes a serious threat to simple power analysis resistant elliptic curve cryptosystems involving unified algorithms, for e.g. Edward curve unified formula. This attack can be mounted even in presence of differential power analysis resistant randomization schemes. In this paper we have designed an effective countermeasure for HCCA protection, where the dependency of side-channel leakage from a school-book multiplication with the underling multiplier operands is investigated. We have shown how changing the sequence in which the operands are passed to the multiplication algorithm introduces dissimilarity in the information leakage. This disparity has been utilized in constructing a zero-cost countermeasure against HCCA. This countermeasure integrated with an effective randomization method has been shown to successfully thwart HCCA. Additionally we provide experimental validation for our proposed countermeasure technique on a SAKURA-G platform. To the best of our knowledge, this is the first time that asymmetry in information leakage has been utilized in designing a side channel countermeasure.**

I. Introduction

Elliptic curve cryptosystems are emerging as a primary choice for securing light-weight embedded devices as it incorporates more security per key bit compared to RSA [1], thus qualifying as a less resource hungry alternative. Also with the recent explosion of internet of things (IoT), applications using light-weight hardware devices are increasing exponentially which in turn make the security of the underlying devices imperative. However the hardware implementations of cryptographic applications suffer an inevitable insecurity in terms of side-channel leakage, even though the system is theoretically protected. Side channel leakage of information through power consumption [2], electromagnetic (EM) dissipation, acoustic channel [3], etc makes the system weakly protected and may lead to complete secret key recovery. A naíve implementation of an elliptic curve (EC) scalar multiplication algorithm, consisting of sequential doubling and addition operations, can be broken through simple power analysis (SPA) [4] with only a single trace of execution. This motivates researchers to construct cryptosystems which are inherently secure against SPA. Atomic scheme algorithms have been introduced in [5], [6] which transform the doubling and addition operation into a uniform structure, such that it becomes infeasible to distinguish an addition operation from a doubling from a single power trace. However these atomic scheme algorithms still involve different formulae for addition and doubling, which has motivated researchers for further unification. In [7] a unified addition formula is designed for a Weierstrass form of elliptic curve, for both addition and doubling. While in [8] a new form of curve, named Edward curve has been built involving a complete addition formula which gives a valid elliptic curve point as output for any two curve points taken as input, thus taking care of both addition and doubling. Recent extensive research involving use of Edward curve in cryptosystems reveals its implementation-friendliness [9], [10], [11], [12]. Also it is being considered as a safe curve with respect to a number of important factors (ladder security, twist security). [13] contain details on the defined safe curve criteria. Indeed because of the presence of single formula for both point addition and point doubling, an Edward curve implementation, similarly Brier-Joye unified formula [7] is SPA resistant. We note here that there exists advanced attacks such as differential power analysis (DPA) attack [4] which can exploit a SPA-resistant implementation, thus considered as a serious threat to elliptic curve cryptography (ECC) designs. However it requires access to a significantly large number of power or EM traces of EC scalar multiplication executions, with a fixed secret key, hence this scenario is not directly applicable to ECDSA, where a secret scalar is used only once. However, the Big Mac attack by [14] introduces an advanced form of single trace attacks later termed as horizontal attacks which exposes even an SPA protected implementation. Several horizontal attack approaches followed the Big Mac analysis in [15], [16], [17] which were mainly focused on RSA based exponentiation algorithms. Authors in [18] have put forward the idea of horizontal attack in case of elliptic curve cryptography. The attack combines methodologies from the well established horizontal attack [14] and the idea of collision attack (introduced in [19]), hence termed as horizontal collision correlation analysis (HCCA) which breaks an atomic scheme ECC algorithm or a unified ECC algorithm equipped with SPA-resistance. Even when the design is protected against advanced attacks such as DPA, refined power analysis [20], address-bit differential attack [21] with effective randomization schemes suggested in [22], [23], HCCA can be launched, thus introducing genuine vulnerability in the implementation. It exploits the relation of the secret key value with a property pertaining to the underlying field multiplications involved in a point doubling and point addition operation. It is a unique property based on the sharing of operands between two field multiplications which holds irrespective of any randomization used at each iteration of the scalar multiplication.

Our contribution Our main contribution in this paper is to design a cost-effective yet adequate countermeasure that resists

The authors are with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, 721302, India, E-mail: poulamidas22@gmail.com, {deb.basu.roy,debdeep}@cse.iitkgp.ernet.in

HCCA. Our contribution in this paper can be summarized as follows

- We coin a term *order of operands* to define the sequence in which two operands are passed as parameters to a long integer multiplication routine. We show how the information leakage from a multiplication varies when the *order of operands* in a multiplication is changed. We also derive that the relation between side-channel leakage of two multiplications sharing one (two) common operand (s) is dependent on the order of operands passed to the individual multiplications. We have used two distinguishers - Pearson Correlation metric, and Euclidean Distance metric for our analysis.
- Based on this observation, we propose a countermeasure, that can be applied to the existing unified algorithms of ECC to defeat HCCA. The countermeasure involves two steps: *step* 1 prevents the first scenario of HCCA, while *step* 2 is required to resist the second scenario. *Step* 1 of our countermeasure involves no randomization, instead it converts the unified algorithm into a safer form, such that the relation between side-channel leakage of multiplications based on property of operand sharing cannot be exploited. The countermeasure requires determination of the safe sequence through our proposed algorithm. As a result, there is no additional timing and area overhead on the implementation. The *step* 2 is based on a randomization technique that is able to counter the second phase of HCCA, by using minimal overhead. We show how the implementation integrated with our proposed countermeasure becomes resistant against HCCA.
- Finally we provide extensive results of mounting HCCA on the proposed countermeasure. The results have been validated on SAKURA-G with Electromagnetic (EM) traces.

Paper organization The organization of the paper is as follows. In Section 2 we recall related works in horizontal side channel analysis. In Section 3, we provide a brief review on HCCA. Section 4 provides a theoretical validation of our countermeasure idea, followed by description of our proposed countermeasure. Section 5 includes actual experimental results of HCCA and our countermeasure. Section 6 provides a justification of our countermeasure resistivity against other horizontal attacks. In Section 7 we include a comparison note with other countermeasures. Finally we conclude in Section 8.

II. Related Work

Big Mac analysis [14] introduced the idea of applying differential power analysis along the length of a single exponentiation trace of RSA. It shows how the data dependency during the pre-computation phase can be exploited to identify exponent digits involved in a long integer multiplication during an m-ary RSA exponentiation. The vulnerability is shown to increase if the length of the key increases exposing more multiplication traces to compare with. Authors in [24] applies a novel technique of distinguishing multiplication from squaring operations based on the difference in their expected Hamming weight distribution. However its a vertical attack gathering information from several traces along the same region of a long integer multiplication. In [25] the idea of horizontal attack on an RSA exponentiation has been strengthened by exploiting a significant number of potential collision pairs obtained within a long integer multiplication, if the underlying operation is a squaring operation. Multiplication operations are expected to result in less collisions compared to squaring due to the presence of different input operands. In [15] a practical vulnerability of using scalar blinding as a DPA countermeasure has been demonstrated. Due to the sparse form of NIST prime, a portion of the secret key remains unblinded and get exposed to vertical collision analysis, the rest part of the key is recovered using horizontal attack techniques. In [17] a generic approach is introduced to break an ECC implementation with the help of one template trace per scalar bit. In [16] the vulnerability of regularized algorithms such as Montgomary Ladder [26], Joye's Add-Only scalar multiplication [27] is highlighted, based on collisions of intermediate results obtained from consecutive iterations. In later section we demonstrate the resistance of our countermeasure from the above mentioned horizontal attacks.

III. Preliminaries

A. Horizontal Collision Correlation Analysis

In this section we discuss the ideology behind horizontal collision correlation analysis (HCCA). First we proceed to explain the attack methodology with the help of an illustration, followed by a summarization of the attack. Before moving to the example describing HCCA, a closer look is given to the field operations underlying ECC doubling and addition operations. It is evident that, ECC point addition and point doubling operations are associated with a number of field multiplication and field addition operations. The underlying field multiplications play an important role in HCCA. The attack is based on the *assumption*: *The adversary can detect when two field multiplications have at least one operand in common* [18]. Without loss of generality we consider distinct field elements as A, B, C, D to be used as operands to field multiplications. Then the possible field multiplication pairs will take one of the following forms: 1) $A \times B$, $C \times D$ sharing no common operand, 2) $A \times B$, $C \times B$ sharing one common operand, 3) $A \times B$, $A \times B$ where both the operands are same. Based on the above class of multiplication pairs, we define the following properties of field multiplication pairs:

- *property* 1*:* when a pair of multiplications (m_i, m_j) share one (two) common operand (s) among themselves.
  - *property* $1a$*:* when a pair of multiplications (m_i, m_j) share exactly one common operand among themselves. For e.g., the pair $(A \times B, C \times B)$ satisfies *property* $1a$.
  - *property* $1b$*:* when a pair of multiplications (m_i, m_j) share exactly two operands, i.e. they denote the same multiplications. For e.g., the pair $(A \times B,\ A \times B)$ satisfies *property* $1b$.
- *property* 2*:* when a pair of multiplications (m_i, m_j) share no common operand among themselves. For e.g., the pair

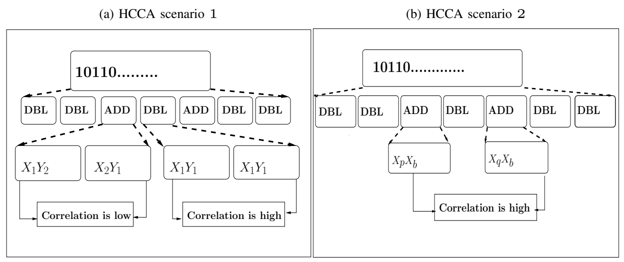


Fig. 1: Horizontal Collision Correlation Analysis (HCCA)

($A \times B$, $C \times D$) having independent operands satisfies *property* 2.

Such relation between field multiplication operations is exploited to identify the doubling and addition operations computed during an ECC scalar multiplication, which in turn is directly dependent on the secret key. Hence identification of doubling and addition operations leads to the recovery of the underlying unknown key. Now we proceed to illustrate the possible scenarios of HCCA. Figure. $1(a)$ illustrates an occurrence of first phase of HCCA. Without loss of generality, a key sequence has been considered as $10110\ldots$ which can be expanded as DBL, DBL, ADD, DBL, ADD, DBL, DBL,..., where DBL represents a point doubling operation, while ADD denotes a point addition operation as shown in Figure $1(a)$. Each of the ADD/ DBL operations consist of underlying field additions and field multiplications. For an instance, it can be observed in Figure $1(a)$, that there exists a multiplication pair (X_1Y_2, X_2Y_1) within the addition operation, satisfying *property* 2 of sharing operands. While a pair (X_1Y_1, X_1Y_1) can be found in case of doubling satisfying the *property* $1b$ of sharing operands. Now, according to [18] if the correlation between the power traces of two concerned multiplication pairs be considered, the multiplication pair (X_1Y_2, X_2Y_1) should give low correlation value, with respect to the correlation value obtained from the multiplication pair (X_1Y_1, X_1Y_1) as shown in [18]. If significant difference between the correlation values is obtained, then the doubling and addition operations can be successfully identified, leading to the complete secret key recovery.

It can be observed that for *scenario* 1 to hold we require *property* 1 to hold for a multiplication pair in case of doubling operation, and *property* 2 to hold for a multiplication pair within addition operation, or vice versa. In absence of such favourable situation, HCCA *scenario* 1 cannot be applied.

In case of the Edward curve unified formula written in the form given in table I we observe that both the doubling and addition operation contains multiplication pair satisfying *property* 1, hence distinction cannot be done between the two operations on the basis of operand sharing property of field multiplications. For an instance there exists a multiplication pair in case of addition as ($R_1 \times R_6$, $R_1 \times R_7$) which actually have the value (($Z_1Z_2) \times (Z_1Z_2 - dX_1X_2Y_1Y_2)$, $Z_1Z_2 \times (Z_1Z_2 + dX_1X_2Y_1Y_2)$), sharing the operand R_1 or Z_1Z_2. While in case of doubling one multiplication pair can be found as ($R_1 \times R_6$, $R_1 \times R_7$), with the value as (($Z_1^2) \times (Z_1^2 - dX_1^2Y_1^2)$, $(Z_1^2) \times (Z_1^2 + dX_1^2Y_1^2)$) having the common operand R_1 as Z_1^2.

Edward curve formula with Intermediate steps in Addition operation	
$(X^2+Y^2)Z^2 = c^2(Z^2+dX^2Y^2)$	$R_1 = Z_1Z_2$, $R_2 = R_1^2$,
$c \neq 0$, $\sqrt{d} \notin F_p$	$R_3 = X_1X_2$, $R_4 = Y_1Y_2$
	$R_5 = dR_3R_4$, $R_6 = R_2 - R_5$
	$R_7 = R_2 + R_5$
	$X_3 = R_1R_6((X_1+Y_1)(X_2+Y_2) - R_3 - R_4)$
	$Y_3 = R_1R_7(R_4 - R_3)$
	$Z_3 = R_6R_7$

Addition steps (called with two different points)	Doubling steps (called with same point)
$Z_1 \times Z_2$	$Z_1 \times Z_1$
$Z_1Z_2 \times Z_1Z_2$	$Z_1^2 \times Z_1^2$
$X_1 \times X_2$	$X_1 \times X_1$
$Y_1 \times Y_2$	$Y_1 \times Y_1$
.	.
.	.
$\mathbf{R_1} \times \mathbf{R_6} = (Z_1Z_2) \times (Z_1Z_2 - dX_1X_2Y_1Y_2)$	$\mathbf{R_1} \times \mathbf{R_6} = (Z_1^2) \times (Z_1^2 - dX_1^2Y_1^2)$
$\mathbf{R_1} \times \mathbf{R_7} = (Z_1Z_2) \times (Z_1Z_2 + dX_1X_2Y_1Y_2)$	$\mathbf{R_1} \times \mathbf{R_7} = (Z_1^2) \times (Z_1^2 + dX_1^2Y_1^2)$
.	.
.	.

TABLE I: Comparison of operand sharing between addition and doubling in Edward curve equation [28]

Alternatively adversary proceeds to mount second version of HCCA or *scenario* 2. From a standard left-to-right double-and-add algorithm (Algorithm 1) it can be observed that doubling operation DBL involves a single parameter which changes at every iteration. On the other hand, the addition routine ADD takes two parameters as input, one of which is always the base point of the curve (B_p in Algorithm 1). Based on this fact, we proceed to describe the attack methodology with the help of Figure $1(b)$. The base point B_p is denoted by the projective coordinates as (X_b, Y_b, Z_b), the other two points P, Q concerned with the two additions are given as (X_p, Y_p, Z_p) and (X_q, Y_q, Z_q) respectively. When Algorithm 1 is run with an underlying Edward curve equation the two additions will be performed as ADD(P, B_p) and ADD(Q, B_p). There will exist two field multiplications (X_pX_b) and (X_qX_b) underlying in the corresponding addition operations sharing operand X_b, thus satisfying *property* $1a$. However in case of doubling, due to the variation of the input point with every iteration such a scenario will not arise. Evidently all the additions and doublings can be identified following the above correlating mechanism which will lead to the recovery of the secret key.

Algorithm 1: Left-to-Right Scalar Multiplication Algorithm using Edward curve

```
Data: Point $P$, scalar $k = \{k_{m-1}, k_{m-2}, k_{m-3} \ldots k_2, k_1, k_0\}$, where
      $k_{m-1} = 1$
Result: $kP$
$Q = P$
for $i = m-2$ to $0$ do
    $Q = ADD(Q, Q)$
    // in general, doubling is called as: $Q = DBL(Q)$
    if $k_i == 1$ then
        $Q = ADD(Q, B_p)$
    end
end
Return $Q$
```

We summarize below the above illustrated HCCA scenarios. An ECC point doubling operation can be decomposed into a sequence of n_d multiplications given as: $\{d_1, d_2, \ldots, d_{n_d}\}$, denoted by the set_d. Equivalently, an addition operation consists of a sequence of n_a multiplications given as: $\{a_1, a_2, \ldots, a_{n_a}\}$, denoted by set_a. Now we define *property* 3 for the above developed sets as: S be a set of n multiplications denoted by $\{ m_1, m_2, \ldots, m_n \}$, such that $\exists$ at least one pair (m_i, m_j), where m_i and $m_j \in S$, $i \neq j$, which satisfies *property* 1 of sharing operands, then the set S is said to satisfy the *property* 3. First phase of HCCA or *scenario* 1 is based on the following *condition* 1*:* {Only one of the sets set_d and set_a should satisfy the set property 3}. If *condition* 1 holds, the adversary aims at differentiating between an addition and doubling operation. Consequently the adversary can recognize all the doubling and addition operations in a sequential manner by launching HCCA. If *condition* 1 does not hold, adversary may mount the *scenario* 2 of HCCA. Note that the basis of *scenario* 2 of HCCA is based on the fact: *one of the addition parameters is always the base point*, which holds independent of the underlying curve equation or the unified algorithm steps involved in the scalar multiplication. The method of correlating between a pair traces has been taken in [18] as the Pearson Correlation Distinguisher. However we show in later sections that better results of HCCA and validation of our proposed countermeasure can obtained from Euclidean Distance Distinguisher.

IV. Our Proposed Countermeasure

We propose here a two-fold countermeasure technique which ensures the resistance of an unified ECC algorithm against horizontal collision correlation attack (HCCA). The *step* 1 of our proposed countermeasure centers around the concept of reordering of field operands underlying a field multiplication. It involves transforming the ECC point doubling and point addition operations into a secure form, such that even if *condition* 1 holds, *it is not revealed to the adversary*. In other words, the information of one of the operations satisfying *property* 3 is hidden through our implementation. The resultant implementation is thus resistant against *scenario* 1 of HCCA. However still the design is vulnerable to the *scenario* 2. We incorporate *step* 2 to the existing design, by introducing an effective randomization scheme. Our ECC implementation integrated with our proposed countermeasures becomes resistant against both scenarios of HCCA. Our *step* 1 requires zero overhead of resources in case of the Edward curves unified formula as well as Brier-Joye unified formula. It is based on an observation that the leakage from the power consumption is dependent on the ordering of operands in a field multiplication. This discrepancy in leakage occurs as the ordering of the operands brings in asymmetry in the leakage, which we exploit to develop our countermeasure. We note that although the concept of asymmetric leakage has been addressed in [29] in case of multipliers, however authors of [29] don't exploit its applicability. To the best of our knowledge, this is the first countermeasure design which utilizes asymmetry in information leakage of multiplier operands.

A. Asymmetric Leakage of Field Multiplication

In this section we explain our theoretical rationale behind the asymmetric leakage of field multiplications, which contribute in constructing our countermeasure scheme. We begin our discussion with an introduction to Long Integer Multiplication (LIM) shown in Algorithm 2. The long integer multiplication routine is called to compute underlying field multiplications involved in the ECC point addition, doubling operations. The LIM takes two field operands X, Y as input and outputs their product XY. Each of the field operands passed as parameter in the LIM routine consists of underlying t words, each of size w. The result can be of size $2t$, and is stored in a register of length $2t$ words. The algorithm is run $\mathcal{O}(t^2)$ times.

Algorithm 2: Long Integer Multiplication algorithm(LIM)

```
Data: : {$X = (X[t], X[t-1], ...., X[1])_{2^w}$} , {$Y = (Y[t], Y[t-1], ...., Y[1])_{2^w}$}
Result: : {$X.Y$}
begin
    for $i \leftarrow 1$ to $2t$ do
        $R[i] = 0$
    end
    for $i \leftarrow 1$ to $t$ do
        $C = 0$ ;
        for $j \leftarrow 1$ to $t$ do
            $(U, V)_{2^w} = (U, V)_{2^w} + C$ ;
            $(U, V)_{2^w} = (U, V)_{2^w} + R[i + j - 1]$ ;
            $R[i + j - 1] = V$ ;
            $C = U$;
        end
        $R[i + t] = C$ ;
    end
    return $R$ ;
end
```

To establish the reasoning behind asymmetry in leakage of field multiplications, we introduce here an information leakage model which will guide us towards the theoretical basis of our countermeasure. Generally, in case of an iterative algorithm, a calculation C_i is identified, which is operated at each iteration of the algorithm execution. The output O_i of the calculation C_i is updated at every iteration to a specific register location. The value of the output O_i computed and stored at each iteration leaks an information. This information leakage is denoted as $l(O_i)$, which can be approximated using a function of O_i i.e $f(O_i)$. The information leakage at each iteration gets augmented iteratively to result in a vector $< f(O_i) >$. In case of Algorithm 2, we consider an instance of the long integer multiplication run with input field operands $A = (a_t, a_{t-1}, \ldots, a_2, a_1)$, $B = (b_t, b_{t-1}, \ldots, b_2, b_1)$ which results in the output $A \times B$. At $(i,j)^{th}$ iteration we can associate the calculation $C_{i,j}$ with the partial product computation $a_i \times b_j$. The output of the partial product $O_{i,j} = a_i b_j$ is stored in every iteration, which leaks an information $l(O_{i,j})$. We assume that the information leakage $l(O_{i,j})$ follows Hamming weight power model. As a result the function $f(O_{i,j})$ is approximated with the help of the Hamming weight of the output value $O_{i,j}$. So we consider $f(O_{i,j}) = H(O_{i,j})$, where $H(x)$ implies the Hamming weight of the value x. Based on the leakage model considered, the information leakage of long integer multiplication can be represented by an augmented vector, denoted as $< H(O_i) >$, or $< H(a_i b_j) >$. It is evident from Algorithm 2 that the sequence of partial products changes when the order of the operands passed as parameter to the LIM routine is swapped. We consider the information leakage $l(a_i, b_j)$ at each iteration, corresponding to partial product $a_i \times b_j$ computed during an instance of LIM(A,B) execution. It is observed that the vector is formed as $< l_{(a_0,b_0)}$, $l_{(a_0,b_1)}, \ldots$, $l_{(a_0,b_{t-1})}, \ldots$,

$l_{(a_{t-1},b_{t-1})}$ >. While the one obtained during computation of LIM(B, A) can be presented as < $l_{(b_0,a_0)}$, $l_{(b_0,a_1)}$,. . ., $l_{(b_0,a_{t-1})}$,. . ., $l_{(b_{t-1},a_{t-1})}$ >. This asymmetry in the sequence of the two vectors contribute as a distinguisher between two multiplications. The following lemma has been introduced to emphasize the dependence of information leakage from the intermediate partial products $l(a_i, b_j)$ on the input operands A, B during the computation of LIM(A, B).

Lemma IV.1. *The probability of collision of Hamming weight of a pair (mn, pn) is more than the probability of collision of the pair (mn, pq).*

Proof. Let $Y = (y_1, y_2)$ be the event that the Hamming weight of the output of two multiplications y_1 and y_2 collide. Let X be a random variable such that ($X = a$) denotes the colliding Hamming weight of the event ($Y = (y_1, y_2)$) is a. If each operand are of size w bits, then the number of all possible values of one operand is 2^w.
Case 1: The probability of collision of the pair (mn, pn) is computed as $P_1 = P(Y = (mn, pn)) = \{P(X = 0) \cup P(X = 1) \cup P(X = 2) \cup \ldots \cup P(X = w)\} = P(X = 0) + P(X = 1) + P(X = 2) + \ldots + P(X = w)$. Now, $P(X = a) = \frac{n_a}{(2^w)^3}$, where n_a is the number of cases when the two multiplications collide with a Hamming weight value a. Since we have three distinct operands in the two multiplications the total number of multiplication pairs formed is $(2^w)^3$. Thus $P_1 = \frac{n_0}{(2^w)^3} + \frac{n_1}{(2^w)^3} + \ldots + \frac{n_w}{(2^w)^3} = \frac{n_0+n_1+\ldots+n_w}{(2^w)^3}$.
Case 2: The probability of collision of the pair (mn, pq) is computed as $P_2 = P(Y = (mn, pq)) = \{P(X = 0) \cup P(X = 1) \cup P(X = 2) \cup \ldots \cup P(X = w)\} = P(X = 0) + P(X = 1) + P(X = 2) + \ldots + P(X = w)$. Now, $P(X = a) = \frac{n_a}{(2^w)^4}$, where n_a is the number of cases when the two multiplications collide with a Hamming weight value a. Since we have four distinct operands in the two multiplications the total number of multiplication pairs formed is $(2^w)^4$. Thus $P_2 = \frac{n_0}{(2^w)^4} + \frac{n_1}{(2^w)^4} + \ldots + \frac{n_w}{(2^w)^4} = \frac{n_0+n_1+\ldots+n_w}{(2^w)^4}$.
From the probability values of P_1 and P_2, we observe that $P_1 > P_2$. Hence Proved. □

To calculate the relationship between information leakage of two long integer multiplications we have considered the following metrics

1) Pearson Correlation Metric: : Considering underlying field operands as: A, B, A', B', the correlation between two long integer multiplications LIM(A, B) and LIM(A', B') can be approximated with the Pearson correlation coefficient computed between two vectors $< H(a_i b_j) >$, $< H(a'_i b'_j) >$ (following similar notation as above). Let us denote the two vectors as $H(AB)$ and $H(A'B')$ respectively. The correlation is obtained as follows

$$\rho = \frac{Covariance(H(AB), H(A'B'))}{\sqrt{Variance(H(AB))}\sqrt{Variance(H(A'B'))}} \tag{1}$$

We now onwards denote the covariance between two vectors as $cov(H(AB), H(A'B')$, variance as $var(H(AB))$. The standard deviation from the information leakage of a long integer multiplication LIM(A, B) is denoted as $std(AB)$. It is obtained as below

$$std(AB){=}std(<H(AB)>){=}\sqrt{\frac{\sum_{i=0,j=0}^{t-1} H(a_i b_j)^2}{t^2} - \left(\frac{\sum_{i=0,j=0}^{t-1} H(a_i b_j)}{t^2}\right)^2} \tag{2}$$

We define four correlations based on following long integer multiplications LIM(A, B), LIM(B, C), LIM(C, B), LIM(C, D). The following correlation is obtained from LIM(A, B) and LIM(C, B)

$$\rho_1 = \frac{\left(\frac{\sum_{i=0,j=0}^{t-1} H(a_i b_j)H(c_i b_j)}{t^2}\right) - \left(\frac{\sum_{i=0,j=0}^{t-1} H(a_i b_j)}{t^2}\right)\left(\frac{\sum_{i=0,j=0}^{t-1} H(c_i b_j)}{t^2}\right)}{std(AB)std(CB)} \tag{3}$$

where we denote $\sum_{i=0,j=0}^{t-1} H(a_i b_j)H(c_i b_j)$ as α, where α can be expanded as

$$\begin{aligned}\alpha =&(H(a_0b_0)H(c_0b_0) + H(a_0b_1)H(c_0b_1) + \ldots + H(a_0b_{t-1})(c_0b_{t-1})\\ &+ H(a_1b_0)H(c_1b_0) + \ldots + H(a_{t-1}b_{t-1})H(c_{t-1}b_{t-1}))\end{aligned} \tag{4}$$

The following correlation is obtained from LIM(A, B) and LIM(B, C)

$$\rho_2 = \frac{\left(\frac{\sum_{i=0,j=0}^{t-1} H(a_i b_j)H(b_i c_j)}{t^2}\right) - \left(\frac{\sum_{i=0,j=0}^{t-1} H(a_i b_j)}{t^2}\right)\left(\frac{\sum_{i=0,j=0}^{t-1} H(b_i c_j)}{t^2}\right)}{std(AB)std(BC)} \tag{5}$$

where $\sum_{i=0,j=0}^{t-1} H(a_i b_j)H(b_i c_j)$ can be expressed as β, which takes the form

$$\begin{aligned}\beta =&(H(a_0b_0)H(b_0c_0) + H(a_0b_1)H(b_0c_1) + \ldots + H(a_0b_{t-1})h(b_0c_{t-1})\\ &+ H(a_1b_0)h(b_1c_0) + \ldots + H(a_{t-1}b_{t-1})H(b_{t-1}c_{t-1})).\end{aligned} \tag{6}$$

Here we consider the correlation coefficient between a multiplication pair with *property* 2, computed from LIM(A, B) and LIM(C, D).

$$\rho_3 = \frac{\left(\frac{\sum_{i=0,j=0}^{t-1} H(a_i b_j)H(c_i d_j)}{t^2}\right) - \left(\frac{\sum_{i=0,j=0}^{t-1} H(a_i b_j)}{t^2}\right)\left(\frac{\sum_{i=0,j=0}^{t-1} H(c_i d_j)}{t^2}\right)}{std(AB)std(CD)} \tag{7}$$

where $\sum_{i=0,j=0}^{t-1} H(a_i b_j)H(c_i d_j)$ is coined as γ, represented as

$$\begin{aligned}\gamma =&(H(a_0b_0)H(c_0d_0) + H(a_0b_1)H(c_0d_1) + \ldots + H(a_0b_{t-1})H(c_0d_{t-1})\\ &+ H(a_1b_0)H(c_1d_0) + \ldots + H(a_{t-1}b_{t-1})H(c_{t-1}d_{t-1})).\end{aligned} \tag{8}$$

We develop here few Lemmas which will be required consequently to support the theoretical foundation of our countermeasure. Few terms which will be used in the following Lemma are introduced here. Four mutually distinctive word multipliers are considered as m, n, p, q which will be used as operands to word level multiplications such as mn, pn, and pq. As defined above, A and B denote two field multiplications operands which will be used as parameters in the LIM routine. Now we proceed to the Lemmas.

Lemma IV.2. *The standard deviation of a Hamming weight vector obtained from LIM(A, B) is same as that obtained as LIM(B, A), i.e std(AB) = std(BA).*

Proof. The vector composed from leakage information of LIM(A, B) can be expanded as $< H(a_0, b_0)$, $H(a_0, b_1)$,. . ., $H(a_0, b_{t-1})$,. . ., $H(a_{t-1}, b_{t-1}) >$. While the vector obtained from leakage information of LIM(B, A) is represented as $<$ $H(b_0, a_0)$, $H(b_0, a_1)$,. . ., $H(b_0, a_{t-1})$,. . ., $H(b_{t-1}, a_{t-1}) >$. It can be observed that the two vectors are two different arrangements of same underlying elements. As a result $std(AB)$ = $std(BA)$. Hence proved. □

If we denote $mean(X)$ as the mean value of a vector X, on the basis of a similar argument we can also show that $mean(AB) = mean(BA)$.

Lemma IV.3. *$cov(H(AB), H(CB)) \neq cov(H(AB), H(BC))$. When $C = A$, $cov(H(AB), H(AB)) \neq cov(H(AB), H(BA))$.*

Proof. The two covariances $cov(H(AB), H(CB))$ and $cov(H(AB), H(BC))$, can be represented as

$$cov(H(AB), H(CB)) = \alpha - mean(AB)mean(CB) \tag{9}$$

$$\begin{aligned} cov(H(AB), H(BC)) =& \beta - mean(AB)mean(BC) \\ =& \beta - mean(AB)mean(CB) \end{aligned} \tag{10}$$

Since, from Lemma 2. $mean(BC) = mean(CB)$, the second term in both the covariances are $mean(AB)mean(CB)$. Also, from equations 4 and 6, $\alpha \neq \beta$, as a result we can conclude

$$cov(H(AB), H(CB)) \neq cov(H(AB), H(BC)).$$

When $C = A$: from equation 4 and 6, we show that still $\alpha \neq \beta$. The value of α can be expressed as

$$\begin{aligned} \alpha =& (H(a_0b_0)H(a_0b_0) + H(a_0b_1)(a_0b_1) + \ldots + H(a_0b_{t-1})H(a_0b_{t-1}) \\ &+ H(a_1b_0)H(a_1b_0) + \ldots + H(a_{t-1}b_{t-1})H(a_{t-1}b_{t-1})) \\ =& (H(a_0b_0)^2 + H(a_0b_1)^2 + \ldots + H(a_0b_{t-1})^2 \\ &+ H(a_1b_0)^2 + \ldots + H(a_{t-1}b_{t-1})^2) \end{aligned} \tag{11}$$

While β can be reduced as

$$\begin{aligned} \beta =& (H(a_0b_0)H(b_0a_0) + H(a_0b_1)H(b_0a_1) + \ldots + H(a_0b_{t-1})H(b_0a_{t-1}) \\ &+ H(a_1b_0)H(b_1a_0) + \ldots + H(a_{t-1}b_{t-1})H(b_{t-1}a_{t-1})) \\ =& (H(a_0b_0)^2 + H(a_0b_1)(b_0a_1) + \ldots + H(a_0b_{t-1})H(b_0a_{t-1}) \\ &+ H(a_1b_0)h(b_1a_0) + \ldots + H(a_{t-1}b_{t-1})^2). \end{aligned} \tag{12}$$

From equations 11 and 12, we can observe that $\alpha \neq \beta$. As a result when $C = A$, we can conclude similarly that

$$cov(H(AB), H(AB)) \neq cov(H(AB), H(BA)).$$

□

Lemma IV.4. *$\rho_1 > \rho_2$ for the case: $A = C$.*

Proof. When $A = C$, precisely the two multiplications pairs considered are: (LIM(A, B), LIM(A, B)) and (LIM(A, B), LIM(B, A)). The correlation ρ_1 between (LIM(A, B), LIM(A, B)) can be computed as

$$\begin{aligned} \rho_1 =& \frac{cov(H(AB), H(AB))}{\sqrt{var(H(AB))}\sqrt{var(H(AB))}} \\ =& \frac{var(H(AB))}{var(H(AB))} \text{ , since } cov(X, X) = var(X) \\ =& 1 \end{aligned}$$

While, the correlation ρ_2 between (LIM(A, B), LIM(B, A)) can be computed as

$$\begin{aligned} \rho_2 =& \frac{cov(H(AB), H(BA))}{\sqrt{var(H(AB))}\sqrt{var(H(BA))}} \\ =& \frac{cov(H(AB), H(BA))}{var(H(AB))} \\ <& 1 \end{aligned}$$

Since from Lemma 3,

$$cov(H(AB), H(AB)) \neq cov(H(AB), H(BA)).$$

Hence it is proved that $\rho_1 > \rho_2$, when $C = A$. □

With the help of the lemmas discussed above, we make the following observations:

Observation 1: $\rho_1 \neq \rho_2$ From equations 3, 5, we can recollect the mathematical forms of ρ_1 and ρ_2. From Lemma 2, we can conclude that $std(AB) = std(BA)$. As a result, the denominators in case of both the correlations are equal. From Lemma 3 we have the result that

$$Cov(H(AB), H(CB)) \neq Cov(H(AB), H(BC)).$$

Consequently numerators of the two correlations are unequal. Also, since From Lemma 2, $mean(AB) = mean(BA)$, the difference in value arises from the unequal values of α and β. We give a closer look at the forms of α and β to observe that: 1) each term in α takes the form $H(a_ib_j)H(c_ib_j)$ where the word multiplications share operand b_j. 2) each term in β is of the form $H(a_ib_j)H(b_ic_j)$, where the word multiplications have no common operand. Utilizing Lemma 1, we can conclude that each term in α has a higher probability of collision with respect to each term in β.

Observation 2: $\rho_2 \approx \rho_3$ To make a comparison between the values of ρ_2 and ρ_3, we look at the form of each of the terms present in the two equations take: 1) each term in β is of the form $H(a_ib_j)H(b_ic_j)$, where the word multiplications have no common operand. 2) each term in γ is of the form $H(a_ib_j)H(c_id_j)$, where the word multiplications are devoid of any common term. We conclude from our observation that, the two correlation coefficients take similar form.

Observation 3: $\rho_1 > \rho_2$ for a multiplication pair with *property* 1b A multiplication pair satisfying *property* 1*b*, implies same multiplications are being computed. From Lemma 4, we obtain that in such a case ρ_1 will always be greater than ρ_2 irrespective of the underlying field element values involved. Hence $\rho_1 > \rho_2$ occurs with high probability in such a case.

From the above observations, the importance of ordering of operands in underlying field multiplications can be inferred. Based on our inference, we suggest that the information leakage due to sharing of operands can be hidden by operand reordering. This fact has been exploited in designing *step* 1 of our countermeasure which will be explained in the following subsection. We introduce next the Euclidean distance metric for evaluation of our leakage analysis.

2) Euclidean Distance Metric: Euclidean distance metric has been advantageous in implementing the Big Mac attack in [14], also it has been utilized in a subsequent horizontal attack introduced in [25]. We also consider evaluation of our leakage analysis with Euclidean distance metric. For a pair of long integer multiplications LIM(A, B) and LIM(A', B'), the Euclidean distance between the two information leakage vector $H(AB)$ and $H(A'B')$ can be represented as vector d, where

$$d_{ij} = \sqrt{(H(a_i b_j) - H(a'_i b'_j))^2} \tag{13}$$

The mean Euclidean distance obtained accumulating the individual information leakage values can be obtained as

$$d_{mean} = \frac{1}{l^2} \sum_{n=1}^{l^2} d_{ij} \tag{14}$$

We note here that our theoretical observation made, based on the Pearson correlation metric will be valid under Euclidean metric also, as both have been defined on same underlying leakage model. However in later section we show that practical results reflect better validation of the HCCA attack implementation and our proposed countermeasure in case of Euclidean metric than the previously discussed Pearson correlation metric. We have deduced the reason behind such findings in the Experimental section (Section V). We move on to discuss our countermeasure idea in the following subsection.

B. *Preventing scenario* 1 *by choosing safe sequence*

Step 1 is designed on the basis of the idea of reordering of operands discussed in the previous subsection. It attempts to transform the series of field multiplications underlying ECC point doubling and point addition operation into a HCCA - resistant form. In other words, it makes the implementation secure against *scenario* 1 of HCCA. As can be noted in section III-A, an ECC implementation becomes vulnerable to *scenario* 1 of HCCA if only one of the addition or doubling operation satisfies *property* 3. The idea is to alter the operation containing *property* 3, into a form where information regarding operand sharing between field multiplications is hidden. Consequently it is not revealed to the adversary whether any doubling or addition operation contains *property* 3 or not. Hence the basis of distinction between doubling and addition operation is concealed. It should be noted that the transformation technique mainly involves rearrangement of multiplication operands. This process does not incorporate any randomization or any extra operation. Therefore the cost of this countermeasure step is zero in terms of area as well as timing overhead. Moreover, the order of operands are decided beforehand and can be precomputed before implementing the design, requiring only one time effort from the designer's point of view. We design an algorithm, named *safe_sequence_converter* routine presented in Algorithm 3 which takes care of the transformation process of *step* 1. We proceed to portray our transformation mechanism through an illustration, which will be followed by a description of our designed Algorithm 3.

We have considered the Edward curve unified formula shown in [28] for explaining our conversion scheme. It can be noted that the Edward curve unified formula involves a single formula which is used for both addition and doubling.

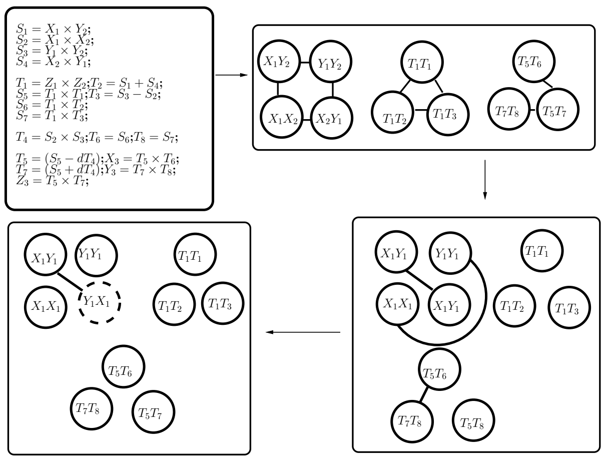


Fig. 2: Safe sequence transformation of Edward curve formula

It underlies a series of field multiplication operations which have been listed in Figure 2. We note that the multiplications are written with respect to the addition operation, i.e when two distinct points (X_1, Y_1, Z_1), and (X_2, Y_2, Z_2) are taken as input. To construct a safe sequence we need to find out which are the multiplications which share operands among themselves. To do so, we construct an undirected graph with the individual multiplications as the graph vertices, whereas an edge is constructed between two graph vertices if the two underlying multiplications satisfy *property* 1 of sharing operands (*edge property*). We observe in the Figure 2 how edges are formed between (X_1X_2, X_1Y_2), (X_1X_2, X_2Y_1), (Y_1Y_2, X_1Y_2) and so on. Furthermore, we witness that the graph is not completely connected, instead it is composed of a number of islands. One may argue that, multiplications such as T_5T_6 involve operand T_5 which is of the form T_1T_2, so it is sharing a common operand T_1. This is actually not true because, the multiplication output of $(T_1T_2)\ mod\ F_p$, where F_p is the underlying field prime, is stored in the location T_5, and hence it is statistically independent from T_1. Now we make a crucial observation that, the operand sharing obtained from the graph considered reveals all the operand sharing multiplications which will be present in the addition operation. But if we consider the graph corresponding to the doubling operation where points (X_1, Y_1, Z_1), and (X_2, Y_2, Z_2) are the same, it can be observed that the previous operand sharing will still be present along with some possible extra operand sharing vertices. So the operand sharing edges obtained from the addition operation graph illustrated above are the edges *common* to both addition and doubling operations. As a result, *they don't qualify in distinguishing between addition and doubling operations.* Evidently, the operand sharing edges which are found only in case of doubling operation may contribute in the distinction. To get a closer look we consider the complements of the islands of our previously constructed graph. Note that we are not interested in the edges between islands in the complement graph because they don't share operands among themselves. We also replace the vertex values with the respective forms of doubling operation. For e.g. X_1Y_2 will be replaced with X_1Y_1. The complement of the islands are considered here to concentrate on those edges which will be formed only in case of doubling operation. However the complement of the islands will include both essential edges (for e.g edge between two vertices each containing value X_1Y_1) as well as redundant edges (for e.g. edge between two vertices with values X_1X_1 and Y_1Y_1 respectively which do not satisfy the *edge property*). We remove the redundant edges, and look only at the essential edges because they are the ones which will help in distinguishing and addition operation from a doubling operation. In this case, doubling operation involves X_1Y_1, X_1Y_1 operated twice, which are satisfying *property* $1b$. On the other hand, addition operation consists of two underlying multiplications X_2Y_1, X_1Y_2 satisfying *property* 2 of sharing operands. Thus they successfully depict *scenario* 1 of HCCA. Based on our ***observation* 2** and ***observation* 3**, we rearrange the multiplications as X_1Y_1 and Y_1X_1, so that the their operand sharing property remains hidden. As was observed in subsection 3, the information leakage for the pair LIM(X_1, Y_1), LIM(Y_1, X_1) will be similar to that of the pair LIM(X_2, Y_1), LIM(Y_1, X_2). (here we refer to the long integer multiplication routine LIM). So we suggest to swap the order of operands of the second multiplication. From lemma IV.5 we get that the problem of swapping operands of field multiplications can be solved by the problem of two-colorability of a graph. So if the final reduced graph with the islands containing essential edges be two-colorable, then we proceed to color the graph with two colors, and eventually swap the operands of those vertices which belong to the class of one particular color.

In a similar fashion, we transform the Brier-Joye unified formula shown in [7] into a secure structure. The transformation steps corresponding to the Brier-Joye formula is portrayed in Figure 3.

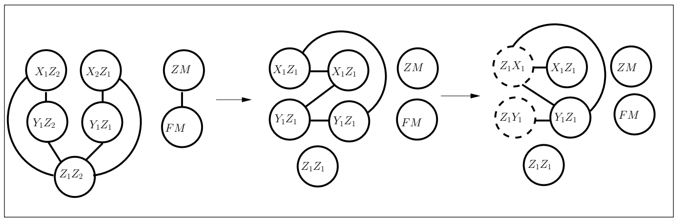


Fig. 3: Safe sequence transformation of Brier-Joye unified formula

Before proceeding to state the following lemma IV.5, we give here a rationale behind the *operand swapping problem* formulation. In our *operand swapping problem*, we need to identify a set of vertices which need to go through operand swapping, keeping other vertices intact as before so that the overall set reaches a secure form. So it is depictable that the vertex set needs to be partitioned into two sets. The set of vertices which requires operand swapping is called the *swap set*. while the other set is named as *uninterrupted set*. Also it can be perceived that in any edge, since the edge has been created due to operand sharing of two vertices, one of the vertex of the edge should be swapped, thus should belong to *swap set*. While the other vertex should belong to the *uninterrupted set*. Furthermore, there does not exist an edge such that both of their end vertices belong to the *swap set*, or the *uninterrupted set*. Suppose there exists one such edge, then if both vertices belong to the *swap set* then then it implies in case of both the vertices, the vertex operands have been swapped. But this is equivalent to the state before swapping. For e.g. it means a vertex pair (X_1Y_1, X_1Y_1) has been swapped to (Y_1X_1, Y_1X_1), which does not solve our aim of information masking through operand swapping. This is because the correlation between both the mentioned pairs will be higher with respect to the pair (X_1Y_1, Y_1X_1), as has been proved in lemma IV.4. From this it directly follows why must the vertex ends of any edge belonging to the set E should not belong to the same set (*swap set* or *uninterrupted set*). Naturally, it is alos understood why the vertices belonging to either *swap set* or *uninterrupted set* do not contain any edge between themselves. Now we define the *operand swapping problem* more formally followed by stating the *Two-colorability problem of graph.*

Operand swapping problem or problem a: Given an undirected graph G denoted by the set $\{V,\ E\}$, whether there exists a partition of V as (V_1, V_2) with following conditions: 1) V_1 or *swap set*, consists of elements as $\{v \mid$ operands of v should be swapped$\}$. 2) V_2 or *uninterrupted set*, can be presented as $\{v \mid$ operands of v should not be disturbed$\}$. 3) the edge set E is of the form $\{e \mid e = (v_i, v_j)$, where $(v_i \in V_1, v_j \in V_2)$ or $(v_i \in V_2, v_j \in V_1)\}$.

Two-colorability problem of graph or problem b: Given a graph G as set $\{V,\ E\}$, whether the vertices of the graph can be colored with two colors, such that no two vertices sharing the same edge contain the same color i,e in other words to check whether the graph is a bipartite graph. Now we are ready to state the lemma IV.5.

Lemma IV.5. *The problem of swapping of vertex operands (multiplication operands) in an undirected graph is polynomial time reducible to the problem of two-colorability of a graph.*

Proof. An instance of graph G is fed to the *problem b*, which returns the decision in polynomial time whether the input graph is two-colorable or not. If the answer is yes, then the graph is passed to a graph coloring algorithm that returns the resultant graph colored with two colors. Without loss of generality the two colors can be named as *color1* and *color2*. We define the set of vertices colored with *color1* as *swap set*, while the set of vertices colored with *color2* as *uninterrupted set*. Thus we have determined a solution for the instance of *problem a*. Hence proved. □

Now we give a closer look at the correctness of the polynomial reduction of *problem a* into *problem b*. As was mentioned in the above proof, the solution for the instance of the graph considered corresponding to *problem b* gives back the graph instance colored with two colors, based on the graph coloring algorithm. The vertices having *color1* form *set1*, while the vertices colored with *color2* form a *set2*. The vertices within *set1* do not contain any edge between them, similarly in *set2*, no two vertices are connected by an edge. For every edge in E, two vertices are colored with two distinct colors, which implies the two vertices belong to two different vertex sets. We can consider *set1* as the *swap set*, on the other hand the *set2* can be considered as the *uninterrupted set* required for the solution of *problem a*. The sets obtained from solution to *problem b* also satisfies the condition for the edge set that every edge should contain vertices belonging to the two different sets, so that for every edge the vertex belonging to the *swap set* should undergo operand swapping, while the other vertex from *uninterrupted set* should remain unaltered. That is why the solution obtained from *problem b* qualify as a solution for *problem a*.

Algorithm 3: Safe_sequence_converter() : Algorithm to determine safe operand ordering of multiplication pairs

```
Data: : Set S = { $m_i \mid i \in \{1, n\}$, where $n$ is the number of multiplications}
Result: : Set S' = { $m'_i \mid i \in \{1, n\}$, where $n$ is the number of multiplications}
begin
    // Step 1
    Create_Graph() ;
    Find_GraphComponents() ;
    // Step 2
    Find_Safeseq_$\hat{G}$() ;
end

Create_Graph(): ;
begin
    Initialize Graph G ;
    for $i \leftarrow 1$ to $n$ do
        $AddVertex(G, S[i])$ ;
        // create vertices of graph G
    end
    for $i \leftarrow S[0]$ to $S[n-1]$ do
        for $j \leftarrow S[0]$ to $S[n-1]$ do
            if $i \neq j$ and $share\_operand(S[i], S[j]) ==$ TRUE then
                $AddEdge(G, S[i], S[j])$ ;
                // create edges of graph G
            end
        end
    end
end

Find_GraphComponents(): // find Islands of the Graph
begin
    for $v \leftarrow 0$ to $G \rightarrow V - 1$ do
        $Visited[v] =$ FALSE
    end
    $seg\_count = 1$ ;
    for $v \leftarrow 0$ to $G \rightarrow V - 1$ do
        if $Visited[v] ==$ FALSE then
            $Island[seg\_count] = Clone\_Graph(G, v)$ ;
            // 1)clone the graph island containing vertex
                $v$
            // 2)set the visited vertices
            $Seg\_array[seg\_count].ele = v$ ;   // keep track of
            starting node of the island
            $seg\_count = seg\_count + 1$ ; // keep track of the
            number of islands formed
        end
    end
end

Find_Safeseq_$\hat{G}$(): // find safe sequences
begin
    for $i \leftarrow 0$ to $seg\_count - 1$ do
        $G_1 = Construct\_ComplementGraph(Island[i])$ ;
        $Remove\_redundant\_edges(G_1)$ ;
        // remove the edges not satisfying the edge
            property
        if $Colorable\_2(G_1) ==$ TRUE then
            $Color\_Graph(G_1, RED, BLACK)$ ;
        end
        $Swap\_Order(G_1)$ ;
        for $v \leftarrow 0$ to $(G_1 \rightarrow V - 1)$ do
            $S'.add(G_1 -> array[v].data)$ ;
        end
    end
end
```

C. Preventing scenario 2 by randomizing the Base point

Once the adversary fails to launch the *scenario* 1, she may exploit the possibility of *scenario* 2 of HCCA. As was discussed in section III-A, it is based on the observation that an addition operation involves two elliptic curve points, out of which one is always the base point. Let us consider two sets of field multiplications, S_1 as $\{m_i \mid m_i \in addition_i$ $\}$,while S_2 denoted as $\{m_j \mid m_j \in addition_j\ \}$. It can be directly observed that since there is a common elliptic curve point, passed as parameter to both the addition operations, there will exist a multiplication pair $(m_1,\ m_2)$, such that $m_1 \in addition_i$, $m_2 \in addition_j$ and $(m_1,\ m_2)$ shares one multiplication operand satisfying *property* $1a$. With this observation the attacker can launch HCCA on a single trace and identify all the addition operations, subsequently also the doubling operations.

We propose here a method based on a randomization scheme which aims at thwarting the *scenario* 2 of HCCA with minimal timing or area overhead. The technique is based on the idea of randomizing the base point at every execution of addition operation so that any two multiplications chosen from two addition operations become free from the operand sharing property. Based on standard projective coordinate system, the equivalence between two elliptic curve points can be defined as $(X_1,\ Y_1,\ Z_1) \sim (X_2,\ Y_2,\ Z_2)$ if $X_2 = \lambda X_1$, $Y_2 = \lambda Y_1$ and $Z_2 = \lambda Z_1$, where $\lambda \in F_p^*$. Any point $(X,\ Y,\ Z)$ can be randomized by using a random $\lambda \in F_p^*$ into the form R_p as $(\lambda X,\ \lambda Y,\ \lambda Z)$. We use this randomized base point as input to every addition operation. Our randomization method is based on execution of a random permutation for every scalar multiplication run. The set of numbers used in the permutation process can be represented by the set *perm* as $\{i \mid i \in [1, |A|]\}$, where $|A|$ denotes the maximum number of addition operation possible for a key $\in [1, order(E)]$, $order(E)$ is the order of the underlying elliptic curve. Every execution of the scalar multiplication algorithm involves one random permutation of the set *perm*. The λ value chosen for consecutive addition operations are chosen from the consecutive elements of the set *perm*. The addition operation once achieved by using the random point R_p requires derandomization such that the final result is same as that obtained from the fixed base point B_p. The derandomization involves three field multiplications of the form $(\lambda^k)^{-1} \times X_3$, $(\lambda^k)^{-1} \times Y_3$, $(\lambda^k)^{-1} \times Z_3$ per addition operation, where $(X_3,\ Y_3,\ Z_3)$ represents the intermediate output by using R_p (k is a constant). The computation involving λX, λY, λZ and $(\lambda^k)^{-1}$ for varied $\lambda \in$ *perm* can be stored before implementing the design. Note that this precomputation step is curve-specific, and is fixed for a base point. Consequently the precomputed values can be used during each addition operation based on the value of λ chosen. Thus the only extra cost involved in incorporating this countermeasure step includes three field multiplications per addition operation. It should be noted that only step 2 cannot prevent HCCA, since the *scenario* 1 of HCCA can be exploited within an addition and a doubling operation, although the base point randomization is present. Hence to prevent HCCA completely we require to integrate our countermeasure to the ECC design containing both the steps.

D. Overhead analysis of our countermeasure

In table II we give the overhead analysis of our countermeasure. As has been discussed in the previous section, *step* 1 involves a precomputation phase of constructing the safe sequence of an unified addition (doubling) operation, but requires no runtime overhead of time and area, once the safe form is obtained. *Step* 2 contains a precomputation step for randomization of the base point. We precompute randomized Base point values (λX, λY, λZ) for several values of $\lambda \in \{1, \ldots, L\}$, where $L = log(\#E(order of the EC))$. During runtime, each addition step requires a derandomization step at the end of the computation which involves three field multiplications.

TABLE II: Overhead analysis of our countermeasure

	countermeasure step 1	*countermeasure step* 2
Precomputation	Algorithm 1 with $\mathcal{O}(n^2)$ time and $\mathcal{O}(n^2)$ space, where n is number of verteces in the graph	computation of (λX, λY, λZ), where $\lambda \in \{1,\ldots,L\}$. This process requires $\mathcal{O}(L)$ time and $\mathcal{O}(L)$ space
Runtime	zero timing and area overhead	timing overhead of three field multiplications per addition, zero area overhead

V. Experimental Results

In earlier sections, we have established the basis of horizontal collision correlation attack along with the strategies to thwart this attack methodology. It is evident from [18] and our previous discussions that ECC scalar multiplication in both Edward curve and NIST curve is vulnerable to HCCA. Specifically, the Edwards curve implementation incorporating unified formula is extremely vulnerable to HCCA as there exists a pair of multiplication which shares both the operand during execution of point doubling. Hence an adversary is expected to observe extremely high similarities when he/she compares the power trace of aforementioned multiplications, sharing both the operands. The previous work on HCCA [18] uses Pearson's correlation coefficient as the measure of similarity between the power traces. The paper shows that power traces of multiplications having both of their operand shared exhibit high correlation value between them whereas power traces of multiplications having no operand shared show low correlation value between them. However, the experimental validation of HCCA has been provided through simulated side channel traces, but not on actual side channel (power or electromagnetic) traces which have been obtained from hardware. The scenario in actual hardware may differ significantly as the actual side channel traces (power or electromagnetic) are contaminated with system noise along with algorithmic noise. In this paper, we experimentally validate HCCA and our countermeasure against HCCA on actual FPGA. We have already introduced Euclidean distance as an alternative distinguisher for HCCA in previous discussions. In this section, we will first provide experimental validation of the fact that Euclidean distance serves as a better distinguisher than Pearson's correlation coefficient in HCCA context. Then we will provide experimental validation of the effectiveness of *step 1* of our proposed countermeasure in preventing HCCA.

(a) EM Trace of a Single Multiplication (b) Region of Interest

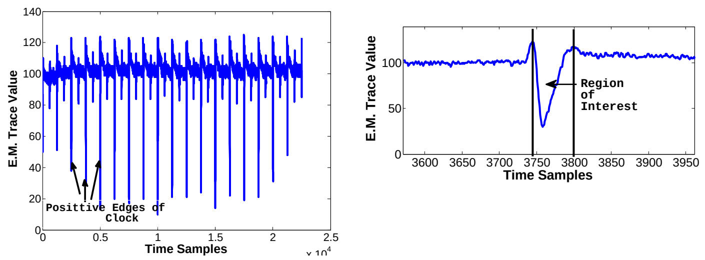


Fig. 4: Multiplication EM Trace

We have used SASEBO-GII as the hardware platform for evaluating HCCA and countermeasure *step 1*. All the algorithms are implemented on cryptographic FPGA of SASEBO-GII (XC5VLX50). As the first step of experimental validation, we have implemented a long integer multiplier of bit-width 255 bits. The implemented multiplier has similar architecture of [30] and requires 18 clock cycles to complete one single multiplication (without modular reduction). The architecture has an area overhead of 800 slices with 11 DSP blocks. Using this multiplier, we have acquired around 1000 EM traces where each trace consist of four multiplications involving 4 different operands. Let us term these operands as A, B, C, D. The multiplication operation within a single EM trace can be listed as 1) $A \times B$, 2) $C \times D$, 3) $B \times A$, 4) $A \times B$.

Now, for the HCCA to be successful on ECC scalar multiplication in Edward curve, an adversary should be able to understand that whether a pair of multiplication have both of their operand shared or not. More specifically adversary must have a distinguisher which should distinguish between pair of multiplications like $\{A \times B,\ A \times B\}$ and $\{A \times B,\ C \times D\}$. The EM trace of a single multiplication is shown in figure 4(a) where we have pointed the positive edge of the clock cycles. To calculate Euclidean distance (and Pearson's correlation coefficient) between the EM traces of a pair of multiplication, we have identified a *region of interest* around each positive edge of the clock cycle as shown in figure 4(b) as all switching activity of the underlying multiplier happens at the positive edge of the clock cycle. We calculate the Euclidean distance between two EM traces, denoting a pair of multiplication, for each *region of interest* by measuring the distance between each point of the *region of interest* for each clock cycle and calculating mean of that.

Thus Euclidean distance between two *region of interest* T_{1_R} and T_{2_R} of two EM traces T_1 and T_2 is calculated as follows:

$$d = \frac{1}{n} \sum_{i=0}^{i=n-1} |T_{1_{R,i}} - T_{2_{R,i}}| \quad (15)$$

where n is the length of the *region of interest*. So for each *region of interest* we calculate a single Euclidean distance value.

Similarly to compare Pearson's correlation coefficient with Euclidean distance as a distinguisher, we have computed Pear-

(a) Evaluation of HCCA on a Stand-alone Field Multiplier with Euclidean Distance

(b) Evaluation of *step* 1 on a Stand-alone Field Multiplier with Euclidean Distance

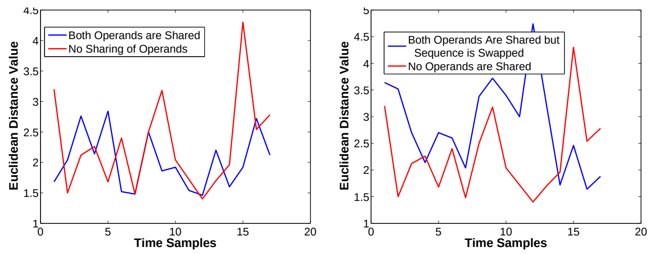


Fig. 5: Evaluation of HCCA and Countermeasure *step* 1 on a Stand-alone Field Multiplier with Euclidean Distance

(a) Evaluation of HCCA on a Stand-alone Field Multiplier with Correlation

(b) Evaluation of *step* 1 on a Stand-alone Field Multiplier with Correlation

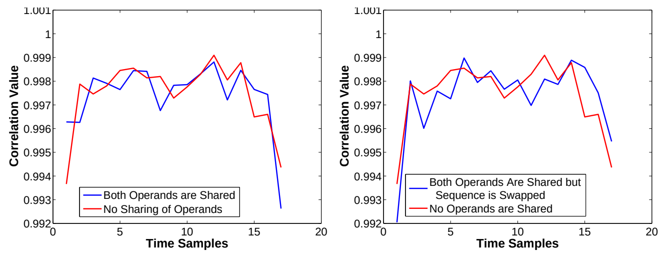


Fig. 6: Evaluation of HCCA and Countermeasure *step* 1 on a Stand-alone Field Multiplier with Correlation

(a) Evaluation of HCCA Through Surface Plot of Multiple EM traces of a Stand-alone Field Multiplier

(b) Evaluation of *step* 1 Through Surface Plot of Multiple EM traces of a Stand-alone Field Multiplier

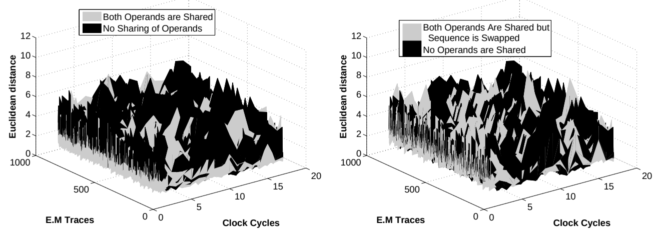


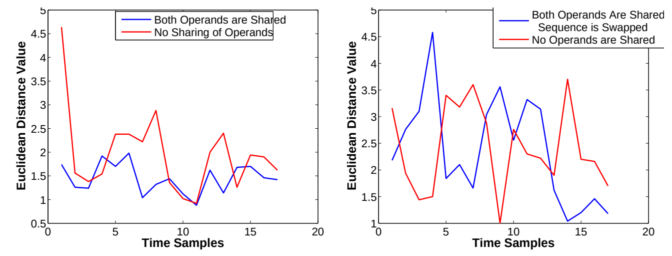
Fig. 7: Evaluation of HCCA and *step* 1 with multiple traces of Stand-alone Field Multiplier

son's correlation coefficient between each *region of interest* for every clock cycle between a pair of EM trace, where each EM trace denotes a multiplication operation. Thus in this case also for each *region of interest* we get a single correlation value.

We will first show our evaluation of a stand alone field multiplication operations. As we have already mentioned we have collected around 1000 EM traces for each multiplication operation indicating three different scenario: 1. A pair of multiplication having both of their operand shared 2. A pair of multiplication having none of their operand shared and 3. A pair of multiplication having both operand shared but operand sequence is swapped. Analysis of HCCA and countermeasure *step* 1 using Euclidean distance and Pearson correlation coefficient distinguisher are presented in figure 5 and 6. Figure 5(a) clearly shows that using Euclidean distinguisher, the adversary can easily distinguish whether between pair of multiplication having shared operand and pair of multiplication having no operand shared. However, this decision can not be taken by using correlation as shown in figure 6(a). The reason behind this disparity is as follows: correlation is a measure of similarity between two vectors. So if two regions of interest are of same nature but are separated by some distance, they will highly correlate with each other. Now, due to switching activity of the circuit, the circuit draws power from the supply and that is indicated on the EM trace by the dip from the reference level. A high switching activity will induce more dip from the reference level and low switching activity will induce less dip in the EM trace. Hence two *regions of interest* corresponding to two different Hamming weights will have different dip from the reference level, but will be of same nature. Hence they can be better distinguished by Euclidean distance rather than correlation.

The validation of the countermeasure *step* 1 is shown in figure 5(b). The figure clearly exhibit the effectiveness of the countermeasure as it is now not possible to distinguish between pair of multiplication having shared operand and pair of multiplication having no operand shared. We have repeated this experiment with multiple traces. A combined surface plot of all the traces for both HCCA and countermeasure *step* 1 is shown in figure 7. This result again proves the efficiency of Euclidean distance countermeasure for HCCA and the effectiveness of countermeasure *step* 1 in preventing HCCA.

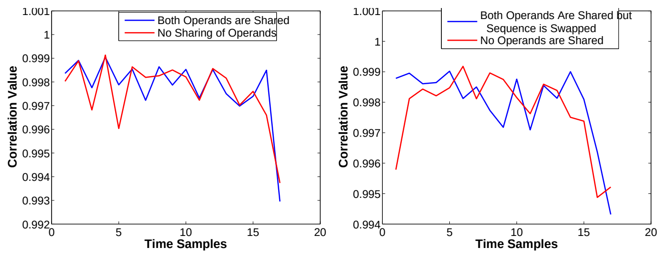
Next, we will show results on EM traces of actual ECC scalar multiplication in Edward curve. We have implemented *Curve1174* on SASEBO-GII evaluation board and has collected around 400 EM traces of scalar multiplication. As we have already mentioned in the previous sections that in Edward curve unified formula, point doubling involves a pair of field multiplication having both of their operands shared whereas point addition does not have any field multiplication which share both of the operand. Success of HCCA depends upon whether an adversary can distinguish between a pair of field multiplication having both of their operand shared and a pair of field multiplication having no common operand. If the adversary can do this, he can distinguish between point doubling and point addition operation which will directly give him the knowledge about secret scalar value. By using *step* 1 of the countermeasure, we aim to remove the threat of HCCA. The objective is to make the job of distinguishability between pair of multiplication having no operand shared and pair of multiplication having both of their operand shared difficult. Figure 8 shows that the result on Edward curve scalar multiplier is consistent with the observation that we made on figure 5. The *step* 1 of our countermeasure again act as an efficient protector of the implementation against HCCA. The ineffectiveness of correlation is shown in figure 9. Finally result through surface plot on multiple traces is shown in figure 10 which again supports the effectiveness of countermeasure *step* 1.



(a) Evaluation of HCCA on Edward Curve Scalar Multiplier with Euclidean Distance

(b) Evaluation of *step* 1 on Edward Curve Scalar Multiplier with Euclidean Distance

Fig. 8: Evaluation of HCCA and *step* 1 on Edward Curve Scalar Multiplier with Euclidean Distance



(a) Evaluation of HCCA on Edward Curve Scalar Multiplier with Correlation

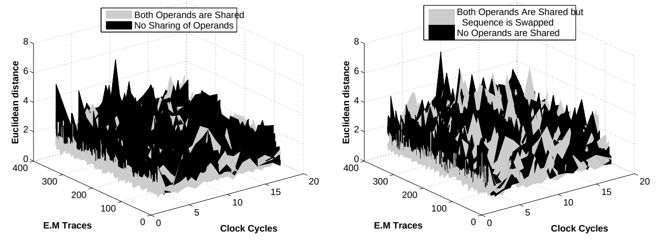
(b) Evaluation of *step* 1 on Edward Curve Scalar Multiplier with Correlation

Fig. 9: Evaluation of HCCA and *step* 1 on Edward Curve Scalar Multiplier with Correlation

VI. Resistance against related Horizontal attacks

A. *Resistance against [16], [17]*

The attack demonstrated in [16] is applicable in case of regular algorithms, where both doubling and addition operation are computed during each iteration. Two registers are considered to store the intermediate results of each iteration. The values of the two registers observed over consecutive iterations are dependent on the key, hence lead to retrieval of the secret key. This attack cannot be directly applied to non-regular algorithms. In [17] an incremental key retrieval process has been proposed for an ECC algorithm, where template trace is being created for the i-th iteration based on the already retrieved portion of the secret and the guessed key bit for the current iteration. However in our implementation, countermeasure *step* 2 uses a different random value for each addition (doubling) operation within a scalar multiplication, thus data dependency based on the previously determined key bit value cannot be exploited.



(a) Evaluation of HCCA Through Surface Plot of Multiple EM traces of Edward Curve Scalar Multiplier

(b) Evaluation of *step* 1 Through Surface Plot of Multiple EM traces of Edward Curve Scalar Multiplier

Fig. 10: Evaluation of HCCA and *step* 1 with multiple traces of Edward Curve Scalar Multiplier

B. *Resistance against [25]*

In [25] an attack on RSA is demonstrated, where long integer multiplication computation has been exploited. In case of a squaring operation, the long integer multiplication on operands of length l words involves $(l^2-l)/2$ potential collision pairs of single precision multiplications, which makes the long integer operation vulnerable. However in case of field multiplications of ECC, the number of collision pairs present are less, because of lower bit length of field multiplier operands, hence lower number of collision pairs on the same architecture model. Also the paper does not present any practical results of this collision based attacks. Applicability of [25] in ECC is yet to be exploited.

VII. A note on other countermeasures

In [18] authors have made a discussion on possibilities of potential countermeasures inside the field multiplication operation. Note that a long integer multiplication according to algorithm 2 may be represented as two-dimensional matrix. The countermeasures mentioned are mainly based on randomization and shuffling of the rows and the columns of the matrix thus obtained. The countermeasures proposed are: 1) *shuffling rows and columns* 2) *shuffling and blinding* 3) *global shuffling*. Out of the techniques mentioned, *shuffling rows and columns* scheme uses a random permutation of the rows or the columns during each long integer multiplication. It adds a $t!$ search factor to HCCA, which can be broken for smaller values of t. Here t is the underlying architecture word length, thus can take a maximum value of 64 bit. The *shuffling and blinding* method prevents HCCA but is prone to other attacks like zero-value attack [31]. The *global shuffling* technique presented in [32] utilizes the idea of shuffling simultaneously across rows and columns of the long integer multiplication partial product matrix, thus increasing the search factor to $t^2!$. This method is resistant against HCCA due to the sufficiently large search space introduced. However it involves incorporating the randomization technique to every field multiplication which includes generation of a random permutation, and execution of an additional loop to take care of the carry propagation of the partial products. The execution of the additional loop and generation of random permutation increases clock cycle requirement of the long integer multiplication which in turn increases the timing overhead of the design. On the contrary it should be noted from table II, our countermeasure *step* 1 requires zero timing and area overhead at runtime, and the *step* 2 bears minimal overhead due to randomization. In our implementation randomization is applied for each addition (doubling) operation during the scalar multiplication, thus involves far less overhead than applying randomization for every long integer multiplication as suggested in the above countermeasure in [32].

VIII. Conclusion

We have shown how the property of asymmetric leakage of field multipliers can be utilized to construct a low-cost countermeasure which is able to defeat the powerful HCCA. We show how a unified addition (doubling) formula can be converted into a safe sequence where, the information leakage from sharing of operands among field multipliers have been hidden. The process of conversion to the desired safe sequence is achieved through our proposed Algorithm 1, once the sequence have been determined through our algorithm there is no runtime overhead requirement for the step 1 of our countermeasure. We have incorporated a randomization technique at the level of each addition operation to tackle another possibility of HCCA vulnerability. We have shown how our countermeasure methodology is able to resist HCCA altogether. We have validated HCCA and our proposed countermeasure scheme on a SAKURA-G platform. Our analysis has been done considering the left-to-right scalar multiplication algorithm, we intend to investigate the cases of various right-to-left scalar multiplication algorithms. Additionally we want to explore applicability of our countermeasure in case of NIST curve atomicity formulae which are still HCCA vulnerable.

References

[1] NIST. The Case for Elliptic Curve Cryptography. 2009.

[2] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 388–397, 1999.

[3] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 444–461, 2014.

[4] Jean-Sébastien Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, pages 292–302, 1999.

[5] Benoît Chevallier-Mames, Mathieu Ciet, and Marc Joye. Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity. *IEEE Trans. Computers*, 53(6):760–768, 2004.

[6] Patrick Longa. Accelerating the scalar multiplication on elliptic curve cryptosystems over prime fields. *IACR Cryptology ePrint Archive*, 2008:100, 2008.

[7] Eric Brier and Marc Joye. Weierstraß elliptic curves and side-channel attacks. In *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002, Paris, France, February 12-14, 2002, Proceedings*.

[8] Harold M. Edwards. A normal form for elliptic curves, 2007.

[9] Daniel J. Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. Twisted edwards curves. In *Progress in Cryptology - AFRICACRYPT 2008, First International Conference on Cryptology in Africa, Casablanca, Morocco, June 11-14, 2008. Proceedings*, pages 389–405, 2008.

[10] Daniel J. Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. Twisted edwards curves. In *Progress in Cryptology - AFRICACRYPT 2008, First International Conference on Cryptology in Africa, Casablanca, Morocco, June 11-14, 2008. Proceedings*, pages 389–405, 2008.

[11] Kwang Ho Kim, Chol Ok Lee, and Christophe Nègre. Binary edwards curves revisited. In *Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings*, pages 393–408, 2014.

[12] Hüseyin Hisil, Kenneth Koon-Ho Wong, Gary Carter, and Ed Dawson. Twisted edwards curves revisited. In *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings*, pages 326–343, 2008.

[13] Daniel J. Bernstein and Tanja Lange. Safecurves: choosing safe curves for elliptic-curve cryptography, http://safecurves.cr.yp.to/. 2014.

[14] Colin D. Walter. Sliding windows succumbs to big mac attack. In *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, number Generators, pages 286–299, 2001.

[15] Benoit Feix, Mylène Roussellet, and Alexandre Venelli. Side-channel analysis on blinded regular scalar multiplications. In *Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings*, pages 3–20, 2014.

[16] Neil Hanley, HeeSeok Kim, and Michael Tunstall. Exploiting collisions in addition chain-based exponentiation algorithms using a single trace. In *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings*, pages 431–448, 2015.

[17] Lejla Batina, Lukasz Chmielewski, Louiza Papachristodoulou, Peter Schwabe, and Michael Tunstall. Online template attacks. In *Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings*, pages 21–36, 2014.

[18] Aurélie Bauer, Éliane Jaulmes, Emmanuel Prouff, Jean-René Reinhard, and Justine Wild. Horizontal collision correlation attack on elliptic curves - - extended version -. *Cryptography and Communications*.

[19] Kai Schramm, Thomas J. Wollinger, and Christof Paar. A new class of collision attacks and its application to DES. In *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003, Revised Papers*, pages 206–222, 2003.

[20] Louis Goubin. A refined power-analysis attack on elliptic curve cryptosystems. In *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, pages 199–210, 2003.

[21] Kouichi Itoh, Tetsuya Izu, and Masahiko Takenaka. Address-bit differential power analysis of cryptographic schemes OK-ECDH and OK-ECDSA. In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, pages 129–143, 2002.

[22] Kouichi Itoh, Tetsuya Izu, and Masahiko Takenaka. A practical countermeasure against address-bit differential power analysis. In *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, pages 382–396, 2003.

[23] Junfeng Fan and Ingrid Verbauwhede. An updated survey on secure ECC implementations: Attacks, countermeasures and cost. In *Cryptography and Security: From Theory to Applications - Essays Dedicated to Jean-Jacques Quisquater on the Occasion of His 65th Birthday*, pages 265–282, 2012.

[24] Frédéric Amiel, Benoit Feix, Michael Tunstall, Claire Whelan, and William P. Marnane. Distinguishing multiplications from squaring operations. In *Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers*, pages 346–360, 2008.

[25] Christophe Clavier, Benoit Feix, Georges Gagnerot, Christophe Giraud, Mylène Roussellet, and Vincent Verneuil. ROSETTA for single trace analysis. In *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*, pages 140–155, 2012.

[26] Marc Joye and Sung-Ming Yen. The montgomery powering ladder. In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, pages 291–302, 2002.

[27] Marc Joye. Highly regular right-to-left algorithms for scalar multiplication. In *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, pages 135–147, 2007.

[28] Daniel J. Bernstein and Tanja Lange. Faster addition and doubling on elliptic curves. In *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, pages 29–50, 2007.

[29] Takeshi Sugawara, Daisuke Suzuki, and Minoru Saeki. Two operands of multipliers in side-channel attack. *IACR Cryptology ePrint Archive*, 2015:291, 2015.

[30] Debapriya Basu Roy, Debdeep Mukhopadhyay, Masami Izumi, and Junko Takahashi. Tile before multiplication: An efficient strategy to optimize DSP multiplier for accelerating prime field ECC for NIST curves. In *The 51st Annual Design Automation Conference 2014, DAC '14, San Francisco, CA, USA, June 1-5, 2014*, pages 177:1–177:6, 2014.

[31] Jovan Dj. Golic and Christophe Tymen. Multiplicative masking and power analysis of AES. In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, pages 198–212, 2002.

[32] Aurélie Bauer, Éliane Jaulmes, Emmanuel Prouff, and Justine Wild. Horizontal and vertical side-channel attacks against secure RSA implementations. In *Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, San Francisco,CA, USA, February 25-March 1, 2013. Proceedings*, pages 1–17, 2013.