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

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 SASEBO 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.

Keywords: ECC, HCCA, countermeasure, asymmetric leakage, field multiplications

1 Introduction

Elliptic curve cryptosystems are emerging as a primary choice for securing light-weight embedded devices as it incorporates more security per key bit with respect 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 imposes 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 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 (Algorithm 2 in Appendix A), associated with two different formulae for addition and doubling, using a Weierstrass form of elliptic curve (equation 9 in Appendix A), 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 has been introduced in [5], [6] which transforms 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. In [7] a unified addition formula is designed for a Weierstrass form of elliptic curve, which involves the same formula for 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. Recently 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). The reader may refer to [13] for 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) 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 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, recently a powerful single-trace attack named horizontal collision correlation analysis (HCCA) is introduced in [14] 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 [15], adrress-bit differential attack [16] with effective randomization schemes suggested in [17], [18], 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. As can be noted in [14] HCCA can be mounted in two phases: 1st phase aims at distinguishing between point doubling and point addition operation, based on this property of field multiplications related to sharing of multiplier operands. In the situation when this property does not hold, 2nd phase of HCCA is launched which intends to correlate two point additions, based on the fact that it always share common base point. It is quite evident that preventing HCCA is extremely essential, since otherwise the implementation becomes exposed to be broken even with a single trace.
Our contribution Our main contribution in this paper is to design a cost-effective yet adequate countermeasure that resists 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 correlation between two multiplications sharing one/ two common operand is dependent on the order of operands passed to the individual multiplications.
- We propose a countermeasure that can be applied to the existing unified algorithms to defeat the first phase of HCCA. Our countermeasure involves no randomization, instead it converts the unified algorithm into a safer form, such that the relation between multiplications based on property of operand sharing cannot be exploited. Our countermeasure requires a single instance of precomputation phase for designing the safe algorithm. As a result, our countermeasure imposes zero cost of timing or area overhead on the implementation.
- We also design a second countermeasure 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 countermeasures becomes resistant against HCCA.
- Finally we provide practical results of HCCA along with our countermeasure on an FPGA board.

Paper organization The organization of the paper is as follows. In Section 2, we give a review on HCCA. Section 3 provides a theoretical validation of our countermeasure idea, followed by description of our proposed countermeasures. Section 4 includes actual experimental results of HCCA. Finally we conclude in Section 5. Other relevant details have been provided in the Appendix.

2 Preliminaries

In this section we introduce the horizontal collision correlation analysis. Background on elliptic curve cryptography with an emphasis on unified addition rule has been given in the Appendix A section. We note here that for efficiency reason, often the ECC curve equation is homogenized into projective coordinates. The entire computation of elliptic scalar multiplication is performed in projective coordinate system and finally the result is transformed to affine coordinate. Throughout the paper, the ECC algorithms considered will be in the standard projective coordinate system, where an ECC point is presented by the form $P = (X, Y, Z)$. Also the underlying F_p is considered to be a prime field with underlying prime p, as hardware implementations of ECC on prime fields give best efficiency results and has been recommended vastly in the literature [19].

2.1 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* [14]. 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 ($A \times B$, $C \times D$) 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 . . . 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 (FA in Figure 1(a)) and field multiplications (FM in Figure 1a). 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 if we consider the correlation between the power traces of two concerned multiplication pairs, 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 [14]. 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.

When the addition and doubling operations do not contain two such favorable multiplication pairs which will lead to distinguishable values of correlation coefficient, the *scenario* 1 of HCCA is unable to retrieve any information regarding the secret key. We investigate the various possibilities when the above situation may arise. It may happen that none of the addition or doubling operations

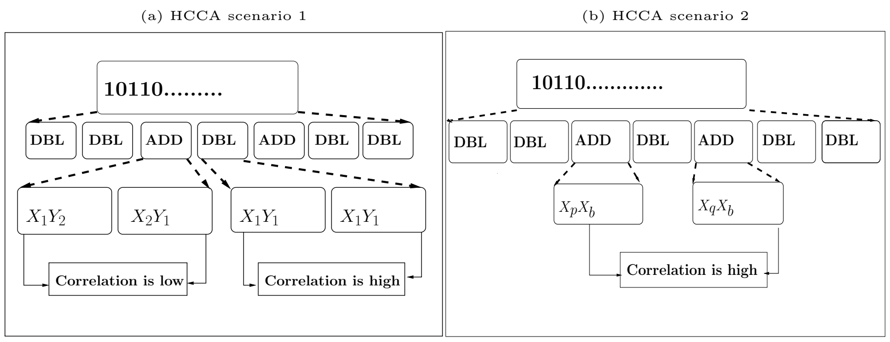


Fig. 1: Horizontal Collision Correlation Analysis (HCCA)

has any such multiplication pair that help in the distinction. Or, it is also possible that both of the addition and doubling possess one such pair. An illustrative example of a situation where the *scenario* 1 of HCCA fails has been provided in Table 3 (in Appendix E).

Alternatively adversary proceeds to mount second version of HCCA or *scenario* 2. From a standard right-to-left double-and-add algorithm (Algorithm 2 in Appendix A) 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 2). 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 2 is run with an underlying Edward curve equation (shown in Table 1 in Appendix A) 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. The details on the intermediate steps involved in case of addition and doubling for the Edward curve considered above has been provided in Appendix E to illustrate the *scenario* 2 of HCCA.

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.

3 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). Our proposed *countermeasure* 1 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 our *countermeasure* 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 textitcountermeasure 1 requires zero overhead of resources in case of the Edward curves unified formula as well as Brier-Joye unified formula. The *countermeasure* 1 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 [20] in case of multipliers, however authors of [20] 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.

3.1 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 5. 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 $R[2t]$. The algorithm is run $\mathcal{O}(t^2)$ times.

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 background 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 5, 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 $< H(a_i b_j) >$. 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

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

It is evident from Algorithm 5 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 $< l_{(a_0,b_0)}$, $l_{(a_0,b_1)}$,. . ., $l_{(a_0,b_{t-1})}$,. . ., $l_{(a_{t-1},b_{t-1})} >$. 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 in the variation of the correlation coefficient value between two multiplications.

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

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

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)

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

where we denote $\sum_{i=0,j=0}^{t-1} H(a_ib_j)H(c_ib_j)$ 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)

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

where $\sum_{i=0,j=0}^{t-1} H(a_ib_j)H(b_ic_j)$ 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).

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

where $\sum_{i=0,j=0}^{t-1} H(a_ib_j)H(c_id_j)$ 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 1. *The probability of collision of a pair (mn, pn) is more than the probability of collision of the pair (mn, pq).*

Lemma 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).*

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

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

The proof of the above lemmas have been provided in Appendix B. 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

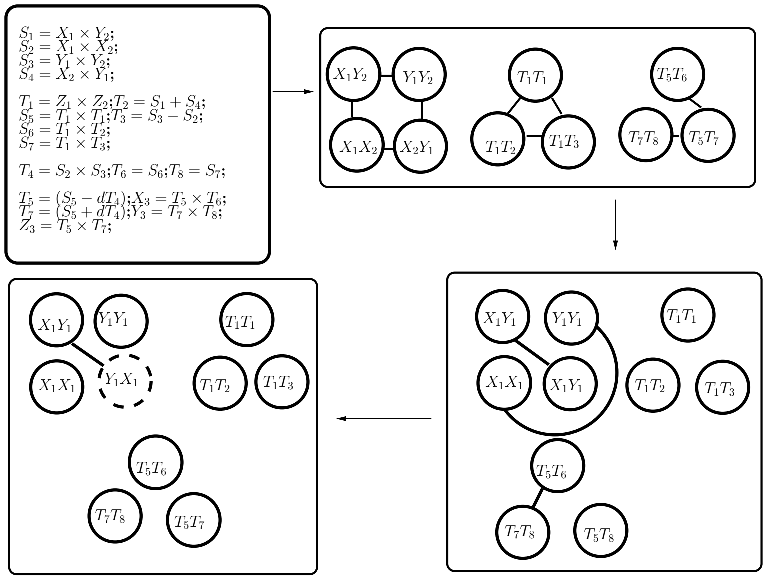
$$Covariance(H(AB), H(CB)) \neq Covariance(H(AB), H(BC)).$$

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 our *countermeasure* 1 which will be explained in the following subsection.

3.2 Countermeasure 1

Countermeasure 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 2.1, an ECC implementation becomes vulnerable to *scenario* 1 of HCCA if only one of the addition or doubling operation satisfies *property* 3. Our *countermeasure* 1 alters 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 countermeasure does not incorporate any randomization or any extra operation. Therefore the cost of this countermeasure 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 1 which takes care of the transformation process of *countermeasure* 1. We proceed to portray our transformation mechanism through an illustration, which will be followed by a description of our designed Algorithm 1.


$S_1 = X_1 \times Y_2;$
$S_2 = X_1 \times X_2;$
$S_3 = Y_1 \times Y_2;$
$S_4 = X_2 \times Y_1;$

$T_1 = Z_1 \times Z_2; T_2 = S_1 + S_4;$
$S_5 = T_1 \times T_1; T_3 = S_3 - S_2;$
$S_6 = T_1 \times T_2;$
$S_7 = T_1 \times T_3;$

$T_4 = S_2 \times S_3; T_6 = S_6; T_8 = S_7;$

$T_5 = (S_5 - dT_4); X_3 = T_5 \times T_6;$
$T_7 = (S_5 + dT_4); Y_3 = T_7 \times T_8;$
$Z_3 = T_5 \times T_7;$

Fig. 2: Safe sequence transformation of Edward curve formula

We have considered the Edward curve unified formula shown in Table 1 (in Appendix A) 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. 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 doubling 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* 1*b*. 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 1, 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 in Appendix G). So we suggest to swap the order of operands of the second multiplication. From lemma 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 Table 1 (in Appendix A) into a secure structure. The transformation steps corresponding to the Brier-Joye formula is portrayed in Figure 4 (in Appendix F).

Lemma 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.*

3.3 Countermeasure Scenario 2

Once the adversary fails to launch the *scenario* 1, she may exploit the possibility of *scenario* 2 of HCCA. As was discussed in section 2.1, 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* 1*a*.

Algorithm 1: Safe_sequence_converter() : Algorithm to determine safe operand ordering of multiplication pairs

```
Data: : Set S = { $m_i$ | $i \in \{1, n\}$, where $n$ is the number of multiplications}
Result: : Set S' = { $m'_i$ | $i \in \{1, n\}$, 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
```

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 countermeasure design 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 derandmization 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 (for e.g $k = 8$ for the Edward curve equation in Table 1). 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 includes three field multiplications per addition operation. The *countermeasure* 2 is presented in Algorithm 4 (in Appendix D).

4 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 [14] and our previous discussions that ECC scalar multiplication in both Edward curve and NIST curve is vulnerable to HCCA. Specifically for the Edwards curve implementation incorporating unified formula is extremely vulnerable to HCCA as there exist a pair of multiplication which share both the operand during execution of point doubling. Hence an adversary is expected to get a very high correlation when he/she compares the aforementioned multiplications, sharing both the operands. In all the previous work focussing on HCCA, the authors have provided experimental validation of HCCA through simulation results. However, the scenario in actual hardware may differ significantly as the power traces are contaminated with system noise along with algorithmic noise. In this paper, we experimentally validate HCCA and our countermeasure against HCCA on actual FPGA. The hardware platform is SASEBO-GII [21], where we have implemented a hardware for long integer multiplication. The implemented 256-bit long integer multiplier requires 605 LUTs and 602 registers and requires 260 clock cycles to execute one multiplication.

The experimentation steps are as follows:

- From software implementation of Edward curve scalar multiplication, we find out two operands which are shared by the multiplication pairs during point doubling execution. Let these operands be denoted as A and B. We also select four operands which are input to the field multiplication during point addition. It must be noted that in case of Edward curve, point addition does not have any multiplication pair which shares both the operands. We denote these operands as C, D, E and F.
- We collect 100 power traces for the following set of multiplications
  1. $A \times B$ -denoting the first multiplication of the multiplication pair of point doubling operation. We denote this set of power traces as $Double - Set1$.
  2. $A \times B$ -denoting the second multiplication of the multiplication pair of point doubling operation. We denote this set of power traces as $Double - Set2$.

3. $C \times D$ -denoting the first multiplication of the corresponding multiplication pair of point addition operation.We denote this set of power traces as $Addition - Set1$.
4. $E \times F$ -denoting the second multiplication of the corresponding multiplication pair of point addition operation. We denote this set of power traces as $Addition - Set2$.
5. $B \times A$ -denoting the second multiplication of the multiplication pair of point doubling operation with the operands position swapped. We denote this set of power traces as $DoubleReverse - Set1$.

– Finally we compute mean of each set of power traces and calculate following correlation values
  1. $Corr - DD$: correlation between the mean of $Double - Set1$ and $Double - Set2$
  2. $Corr - AA$: correlation between the mean of $Addition - Set1$ and $Addition - Set2$
  3. $Corr - DR$: correlation between the mean of $Double - Set1$ and $DoubleReverse - Set1$

Figure 3(a) shows the corresponding plot of correlation values at each clock cycle. To compute the correlation, we have extracted 100 points from each clock cycle of the power traces where register update occurs and have computed the correlation value between the extracted power traces. Figure 3(a) shows the correlation plot of 200 such clock cycles. From the result it is evident that multiplication pair having both the operands shared is easily distinguishable from the multiplication pair having no common operands, which validates the HCCA premises on the actual hardware. Moreover when we change the operands sequence, we can no longer distinguish between multiplication pair having shared operands and multiplication pair without any shared operands, which in turn validates the effectiveness of the proposed countermeasure. We have also validated our countermeasure on actual Edward curve scalar multiplier implemented on SASEBO G-II. The corresponding result is shown in Figure 3(b), where we show how changing the multiplier input operand sequence affects the leakage of the underlying field multiplier.

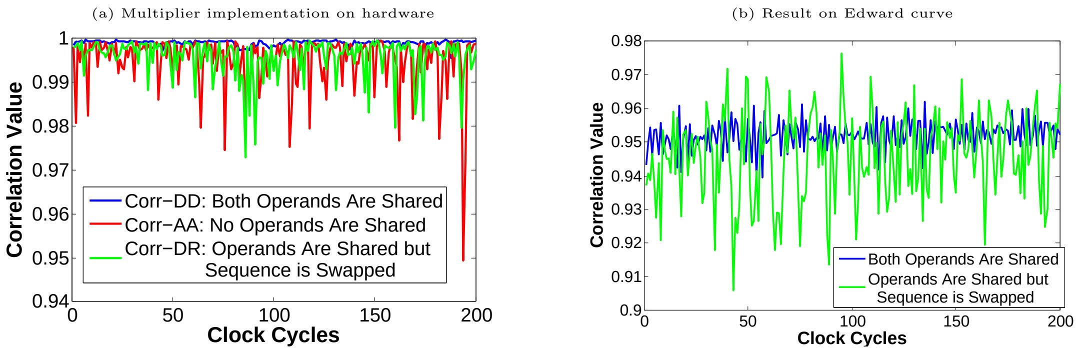


Fig. 3: HCCA and countermeasure validation on SASEBO-GII

5 Conclusion

We have shown how the property of asymmetric leakage of field multipliers can be utilized to construct a simple countermeasure which is able to defeat the powerful HCCA. Since our countermeasure 1 requires one time effort of designing the safe sequence before implementing the ECC algorithm design, which can be taken care of with the help of a compiler, so it is basically free of cost. Our proposed countermeasure 2 also requires minimal cost of precomputation, and additional field multiplications. We argue such overhead is extremely low as compared to another popular countermeasure, the Montgomery Ladder [22] which bears the overhead of dummy point addition operations. We have shown a comparison of our countermeasure effectiveness with respect to the alternatives presented in [14] in appendix C. This motivates us in exploiting the possibility of our countermeasure applicability in wider range of elliptic curve algorithms.

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. Aurélie Bauer, Éliane Jaulmes, Emmanuel Prouff, Jean-René Reinhard, and Justine Wild. Horizontal collision correlation attack on elliptic curves - - extended version -. *Cryptography and Communications*.
15. 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.
16. 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.
17. 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.
18. 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.
19. Tim Güneysu and Christof Paar. Ultra high performance ECC over NIST primes on commercial fpgas. In *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, pages 62–78, 2008.
20. Takeshi Sugawara, Daisuke Suzuki, and Minoru Saeki. Two operands of multipliers in side-channel attack. *IACR Cryptology ePrint Archive*, 2015:291, 2015.

21. Side-channel attack standard evaluation board sasebo-gii specification. 2009.
22. 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.
23. Victor S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*.
24. Neal Koblitz. A family of jacobians suitable for discrete log cryptosystems. In *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, pages 94–99, 1988.
25. Daniel J. Bernstein and Tanja Lange. Explicit database formula, https://hyperelliptic.org/efd/g1p/auto-edwards-projective.html. 2007.
26. 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.
27. 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.
28. 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.

A Elliptic curve cryptography

An elliptic curve E defined over a field $F(P)$, where the underlying prime is P and the characteristic of the field $F(P) \neq 2, 3$, is of the following form, called as short Weierstrass form:

$$y^2 = x^3 + ax + b \tag{9}$$

where a, b are curve constants which $\in F(P)$, and the discriminant $\Delta = (4a^2 + 27b^2) \neq 0$. The group of points (x, y) satisfying equation 9 together with a special point *point at infinity* O form an abelian group G. The underlying group operation o is defined by elliptic curve addition operation '+'. It should be noted that the elliptic curve addition rule as proposed in [23], [24] for an underlying Weierstrass form of elliptic curve based on equation 9 is not complete. It involves two separate formulas for elliptic curve point doubling and point addition operations. On the contrary, if the underlying addition formula involves a unified addition rule it can applied for both addition and doubling computation.

Algorithm 2: Left-to-Right Scalar Multiplication Algorithm

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

Designing unified formulas for ECC is an interesting area of research in the context of elliptic curve cryptography, as well as side channel security. This is because, firstly it provides a single addition formula which completely defines the group addition rule, secondly use of same formula for both addition and doubling inherently incorporates security against simple power analysis. In

Table 1: Unified formula with addition rule

Edward curve formula	Edward curve addition rule [25]
$(X^2+Y^2)Z^2 = c^2(Z^2+dX^2Y^2)$ $c \neq 0$, $\sqrt{d} \notin F_p$	$X_3 = Z_1Z_2(X_1Y_2+X_2Y_1)((Z_1Z_2)^2 - dX_1X_2Y_1Y_2)$ $Y_3 = Z_1Z_2(Y_1Y_2 - X_1X_2)((Z_1Z_2)^2 + dX_1X_2Y_1Y_2)$ $Z_3 = ((Z_1Z_2)^2 - dX_1X_2Y_1Y_2)((Z_1Z_2)^2 + dX_1X_2Y_1Y_2)$
Elliptic curve formula (in projective coordinates)	Brier-Joye unified addition rule [7]
$Y^2Z = X^3 + aXZ^2 + bZ^2$	$X_3 = 2FW$ $Y_3 = R(G-2W) - L^2$ $Z_3 = 2F^3$ where, $U_1 = X_1Z_2$, $U_2 = X_2Z_1$, $S_1 = Y_1Z_2$, $S_2 = Y_2Z_1$, $Z = Z_1Z_2$, $T = S_1 + S_2$, $M = S_1 + S_2$, $R = T^2 - U_1U_2 + aZ^2$, $F = ZM$, $L = MF$, $G = TL$, $W = R^2 - G$.

Table 1, we present unified formulas along with the addition rules for Edward curve [26], as well as the Brier-Joye addition formula [7].

Given a scalar k, scalar multiplication of an ECC point P i.e kP is obtained through a sequence of doubling and addition operations. The hardness of ECC cryptosystems is based on the **elliptic curve discrete logarithm problem** (ECDLP), which states that given a point P, and its scalar product kP, it is computationally difficult to retrieve the secret key k. A scalar multiplication algorithm is shown in Algorithm 2 which works as follows. It iteratively processes the secret key k from right to left, computes a doubling, and a conditional addition operation based on the ith key bit k_i. Note that the addition routine ADD in Algorithm 2 takes two ECC points as input, one of which is always the base point (B_p) of the curve. While the doubling routine DBL in Algorithm 2 takes single ECC point as input, which changes at every iteration. Also it should be noted that the scalar multiplication algorithm with an underlying unified addition formula will be similar to Algorithm 2 only with the difference that, it requires a single function for both doubling and addition. In case of doubling the same point is passed as both the input parameters. Similarly, here also addition involves a fixed base point as one of the input parameters, while for doubling the input point changes every iteration.

Algorithm 3: Left-to-Right Scalar Multiplication Algorithm for unified addition formula

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

B Proofs of the stated lemmas

The proofs of lemma 1-4 which support the theoretical validation of our countermeasure has been provided below. Additionally we include lemma 5 which guides us in the formation of the safe sequence form for *countermeasure* 1.

B.1 Proof of lemma 1

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.

B.2 Proof of lemma 2

Proof. The vector composed from leakage information of LIM(A, B) can be expanded as $< H(a_0, b_0), H(a_0, b_1), \ldots, H(a_0, b_{t-1}), \ldots, H(a_{t-1}, b_{t-1}) >$. While the vector obtained from leakage information of LIM(B, A) is represented as $< H(b_0, a_0), H(b_0, a_1), \ldots, H(b_0, a_{t-1}), \ldots, H(b_{t-1}, a_{t-1}) >$. 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.

B.3 Proof of lemma 3

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

$$Covariance(H(AB), H(CB)) = \frac{\alpha}{t} - mean(AB)mean(CB) \tag{10}$$

$$\begin{aligned} Covariance(H(AB), H(BC)) &= \frac{\beta}{t} - mean(AB)mean(BC) \\ &= \frac{\beta}{t} - mean(AB)mean(CB) \end{aligned} \tag{11}$$

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

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

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

$$\begin{aligned} \alpha =& (H(a_0b_0)H(a_0b_0) + H(a_0b_1)(a_0b_1) + \ldots + H(a_0b_{t-1})H(a_0b_{t-1}) \\ & + H(a_1b_0)H(a_1b_0) + \ldots + H(a_{t-1}b_{t-1})H(a_{t-1}b_{t-1})) \\ =& (H(a_0b_0)^2 + H(a_0b_1)^2 + \ldots + H(a_0b_{t-1})^2 \\ & + H(a_1b_0)^2 + \ldots + H(a_{t-1}b_{t-1})^2) \end{aligned} \tag{12}$$

While β can be reduced as

$$\begin{aligned}\beta =&(H(a_0b_0)H(b_0a_0) + H(a_0b_1)H(b_0a_1) + \ldots + H(a_0b_{t-1})H(b_0a_{t-1})\\&+ H(a_1b_0)H(b_1a_0) + \ldots + H(a_{t-1}b_{t-1})H(b_{t-1}a_{t-1}))\\=&(H(a_0b_0)^2 + H(a_0b_1)(b_0a_1) + \ldots + H(a_0b_{t-1})H(b_0a_{t-1})\\&+ H(a_1b_0)h(b_1a_0) + \ldots + H(a_{t-1}b_{t-1})^2).\end{aligned} \tag{13}$$

From equations 12 and 13, we can observe that $\alpha \neq \beta$. As a result when $C = A$, we can conclude similarly that

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

B.4 Proof of lemma 4

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

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

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

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

Since from Lemma 3,

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

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

B.5 Lemma 5, discussion and proof

Before proceeding to state the following lemma 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 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 also 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 $\{v$ | operands of v should be swapped$\}$ 2) V_2 or *uninterrupted set*, can be presented as $\{v$ | 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.

Lemma 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.*

C A note on other countermeasures

In [14] authors have made a discussion on possibilities of potential countermeasures inside the field multiplication operation. The countermeasures mentioned are mainly based on randomization and shuffling schemes. The countermeasures proposed are: 1) *shuffling rows and columns* 2) *shuffling and blinding 3 global shuffling.* Out of the techniques mentioned, *operand blinding* fails to counter HCCA, *shuffling rows and columns* scheme adds a $t!$ search factor to HCCA, which can be broken for smaller values of t. The *shuffling and blinding* method prevents HCCA but is prone to other attacks like zero-value attack [27]. The *global shuffling* technique presented in [28] 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.

D Countermeasure 2 algorithm

The *countermeasure* 2 that was introduced in Section 3 has been given in Algorithm 4. Note that the precomputation step is curve-specific, the one mentioned in Algorithm 4 includes precomputation for Edward curve formula (in Appendix A).

Algorithm 4: Countermeasure 2

```
Data: secret key $k = \{k_m, \dots, k_2, k_1\}$, Base point $P$
Result: scalar product $kP$
// Precomputation Phase
for $r \leftarrow 1$ to $|A|$ do
    $store[r].point \rightarrow x = X_r$
    $store[r].point \rightarrow y = Y_r$
    $store[r].point \rightarrow z = Z_r$
    $store[r].inv = inv\_lambda$
end
// generate a random permutation $RP$ of the set $perm = \{i \mid i \in [1, |A|]\}$
$index = 1$
// scalar multiplication
for $i = m - 1$ to $1$ do
    $Doubling(P)$
    if $k_i == 1$ then
        $Addition(P,\ store[RP[index]].point)$
        $index = index + 1$
    end
end
$Addition(P_1,\ P_2)$
// Addition steps
// derandmization steps
$result.x = (store[RP[index]].inv) * result.x$
$result.y = (store[RP[index]].inv) * result.y$
$result.z = (store[RP[index]].inv) * result.z$
copy_point($P_1, result$)
```

E Illustration of HCCA

Table 2: Edward curve unified addition or doubling [26]

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

In this section we portray two situations: 1) when *scenario* 1 of HCCA does not work. 2) the attack basis of *scenario* 2 of HCCA. We have utilized following elliptic curve points for our illustration: $P = (X_p, Y_p, Z_p)$, $Q = (X_q, Y_q, Z_q)$, base point $B_p = (X_b, Y_b, Z_b)$, $P_1 = (X_1, Y_1, Z_1)$, $P_1' = (X_1', Y_1', Z_1')$. We proceed with the illustration of a situation to describe when *scenario* 2 will not succeed. When the Edward curve addition is computed through the steps mentioned in Table 2, then both the doubling and addition operation contains multiplication pair satisfying *property* 3 as can be noted in the Table 3. 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 $((\mathbf{Z_1 Z_2}) \times ((Z_1Z_2)^2 - dX_1X_2Y_1Y_2),$ $\mathbf{Z_1 Z_2} \times ((Z_1Z_2)^2 + dX_1X_2Y_1Y_2))$, sharing the operand R_1 or $(\mathbf{Z_1 Z_2})$. 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 $((\mathbf{Z_1^2}) \times (Z_1^4 - dX_1^2Y_1^2),$ $(\mathbf{Z_1^2}) \times (Z_1^4 + dX_1^2Y_1^2))$ having the common operand R_1 as $\mathbf{Z_1^2}$. Hence distinction cannot be done between the two operations on the basis of the correlation values of the corresponding multiplication pairs.

Table 3: Illustrating the situation when *scenario* 1 does not work

ADD(P, P)	ADD(P, B_p)
$Z_p \times Z_p$	$Z_p \times Z_b$
${Z_p}^2 \times {Z_p}^2$	$Z_pZ_b \times Z_pZ_b$
$X_p \times X_p$	$X_p \times X_b$
$Y_p \times Y_p$	$Y_p \times Y_b$
$\dots$	$\dots$
$\dots$	$\dots$
$\dots$	$\dots$
$\mathbf{Z_p}^{\mathbf{2}} \times ({Z_p}^4 - d{X_p}^2{Y_p}^2)$	$\mathbf{Z_pZ_b} \times ((Z_pZ_b)^2 - dX_pX_bY_pY_b)$
$\dots$	$\dots$
$\mathbf{Z_p}^{\mathbf{2}} \times ({Z_p}^4 + d{X_p}^2{Y_p}^2)$	$\mathbf{Z_pZ_b} \times ((Z_pZ_b)^2 + dX_pX_bY_pY_b)$
$\dots$	$\dots$
$({Z_p}^4 - d{X_p}^2{Y_p}^2) \times ({Z_p}^4 + d{X_p}^2{Y_p}^2)$	$((Z_pZ_b)^2 - dX_pX_bY_pY_b) \times ((Z_pZ_b)^2 + dX_pX_bY_pY_b)$
$\dots$	$\dots$

Table 4 illustrates the basis of attack methodology in case of *scenario* 2 of HCCA where, there always exists a multiplication pair satisfying *property* $1(a)$ in between any two additions, for an instance $(Z_p \times \mathbf{Z_b}.\ Z_q \times \mathbf{Z_b})$. On the contrary such pair can not be found in case of two doublings.

Any two doubling operations computation

ADD(P_1, P_1)	ADD(P_1', P_1')
$Z_1 \times Z_1$	$Z_1' \times Z_1'$
${Z_1}^2 \times {Z_1}^2$	${Z_1'}^2 \times {Z_1'}^2$
$X_1 \times X_1$	$X_1' \times X_1'$
$Y_1 \times Y_1$	$Y_1' \times Y_1'$
$\dots$	$\dots$
$\dots$	$\dots$
$\dots$	$\dots$
${Z_1}^2 \times ({Z_1}^2 - d{X_1}^2{Y_1}^2)$	${Z_1'}^2 \times ({Z_1'}^2 - d{X_1'}^2{Y_1'}^2)$
$\dots$	$\dots$
${Z_1}^2 \times ({Z_1}^2 + d{X_1}^2{Y_1}^2)$	${Z_1'}^2 \times ({Z_1'}^2 + d{X_1'}^2{Y_1'}^2)$
$\dots$	$\dots$
$({Z_1}^2 - d{X_1}^2{Y_1}^2) \times ({Z_1}^2 + d{X_1}^2{Y_1}^2)$	$({Z_1'}^2 - d{X_1'}^2{Y_1'}^2) \times ({Z_1'}^2 + d{X_1'}^2{Y_1'}^2)$
$\dots$	$\dots$

Any two addition operations computation

ADD(P, $\mathbf{B_p}$)	ADD(Q, $\mathbf{B_p}$)
$Z_p \times \mathbf{Z_b}$	$Z_q \times \mathbf{Z_b}$
$Z_pZ_b \times Z_pZ_b$	$Z_qZ_b \times Z_qZ_b$
$X_p \times \mathbf{X_b}$	$X_q \times \mathbf{X_b}$
$Y_p \times \mathbf{Y_b}$	$Y_q \times \mathbf{Y_b}$
$\dots$	$\dots$
$\dots$	$\dots$
$\dots$	$\dots$
$Z_pZ_b \times ((Z_pZ_b)^2 - dX_pX_bY_pY_b)$	$Z_qZ_b \times ((Z_qZ_b)^2 - dX_qX_bY_qY_b)$
$\dots$	$\dots$
$Z_pZ_b \times ((Z_pZ_b)^2 + dX_pX_bY_pY_b)$	$Z_qZ_b \times ((Z_qZ_b)^2 + dX_qX_bY_qY_b)$
$\dots$	$\dots$
$((Z_pZ_b)^2 - dX_pX_bY_pY_b) \times ((Z_pZ_b)^2 + dX_pX_bY_pY_b)$	$((Z_qZ_b)^2 - dX_qX_bY_qY_b) \times ((Z_qZ_b)^2 + dX_qX_bY_qY_b)$
$\dots$	$\dots$

Table 4: Illustration of the *scenario* 2 of HCCA

F Brier-Joye safe sequence conversion

We show below how the Brier-Joye unified formula given in Table 1 (in Appendix A) can be transformed in the safe sequence form using our Algorithm 1. Note that we have considered here only the islands containing edges, i.e multiplications sharing at least one operand.

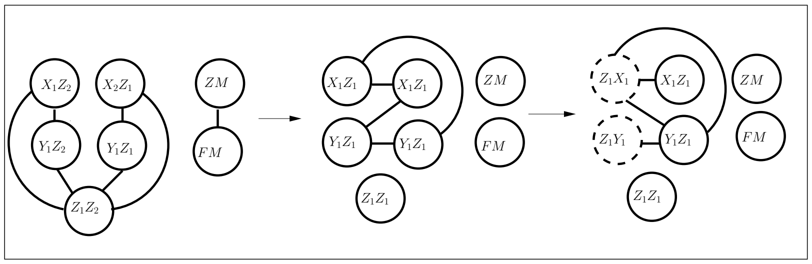


Fig. 4: Safe sequence transformation of Brier-Joye unified formula

G Long integer multiplication (LIM)

The long integer multiplication algorithm presents the school-book multiplication algorithm which has been used to compute the underlying field multiplications involved in point doubling and point addition operations.

Algorithm 5: 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
            $(U,V)_{2^w} = (U,V)_{2^w} + C$
            $(U,V)_{2^w} = (U,V)_{2^w} + R[i+j-1]$
            $R[i+j-1] = V$
            $C = U$
            $R[i+t] = C$
        end
    end
    return $R$
end
```