Journal of Cryptographic Engineering (2019) 9:263–275
https://doi.org/10.1007/s13389-019-00219-1

REGULAR PAPER

Automatic generation of HCCA-resistant scalar multiplication algorithm by proper sequencing of field multiplier operands

Poulami Das[1] · Debapriya Basu Roy[1] · Debdeep Mukhopadhyay[1]

Received: 5 February 2018 / Accepted: 28 July 2019 / Published online: 8 August 2019

Abstract

Horizontal collision correlation analysis, in short HCCA, imposes a serious threat to simple power analysis-resistant elliptic curve cryptosystems involving unified algorithms, e.g., Edwards curve unified formula. This attack can be mounted even in the 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 underlying 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 minimal 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 lightweight embedded devices as it incorporates more security per key bit compared to RSA [28], thus qualifying as a less resource hungry alternative. Also with the recent explosion of Internet of things (IoT), applications using lightweight 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 [23], even though the system is theoretically protected. Side-channel leakage of information through power consumption [23], electromagnetic (EM) dissipation, acoustic channel [14], etc. make the system weakly protected and may lead to complete secret key recovery. A naive implementation of an elliptic curve (EC) scalar multiplication algorithm, consisting of sequential doubling and addition operations, can be broken through simple power analysis (SPA) [10] 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 [8,24] which transform the doubling and addition operations 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 [11] a new form of curve, named Edwards 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 Edwards curve in cryptosystems reveals its implementation friendliness [4,17,22]. Also it is being considered as a safe curve with respect to

The authors would like to thank *ISEA Funding for Research on Next Generation Network Security* for partially supporting their work.

✉ Poulami Das
poulamidas22@gmail.com

Debapriya Basu Roy
dbroy24@gmail.com

Debdeep Mukhopadhyay
debdeep.mukhopadhyay@gmail.com

1 IIT Kharagpur, Kharagpur, India

a number of important factors (ladder security, twist security) [6]. Indeed because of the presence of single formula for both point addition and point doubling, an Edwards curve implementation, similarly Brier–Joye unified formula [7], is SPA resistant. We note here that there exist advanced attacks such as differential power analysis (DPA) attack [10] which can exploit a SPA-resistant implementation and 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 [27] 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 [2,13,16] which were mainly focused on RSA-based exponentiation algorithms. Bauer et al. in [3] have put forward the idea of horizontal attack in case of elliptic curve cryptography. The attack combines methodologies from the well-established horizontal attack [27] and the idea of collision attack (introduced in [25]), and 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 [15], address-bit differential attack [18] with effective randomization schemes suggested in [12,19], HCCA can be launched, thus introducing 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 show how we can design a minimal cost yet effective countermeasure that helps in resisting HCCA. Our contribution 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 used Pearson correlation metric [3] 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 converts the unified algorithm into a safer form, such that the relation between side-channel leakage of multiplications based on operand sharing property 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. We show how the implementation integrated with our proposed countermeasure enhances HCCA resistivity.
– Finally we provide extensive results of mounting HCCA on the proposed countermeasure. The results have been validated on SASEBO-GII with electromagnetic (EM) traces.

2 Related work

Big Mac analysis [27] 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 precomputation 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. Amiel et al. in [1] applies a novel technique of distinguishing multiplication from squaring operations based on the difference in their expected Hamming weight distribution. However, it is a vertical attack gathering information from several traces along the same region of a long integer multiplication. In [9], 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 few collisions compared to squaring due to the presence of different input operands. In [13], 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 gets exposed to vertical collision analysis, and the rest part of the key is recovered using horizontal attack techniques. In [2], a generic approach is introduced to break an ECC implementation with the help of one template trace per bit. In [16], the vulnerability of regularized algorithms such as Montgomery ladder [21] and Joye's add-only scalar multiplication [20] is highlighted, based on collisions of intermediate results obtained from consecutive iterations. In the next section, we discuss the resistance of our countermeasure from the above-mentioned horizontal attacks.

3 Preliminaries

We discuss the idea of horizontal collision correlation analysis (HCCA) attack in this section.

3.1 Horizontal collision correlation analysis

First we proceed to explain the HCCA 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* [3]. 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 and (3) $A \times B$, $A \times B$ where both the operands are same. Note that here no particular assumptions is made on the order in which the operands are passed. However, operands which are common are generally passed in the same order in the two concerned multiplications. Based on the above class of multiplication pairs, we state the following hypotheses for field multiplication pairs:

- *hypothesis* 1: when a pair of multiplications (m_i, m_j) shares one (two) common operand (s) among themselves.
  - *hypothesis* 1*a*: when a pair of multiplications (m_i, m_j) shares exactly one common operand among themselves. For example, the pair ($A \times B$, $C \times B$) satisfies *hypothesis* 1*a*.
  - *hypothesis* 1*b*: when a pair of multiplications (m_i, m_j) shares exactly two operands, i.e., they denote the same multiplications. For example, the pair ($A \times B$, $A \times B$) satisfies *hypothesis* 1*b*.
- *hypothesis* 2 when a pair of multiplications (m_i, m_j) shares no common operand among themselves. For example, the pair ($A \times B$, $C \times D$) having independent operands satisfies *hypothesis* 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, the identification of doubling and addition operations leads to the recovery of the underlying unknown key. Now we proceed to illustrate the attack scenario 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 Fig. 1. Each of the ADD/ DBL operations consist of underlying field additions and field multiplications. For instance, Fig. 1 shows that there exists a multiplication pair (X_1Y_2, X_2Y_1) within the addition operation, satisfying *hypothesis* 2 of sharing operands. A pair (X_1Y_1, X_1Y_1) can be found in case of doubling satisfying *hypothesis* 1*b* of sharing operands. Now, according to [3], if the correlation between the power traces of two concerned multiplication pairs is 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). 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. This is how an attacker can launch HCCA. The detailed intermediate steps of the Edwards curve formula vulnerable to HCCA are shown in Fig. 2.

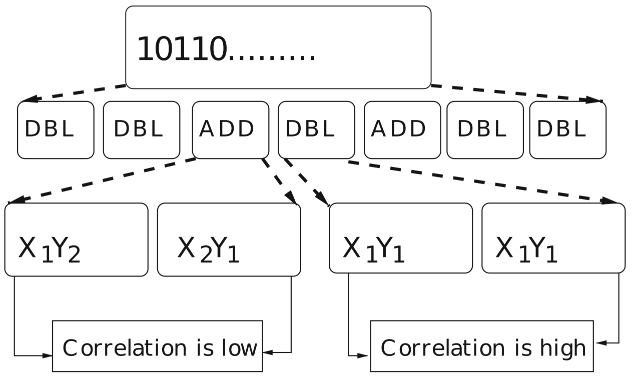


Fig. 1 Horizontal collision correlation analysis (HCCA)

4 Our proposed countermeasure

We propose here a minimal cost countermeasure technique which ensures the resistance of a unified ECC algorithm against HCCA. Our proposed countermeasure involves transforming the ECC point doubling and point addition operations into a secure form, such that even if *hypothesis* 1 holds, *it is not revealed to the adversary*. In other words, the information of one of the operations satisfying *hypothesis* 1 is hidden through our implementation. An ECC implementation integrated with our proposed countermeasure becomes more resistant against HCCA. Our countermeasure can be instantiated with minimal overhead of resources in case of the Edwards curve unified formula as well as Brier–Joye unified formula. It is based on an observation that the leakage

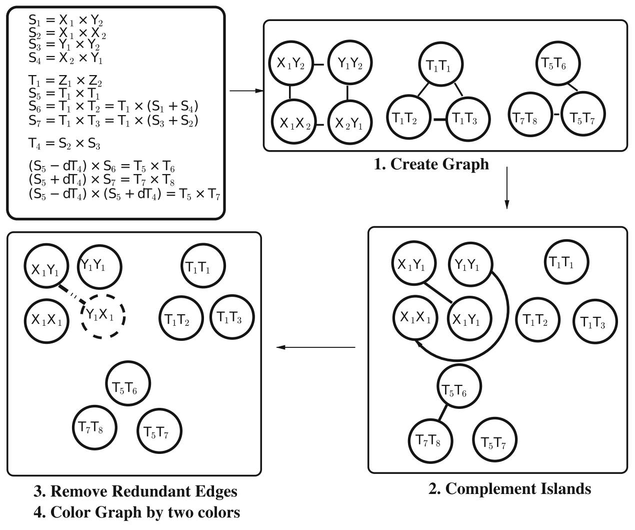

S1 = X1 × Y2
S2 = X1 × X2
S3 = Y1 × Y2
S4 = X2 × Y1
T1 = Z1 × Z2
S5 = T1 × T1
S6 = T1 × T2 = T1 × (S1 + S4)
S7 = T1 × T3 = T1 × (S3 + S2)
T4 = S2 × S3
(S5 − dT4) × S6 = T5 × T6
(S5 + dT4) × S7 = T7 × T8
(S5 − dT4) × (S5 + dT4) = T5 × T7
X1Y2
Y1Y2
X1X2
X2Y1
T1T1
T1T2
T1T3
T5T6
T7T8
T5T7
1. Create Graph
X1Y1
Y1Y1
X1X1
2. Complement Islands
Y1X1
3. Remove Redundant Edges
4. Color Graph by two colors

Fig. 2 Safe sequence transformation of Edwards curve formula: steps 1–4

ALGORITHM 1: Long Integer Multiplication algorithm(LIM)

```
Data: : {$X = (X[t], X[t-1], ...., X[1])_{2w}$ },
        {$Y = (Y[t], Y[t-1], ...., Y[1])_{2w}$ }
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
            $(U, V)_{2w} = x_i \times y_j$ ;
            $(U, V)_{2w} = (U, V)_{2w} + C$ ;
            $(U, V)_{2w} = (U, V)_{2w} + R[i + j - 1]$ ;
            $R[i + j - 1] = V$ ;
            $C = U$;
        end
        $R[i + t] = C$ ;
    end
    return $R$ ;
end
```

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 [26] in case of multipliers and swapping of operands has been suggested as a potential countermeasure, Sugawara et al. in [26] do not exploit its applicability to any ECC cryptosystem. To the best of our knowledge, this is the first countermeasure design for any elliptic curve cryptosystem which utilizes asymmetry in information leakage of multiplier operands.

4.1 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 1 (Algorithm 1 in [3]). The long integer multiplication routine is called to compute underlying field multiplications involved in the ECC point addition and doubling operations. The LIM takes two field operands X and 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.

To establish the reasoning behind asymmetry in leakage of field multiplications, we introduce here an information leakage model which will guide us toward 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 the 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 1, 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 $f(O_{i,j}) = \mathsf{H}(O_{i,j})$, where $\mathsf{H}(x)$ 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 $< \mathsf{H}(O_i) >$ or $< \mathsf{H}(a_i b_j) >$. It is evident from Algorithm 1 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 $< l_{(a_0,b_0)}$, $l_{(a_0,b_1)}, \ldots, l_{(a_0,b_{t-1})}, \ldots, l_{(a_{t-1},b_{t-1})} >$. The one obtained during computation of LIM(B, A) can be presented as $< l_{(b_0,a_0)}$, $l_{(b_0,a_1)}, \ldots, l_{(b_0,a_{t-1})}, \ldots, l_{(b_{t-1},a_{t-1})} >$. This asymmetry in the sequence of the two vectors acts as a distinguisher between two multiplications.

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

4.2 Pearson correlation metric

Considering underlying field operands as: A, B, A', B', the correlation between two long integer multiplications $\mathsf{LIM}(A, B)$ and $\mathsf{LIM}(A', B')$ can be approximated with the Pearson correlation coefficient computed between two vectors $< \mathsf{H}(a_i b_j) >$, $< \mathsf{H}(a_i' b_j') >$ (following similar notation as above). Let us denote the two vectors as $\mathsf{H}(AB)$ and $\mathsf{H}(A'B')$, respectively. The correlation is obtained by the use of *Pearson correlation coefficient* as follows:

$$\rho = \frac{\left(\frac{\sum_{i=0,j=0}^{t-1} \mathsf{H}(a_i b_j)\mathsf{H}(a_i' b_j')}{t^2}\right) - \left(\frac{\sum_{i=0,j=0}^{t-1} \mathsf{H}(a_i b_j)}{t^2}\right)\left(\frac{\sum_{i=0,j=0}^{t-1} \mathsf{H}(a_i' b_j')}{t^2}\right)}{\mathsf{std}(AB)\mathsf{std}(A'B')}. \quad (1)$$

Here the standard deviation from the information leakage of a long integer multiplication $\mathsf{LIM}(A, B)$ is denoted as $\mathsf{std}(AB)$. It is obtained as follows:

$$\begin{aligned}\mathsf{std}(AB) &= \mathsf{std}(< \mathsf{H}(AB) >)\\ &= \sqrt{\frac{\sum_{i=0,j=0}^{t-1} \mathsf{H}(a_i b_j)^2}{t^2} - \left(\frac{\sum_{i=0,j=0}^{t-1} \mathsf{H}(a_i b_j)}{t^2}\right)^2}.\end{aligned} \quad (2)$$

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

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

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

$$\begin{aligned}\alpha = (&\mathsf{H}(a_0 b_0)\mathsf{H}(c_0 b_0) + \ldots + \mathsf{H}(a_0 b_{t-1})(c_0 b_{t-1})\\ &+ \mathsf{H}(a_1 b_0)\mathsf{H}(c_1 b_0) + \ldots + \mathsf{H}(a_{t-1} b_{t-1})\mathsf{H}(c_{t-1} b_{t-1})).\end{aligned} \quad (4)$$

The following correlation is obtained from $\mathsf{LIM}(A, B)$ and $\mathsf{LIM}(B, C)$

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

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

$$\begin{aligned}\beta = (&\mathsf{H}(a_0 b_0)\mathsf{H}(b_0 c_0) + \cdots + \mathsf{H}(a_0 b_{t-1})\mathsf{H}(b_0 c_{t-1})\\ &+ \mathsf{H}(a_1 b_0)\mathsf{H}(b_1 c_0) + \cdots + \mathsf{H}(a_{t-1} b_{t-1})\mathsf{H}(b_{t-1} c_{t-1})).\end{aligned} \quad (6)$$

Here we consider the correlation coefficient between a multiplication pair with *hypothesis* 2, computed from $\mathsf{LIM}(A, B)$ and $\mathsf{LIM}(C, D)$.

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

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

$$\begin{aligned}\gamma = (&\mathsf{H}(a_0 b_0)\mathsf{H}(c_0 d_0) + \cdots + \mathsf{H}(a_0 b_{t-1})\mathsf{H}(c_0 d_{t-1})\\ &+ \mathsf{H}(a_1 b_0)\mathsf{H}(c_1 d_0) + \cdots + \mathsf{H}(a_{t-1} b_{t-1})\mathsf{H}(c_{t-1} d_{t-1})).\end{aligned} \quad (8)$$

We develop here few lemmas which will be required consequently to support the theoretical foundation of our countermeasure. As defined above, A and B denote two field multiplication operands which will be used as parameters in the LIM routine. Now we proceed to the lemmas.

Lemma 1 *The standard deviation of a Hamming weight vector obtained from* $\mathsf{LIM}(A, B)$ *is same as that obtained as* $\mathsf{LIM}(B, A)$*, i.e.,* $\mathsf{std}(AB) = \mathsf{std}(BA)$.

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

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

Lemma 2 $\mathsf{cov}(\mathsf{H}(AB), \mathsf{H}(CB)) \neq \mathsf{cov}(\mathsf{H}(AB), H(BC))$. *When* $C = A$,
$\mathsf{cov}(\mathsf{H}(AB), \mathsf{H}(AB)) \neq \mathsf{cov}(\mathsf{H}(AB), \mathsf{H}(BA))$.

Proof The two covariances $\mathsf{cov}(\mathsf{H}(AB), \mathsf{H}(CB))$ and $\mathsf{cov}(\mathsf{H}(AB), \mathsf{H}(BC))$ can be represented as

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

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

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

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

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

$$\begin{aligned}\alpha &= (\mathsf{H}(a_0b_0)\mathsf{H}(a_0b_0) + \cdots + \mathsf{H}(a_0b_{t-1})\mathsf{H}(a_0b_{t-1}) \\ &\quad + \mathsf{H}(a_1b_0)\mathsf{H}(a_1b_0) + \cdots + \mathsf{H}(a_{t-1}b_{t-1})\mathsf{H}(a_{t-1}b_{t-1})). \\ &= (\mathsf{H}(a_0b_0)^2 + \mathsf{H}(a_0b_1)^2 + \cdots + \mathsf{H}(a_0b_{t-1})^2 \\ &\quad + \mathsf{H}(a_1b_0)^2 + \cdots + \mathsf{H}(a_{t-1}b_{t-1})^2), \quad (11)\end{aligned}$$

while β can be reduced as

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

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

$$\mathsf{cov}(\mathsf{H}(AB), \mathsf{H}(AB)) \neq \mathsf{cov}(\mathsf{H}(AB), \mathsf{H}(BA)). \quad \square$$

Lemma 3 *$\rho_1 > \rho_2$ for the case: $A = C$.*

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

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

while the correlation ρ_2 between ($\mathsf{LIM}(A, B)$, $\mathsf{LIM}(B, A)$) can be computed as

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

From Lemma 3,

$$\mathsf{cov}(\mathsf{H}(AB), \mathsf{H}(AB)) \neq \mathsf{cov}(\mathsf{H}(AB), \mathsf{H}(BA)).$$

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

With the help of the lemmas discussed above, we make the following observations:
Observation 1: $\rho_1 \neq \rho_2$ From Eqs. 3, 5, we can recollect the mathematical forms of ρ_1 and ρ_2. From Lemma 1, we can conclude that $\mathsf{std}(AB) = \mathsf{std}(BA)$. As a result, the denominators in case of both the correlations are equal. From Lemma 2, we have the result

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

Consequently numerators of the two correlations are unequal. Also, since From Lemma 1,

$$\mathsf{mean}(AB) = \mathsf{mean}(BA).$$

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 $\mathsf{H}(a_ib_j)\mathsf{H}(c_ib_j)$ where the word multiplications share operand b_j and 2) each term in β is of the form $\mathsf{H}(a_ib_j)\mathsf{H}(b_ic_j)$, where the word multiplications have no common operand. Each term of α and β takes different forms yielding different values. As a result, ρ_1 is clearly not same as ρ_2.

Observation 2: ρ_2 and ρ_3 are indistinguishable 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 $\mathsf{H}(a_ib_j)\mathsf{H}(b_ic_j)$, where the word multiplications have no common operand and 2) each term in γ is of the form $\mathsf{H}(a_ib_j)\mathsf{H}(c_id_j)$, where the word multiplications are devoid of any common term. The two forms $\mathsf{H}(a_ib_j)\mathsf{H}(b_ic_j)$ and $\mathsf{H}(a_ib_j)\mathsf{H}(c_id_j)$ are indistinguishable, hence rendering β and γ being indistinguishable. We conclude from our observation that the two correlation coefficients take similar form.

Observation 3: $\rho_1 > \rho_2$ for a multiplication pair with hypothesis 1b A multiplication pair satisfying *hypothesis* 1*b* implies same multiplications are being computed. From

Lemma 3, 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 which will be explained in the following subsection.

4.3 Preventing HCCA by choosing safe sequence

The countermeasure 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 HCCA. As can be noted in Sect. 3.1, an ECC implementation becomes vulnerable to HCCA if only one of the addition or doubling operations satisfies *hypothesis* 1. The idea is to alter the operation containing *hypothesis* 1, 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 *hypothesis* 1 or not. Hence, the basis of distinction between doubling and addition operation is concealed.

We swap the operands, to blur the correlation between a pair of multiplications sharing operand (s). Let the multiplications be $(A \times B, A \times B)$ computed as $A \times B$ and $B \times A$. $\mathsf{LIM}(A, B)$ gives the expansion $(< l_{(a_0,b_0)}, l_{(a_0,b_1)}, \ldots, l_{(a_0,b_{t-1})}, \ldots, a_{(b_{t-1},b_{t-1})} >)$, and $\mathsf{LIM}(B, A)$ leads to $(< l_{(b_0,a_0)}, l_{(b_0,a_1)}, \ldots, l_{(b_0,a_{t-1})}, \ldots, l_{(b_{t-1},a_{t-1})} >)$. Here after swapping of operands, still the first and last partial products, namely a_0b_0 and $a_{(t-1)}b_{(t-1)}$, are same for both $A \times B$ and $B \times A$, and hence, an attacker may just focus on these two squares for getting similarity between $A \times B$ and $B \times A$. However, here we would like to comment that firstly, it would be extremely challenging for an attacker to extract accurately the power (EM) trace corresponding to the partial products within a long integer multiplication, where the long integer multiplication needs to be correctly located within the entire trace. If the partial product is computed as point multiplication within a clock cycle time, the trace corresponding to the partial products will be also small and give insufficient information. Secondly, if the partial products are computed for fairly large base width of 64 bits or above, then it is suggested that they be calculated as long integer multiplications also (64 bits $= 4w$, where $w = 16$ bits). Since the partial products are themselves always expanded in the swapping manner ((a_0b_0) in $A \times B$, (b_0a_0) in $B \times A$), computing them as long integer multiplications again diffuses the similarity between the partial products.

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 is 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 2 which takes care of the transformation process of our countermeasure. We proceed to portray our transformation mechanism through an illustration, which will be followed by a description of our designed Algorithm 2.

We have considered the Edwards curve unified formula shown in [5] for explaining our conversion scheme. It can be noted that the Edwards curve unified formula involves a single formula which is used for both addition and doubling. It underlies a series of field multiplication operations which are listed in Fig. 2. We note that the multiplications are written w.r.t. the point 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 *hypothesis* 1 of sharing operands (*edge property*). We observe in Fig. 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_6 which is of the form T_1T_2 (here $T_1 = Z_1Z_2$, $T_2 = S_1 + S_4$), so T_5T_6 is sharing a common operand T_1 with T_1T_1, T_1T_2 or T_1T_3. However, the multiplication output of (T_1T_2) mod F_p, where F_p is the underlying field prime, is stored in the location T_6. Considering the value of the operands T_1, T_2, T_5, T_6 as necessarily random, it can be assumed that although $T_6 = (T_1T_2)$ mod p involves T_1 in its input, it is statistically independent from T_1. The randomization technique discussed in Sect. 6 ensures the randomness of the multiplier operands.

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

$S_1 = X_1 \times Y_2$
$S_2 = X_1 \times X_2$
$S_3 = Y_1 \times Y_2$
$\mathbf{S_4 = Y_1 \times X_2}$

$T_1 = Z_1 \times Z_2$
$S_5 = T_1 \times T_1$
$S_6 = T_1 \times_1 \times (S_3 + S_2)T_2 = T_1 \times (S_1 + S_2)$
$S_7 = T_1 \times T_3 = T$

$T_4 = S_2 \times S_3$

$(S_5 - dT_4) \times S_6 = T_5 \times T_6$
$(S_5 + dT_4) \times S_7 = T_7 \times T_8$
$(S_5 - dT_4) \times (S_5 + dT_4) = T_5 \times T_7$

5. Final sequence for Edwards Curve formula

Fig. 3 Safe sequence transformation of Edwards curve formula: final step

edges obtained from the addition operation graph illustrated above are the edges *common* to both addition and doubling operations. As a result, *they do not 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 do not share operands among themselves. We also replace the vertex values with the respective forms of doubling operation. For example, X_1Y_2 will be replaced with X_1Y_1. The complement of the islands is 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 (e.g., edge between two vertices each containing value X_1Y_1) as well as redundant edges (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 an addition operation from a doubling operation. In this case, doubling operation involves X_1Y_1, X_1Y_1 operated twice, which are satisfying *hypothesis* 1*b*. On the other hand, addition operation consists of two underlying multiplications X_2Y_1, X_1Y_2 satisfying *hypothesis* 2 of sharing operands. Thus, they successfully depict HCCA. Based on ***observation* 2** and ***observation* 3**, we rearrange the multiplications as X_1Y_1 and Y_1X_1, so that their operand sharing property remains hidden. Thus, the information leakage for the pair $\mathsf{LIM}(X_1, Y_1)$, $\mathsf{LIM}(Y_1, X_1)$ will be similar to that of the pair $\mathsf{LIM}(X_2, Y_1)$, $\mathsf{LIM}(Y_1, X_2)$. So we suggest swapping the order of operands of the second multiplication. Our final sequence for Edwards curve formula is presented in Fig. 3.

From Lemma 4, 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,

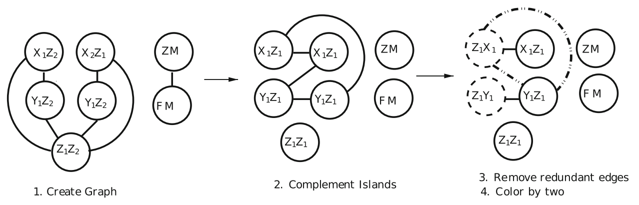


Fig. 4 Safe sequence transformation of Brier–Joye unified formula

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 are portrayed in Fig. 4.

Before proceeding to state Lemma 4, 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 vertices of the edge should be swapped and 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 it implies in case of both the vertices, the vertex operands have been swapped. But this is equivalent to the state before swapping. For example, 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 3. 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 the 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 $\{e|e = (v_i, v_j)$, 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 Lemma 4.

Lemma 4 *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 *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 satisfy 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* qualifies as a solution for *problem a*.

ALGORITHM 2: Safe_sequence_converter() : Algorithm to determine safe operand ordering of multiplication pairs

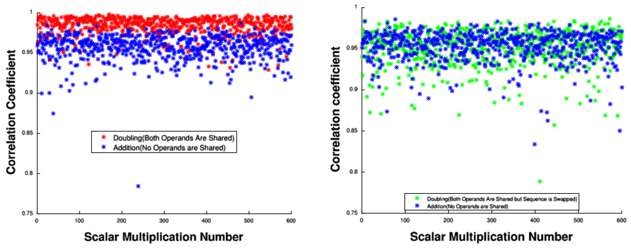
```
Data: : Set S = { $m_i$ | $i \in \{1, n\}$, where $n$ is the number of multiplications}
Result: : Set S' = { $m'_i$ | $i \in \{1, n\}$, where $n$ is the number of multiplications}

begin
    Create_Graph() ;
    Find_GraphComponents() ;
    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
        $Swap\_Order(G_1, RED)$ ;
        for $v \leftarrow 0$ to $(G_1 \rightarrow V - 1)$ do
            $S'.add(G_1 -> array[v].data)$ ;
        end
    end
end
```

5 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 [3] and our previous discussions that ECC scalar multiplication in both Edwards curve and short Weierstrass form based curve is vulnerable to HCCA. Specifically, the Edwards curve implementation incorporating unified formula is extremely vulnerable to HCCA as there exists a pair of multiplications which shares both the operands during execution of point doubling. Hence, an adversary is expected to observe sufficiently high similarities when he/she compares the power trace of aforementioned multiplications, sharing both operands.

We have considered Pearson correlation metric for our experimental validation which has been extensively recommended in the literature. We show our theory of HCCA protection is practically valid using this metric. We have used SASEBO-GII as the hardware platform for evaluating HCCA and countermeasure. All the algorithms are implemented on cryptographic FPGA of SASEBO-GII (XC5VLX50).

We show results on EM traces of actual ECC scalar multiplication for an underlying Edwards curve. We have implemented *Curve1174* on SASEBO-GII evaluation board.



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

(b) Evaluation of our countermeasure on Edwards Curve Scalar Multiplier with Correlation

Fig. 5 Evaluation of HCCA and our countermeasure on Edwards curve scalar multiplier with correlation reduces HCCA vulnerability from 93.33 to 47.5%

The cost of the implementation was 5664 slices, 16 DSPs with the frequency of 125 MHz. We have collected around 600 EM traces of scalar multiplication. As we have already mentioned in the previous sections that in Edwards curve unified formula, point doubling involves a pair of field multiplication having both of their operands shared, whereas point addition does not have any pair of field multiplications which share both of the operands, success of HCCA depends upon whether an adversary can distinguish between a pair of field multiplications having both of their operands shared and a pair of field multiplications having no common operand. If the adversary can achieve this, he can distinguish between point doubling and point addition operations which will directly give him the knowledge about secret scalar value. By using the countermeasure, we aim to remove the threat of HCCA. The objective is to make the job of distinguishability between pair of multiplications having no operand shared and pair of multiplications having both of their operands shared difficult. In Fig. 5a, red plot denotes the pair of multiplication with sharing of operands (within doubling), while the blue plot denotes multiplication pair with no sharing (within addition). It demonstrates HCCA attack, when the red plot has a higher correlation value than the blue plot with a success rate of 93.33% for 600 single scalar multiplication runs. Figure 5b contains a green plot for the pair of multiplications with operand sharing (within the doubling operation), where the operands have been swapped as a measure of the countermeasure. Besides, it contains a blue plot which denotes a pair of multiplications with no sharing of operand (within the addition). In this figure, the number of occasions when the green plot (sharing of operands) has a higher correlation than the blue plot (no sharing of operands) is 285 out of 600 cases, which gives HCCA success rate of 47.5%. Thus, swapping of operands reduces success rate of HCCA from 93.33% to 47.5% as noted using Pearson correlation metric.

ALGORITHM 3: Adding Randomization

```
Data: secret key $k = \{k_m, \ldots, k_2, k_1\}$, Base point $P$
Result: scalar product $kP$
// Precomputation Phase
for $r \leftarrow 1$ to $|L|$ do
    $store[r].point \rightarrow x = X_r$;
    $store[r].point \rightarrow y = Y_r$;
    $store[r].point \rightarrow z = Z_r$;
    $store[r].inv = (r^{deg})^{-1}$ // $r = \lambda$
    ;
end
// generate a random permutation $RP$ of the set $perm$
    $= \{i \mid i \in [1, |L|]\}$
$index = 1$ ;
// 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
```

5.1 Overhead analysis of our countermeasure

In Table 1, we give the overhead analysis of our countermeasure. As has been discussed in the previous section, Algorithm 2 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.

6 Adding randomization

To make our countermeasure resilient, we propose to add a randomization technique along with the operand swapping scheme as shown in Algorithm 3.

The technique is based on randomization of 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. Also, randomization of the base point leads to the randomization of intermediate multiplier operands which is a necessary assumption made in Sect. 4.3 to thwart 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 ran-

Table 1 Overhead analysis of our countermeasure

	Countermeasure overhead (Algorithm 4.3)
Precomputation	Algorithm 2 with $\mathcal{O}(n^2)$ time and $\mathcal{O}(n^2)$ space, where n is number of vertices in the graph
Runtime	Zero timing and area overhead

domized by using a random $\lambda \in F_p^*$ into the form R_p as $(\lambda X, \lambda Y, \lambda Z)$ [10]. 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 $\{i | i \in [1, L]\}$, where L denotes the maximum number of addition operation possible for a key $\in [1, \#(E)]$, where $\#(E)$ 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 is chosen from the consecutive elements of the set *perm*.

Our operand swapping countermeasure equipped with the above-mentioned randomization technique prevents HCCA (as evident from Sect. 5). Additionally it provides resistant against the class of Differential power attacks as proposed in [10], because of the choice of a random value of base point at each iteration of the addition algorithm. Prevention against other class of horizontal attacks is discussed in the following section.

7 Resistance against related horizontal attacks

There are other horizontal attacks that can be found in the literature, which are somewhat similar to HCCA. We demonstrate in the following how they are related to HCCA and to what extent it can be mitigated.

The attack demonstrated in [16] is applicable in case of regular algorithms, where both doubling and addition operations 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 and hence lead to retrieval of the secret key. This attack cannot be directly applied to non-regular algorithms. In [2], an incremental key retrieval process has been proposed for an ECC algorithm, where template trace is being created for the ith iteration based on the already retrieved portion of the secret and the guessed key bit for the current iteration. However, in our implementation, Algorithm 3 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.

In [9], 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 and hence lower number of collision pairs on the same architecture model. Also the paper does not present any practical results of this collision-based attack. Applicability of [9] in ECC is yet to be exploited.

7.1 Rearranging attack

One important attack which is relevant to our design is the rearranging attack. Recall that a field multiplication $\mathsf{LIM}(A, B)$ (refer to Algorithm 1) is a computation of partial products as $\sum_{i=1,j=1}^{t}(a_i b_j)$, which takes the form $\sum_{i=1,j=1}^{t}(b_i a_j)$ when we compute the field multiplication $\mathsf{LIM}(B, A)$—here t represents the number of words as in Algorithm 1. The partial products are shuffled in a deterministic way when $\mathsf{LIM}(B, A)$ is computed instead of $\mathsf{LIM}(A, B)$. An attacker may take advantage of the same observation and proceed as follows. From the power trace of $\mathsf{LIM}(B, A)$, an attacker may extract traces for all the corresponding partial products and then arrange them in the sequence which represents computation of $\mathsf{LIM}(A, B)$. So she can essentially reconstruct the trace of a $\mathsf{LIM}(A, B)$ computation from a power trace for $\mathsf{LIM}(B, A)$. Now it can launch the HCCA attack as before with an original $\mathsf{LIM}(A, B)$ trace and a reconstructed $\mathsf{LIM}(B, A)$ trace, and can distinguish it from a pair $\{\mathsf{LIM}(A, B), \mathsf{LIM}(C, D)\}$. We note here that this attack will work only if the partial product extraction is successfully performed. However if a designer wants to add a mitigation step to the rearranging attack, then we propose the following enhancement of our HCCA countermeasure. We modify Algorithm 2 as follows. Algorithm 2 outputs a set S' containing multiplications, for which the operands to LIM need to be sent in reverse order. We add an extra step here, stating the following. For all the multiplications in set S', multiplication should be computed as $\mathsf{LIM}'(B, A)$, where the algorithm LIM' takes as input, words of length $\frac{w}{2}$, and the rest is same as LIM. Through this modification, $\mathsf{LIM}'(B, A)$ will generate partial products $\sum_{i=1,j=1}^{2t}(a'_i b'_j)$, where $|a'_i| = \frac{w}{2}$, $|b'_j| = \frac{w}{2}$. This yields the same final output and, however, changes all the individual partial products; hence, rearranging attack will not work anymore. More generic modification of the LIM algorithms for multiplications in set S', such that individual partial products are different from the ones in $\mathsf{LIM}(A, B)$, solves our rearranging attack problem.

8 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 demonstrated how a unified addition (doubling) formula can be converted into a safe sequence, where the information leakage from sharing of operands among field multipliers has been hidden. The process of conversion to the desired safe sequence is achieved through our proposed algorithm (Algorithm 2), and once the sequence has been determined through our algorithm, there is no runtime overhead requirement for the countermeasure. We have validated HCCA and our proposed countermeasure scheme on a SASEBO platform. This HCCA countermeasure, since minimal cost can be easily integrated with other horizontal attack countermeasures and vertical attack countermeasures involving randomization techniques [10], thus helps in designing a secure ECC-based crypto-module. For an instance, our HCCA countermeasure integrated with a randomization-based countermeasure has been demonstrated in Sect. 6, which thwarts a larger class of horizontal attacks.

References

1. Amiel, F., Feix, B., Tunstall, M., Whelan, C., Marnane, W.P.: 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, pp. 346–360 (2008)
2. Batina, L., Chmielewski, L., Papachristodoulou, L., Schwabe, P., Tunstall, M.: Online template attacks. In: Progress in Cryptology—INDOCRYPT 2014—15th International Conference on Cryptology in India, New Delhi, India, December 14–17, 2014, Proceedings, pp. 21–36 (2014)
3. Bauer, A., Jaulmes, É., Prouff, E., Reinhard, J.-R., Wild, J.: Horizontal collision correlation attack on elliptic curves—extended version. Cryptogr. Commun. **7**(1), 91–119 (2015)
4. Bernstein, D.J., Birkner, P., Joye, M., Lange, T., Peters, C.: Twisted edwards curves. In: Progress in Cryptology—AFRICACRYPT 2008, First International Conference on Cryptology in Africa, Casablanca, Morocco, June 11–14, 2008. Proceedings, pp. 389–405 (2008)
5. Bernstein. D.J., Lange, T.: Faster addition and doubling on elliptic curves. In: 13th International Conference on the Theory and Application of Cryptology and Information Security Advances in Cryptology—ASIACRYPT 2007, Kuching, Malaysia, December 2–6, 2007, Proceedings, pp. 29–50 (2007)
6. Bernstein, D.J., Lange, T.: Safecurves: choosing safe curves for elliptic-curve cryptography (2014). http://safecurves.cr.yp.to/
7. Brier, E., Joye, M.: Weierstraß elliptic curves and side-channel attacks. In: Proceedings of 5th International Workshop on Practice and Theory in Public Key Cryptosystems Public Key Cryptography, PKC 2002, Paris, France, February 12–14 (2002)
8. Chevallier-Mames, B., Ciet, M., Joye, M.: Low-cost solutions for preventing simple side-channel analysis: side-channel atomicity. IEEE Trans. Comput. **53**(6), 760–768 (2004)
9. Clavier, C., Feix, B., Gagnerot, G., Giraud, C., Roussellet, M., Verneuil, V.: ROSETTA for single trace analysis. In: Proceedings of Progress in Cryptology—INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9–12, 2012. pp. 140–155 (2012)
10. Coron, J.-S.: 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, pp. 292–302 (1999)
11. Edwards, H.M.: A normal form for elliptic curves. Bull. Am. Math. Soc. **44**, 393–422 (2007)
12. Fan, J., Verbauwhede, I.: 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, pp. 265–282 (2012)
13. Feix, B., Roussellet, M., Venelli, A.: 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, pp. 3–20 (2014)
14. Genkin, D., Shamir, A., Tromer, E.: 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, pp. 444–461 (2014)
15. Goubin, L.: 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, pp. 199–210 (2003)
16. Hanley, N., Kim, H.S., Tunstall, M.: 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, pp. 431–448 (2015)
17. Hisil, H., Wong, K.K.-H., Carter, G., Dawson, E.: 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, pp. 326–343 (2008)
18. Itoh, K., Izu, T., Takenaka, M.: 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, pp. 129–143 (2002)
19. Itoh, K., Izu, T., Takenaka, M.: 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, pp. 382–396 (2003)
20. Joye, M.: 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, pp. 135–147 (2007)
21. Joye, M., Yen, S.-M.: 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, pp. 291–302 (2002)
22. Kim, K.H., Lee, C.O., Nègre, C.: 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, pp. 393–408 (2014)
23. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Advances in Cryptology—CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15–19, 1999, Proceedings, pp. 388–397 (1999)
24. Longa, P.: Accelerating the scalar multiplication on elliptic curve cryptosystems over prime fields. IACR Cryptol. ePrint Arch. **2008**, 100 (2008)
25. Schramm, K., Wollinger, T.J., Paar, C.: 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, pp. 206–222 (2003)
26. Sugawara, T., Suzuki, D., Saeki, M.: Two operands of multipliers in side-channel attack. IACR Cryptol. ePrint Arch. **2015**, 291 (2015)
27. Walter, C.D.: 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, pp. 286–299 (2001)
28. Wikipedia: Elliptic curve digital signature algorithm. https://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm, last edited on 7 March (2019)