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

Poulami Das, Deapriya Basu Roy, and Debdeep Mukhopadhyay

Abstract—Horizontal collision correlation analysis (HCCA) imposes a serious threat to simple power analysis resistant elliptic curve cryptosystems involving unified algorithms, for e.g., Edward curve unified formula. This attack can be mounted even in presence of differential power analysis resistant randomization schemes. In this paper, we have designed an effective countermeasure for HCCA protection, where the dependency of side-channel leakage from a school-book multiplication with the 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 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.

Index Terms—ECC, HCCA, countermeasure, asymmetric leakage, field multiplications

I. INTRODUCTION

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

The authors are with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, 721302, India, E-mail: poulamidas22@gmail.com, {deb.basu.roy, debdeep}@cse.iitkgp.ernet.in
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 relation between side-channel leakage of two multiplications sharing one (two) common operand(s) is dependent on the order of operands passed to the individual multiplications. We have used two distinguishers - Pearson Correlation metric, and Euclidean Distance metric for our analysis.
- Based on this observation, we propose a countermeasure, that can be applied to the existing unified algorithms of ECC to defeat HCCA. The countermeasure involves two steps: step 1 prevents the first scenario of HCCA, while step 2 is required to resist the second scenario. Step 1 of our countermeasure involves no randomization, instead it converts the unified algorithm into a safer form, such that the relation between side-channel leakage of multiplications based on property of operand sharing cannot be exploited. The countermeasure requires determination of the safe sequence through our proposed algorithm. As a result, there is no additional timing and area overhead on the implementation. The step 2 is based on a randomization technique that is able to counter the second phase of HCCA, by using minimal overhead. We show how the implementation integrated with our proposed countermeasure becomes resistant against HCCA.
- Finally we provide extensive results of mounting HCCA on the proposed countermeasure. The results have been validated on SASEBO-GII with Electromagnetic (EM) traces.

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

II. RELATED WORK

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

III. PRELIMINARIES

We discuss the idea of Horizontal Collision Correlation Attack (HCCA) in this section.

A. Horizontal Collision Correlation Analysis

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

- property 1: when a pair of multiplications $(m_i, m_j)$ share one (two) common operand(s) among themselves.
  - property 1a: when a pair of multiplications $(m_i, m_j)$ share exactly one common operand among themselves. For e.g., the pair ($A \times B$, $C \times B$) satisfies property 1a.
  - property 1b: when a pair of multiplications $(m_i, m_j)$ share exactly two operands, i.e. they denote the same multiplications. For e.g., the pair ($A \times B$, $A \times B$) satisfies property 1b.
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 $\text{DBL, DBL, ADD, DBL, ADD, DBL, DBL,...}$. Where DBL denotes a point doubling operation, while ADD denotes a point addition operation as shown in Figure 1(a). Each of the ADD/DBL operations consist of underlying field additions and field multiplications. For an instance, it can be observed in Figure 1(a) that there exists a multiplication pair $(X_1Y_2, X_2Y_1)$ within the addition operation, satisfying property 2 of sharing operands. While a pair $(X_1Y_1, X_2Y_1)$ can be found in case of doubling satisfying the property 1b of sharing operands. Now, according to [18] if the correlation between the power traces of two concerned multiplication pairs be considered, the multiplication pair $(X_1Y_2, X_2Y_1)$ should give low correlation value, with respect to the correlation value obtained from the multiplication pair $(X_1Y_1, X_1Y_1)$ as shown in [18]. If significant difference between the correlation values is obtained, then the doubling and addition operations can be successfully identified, leading to the complete secret key recovery.

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

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

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

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

```
Data: Point $P$, scalar $k = \{k_{m-1}, k_{m-2}, k_{m-3}...k_2, k_1, k_0\}$, where $k_{m-1} = 1$
Result: $kP$

$Q = P$
for $i = m - 2 \text{ to } 0$
do
$Q = ADD(Q, Q)$
if $k_i = 1$
$Q = ADD(Q, B_p)$
end
Return $Q$
```

We summarize below the above illustrated HCCA scenarios. An ECC point doubling operation can be decomposed into a sequence of $n_d$ multiplications given as: $\{d_1, d_2, \ldots, d_{n_d}\}$, denoted by the set $d$. Equivalently, an addition operation consists of a sequence of $n_a$ multiplications given as:
\{a_1, a_2, \ldots, a_m\}$, denoted by $set_2$. 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 $set_2$ and $set_3$ should satisfy the set property 3. If condition 1 holds, the adversary aims at differentiating between an addition and doubling operation. Consequently the adversary can recognize all the doubling and addition operations in a sequential manner by launching HCCA. If condition 1 does not hold, adversary may mount the scenario 2 of HCCA. Note that the basis of scenario 2 of HCCA is based on the fact: one of the addition parameters is always the base point, which holds independent of the underlying curve equation or the unified algorithm steps involved in the scalar multiplication. The method of correlating between a pair traces has been taken in [18] as the Pearson Correlation Distinguisher. However we show in later sections that better results of HCCA and validation of our proposed countermeasure can obtained from Euclidean Distance Distinguisher.

IV. OUR PROPOSED COUNTERMEASURE

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

A. Asymmetric Leakage of Field Multiplication

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

Algorithm 2: Long Integer Multiplication algorithm (LIM)

Data: $\{X = (X[0], X[1]t-1), \ldots, X[l-1]\}$, $Y = (Y[0], Y[1]t-1), \ldots, Y[l-1]\}$

Result: $\{X \cdot Y\}$

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
      \[ (U,V)_{2w} = (U,V)_{2w} + C \];
      \[ (U,V)_{2w} = (U,V)_{2w} + R[i+j-1] \];
      \[ R[i+j-1] = V \];
      $C = U$;
      \[ R[i+t] = C \];
    end
  end
  return $R$;
end

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

\[ l(a_{i-1}, b_{n-1}) > \]. While the one obtained during computation of LIM(B, A) can be presented as 

\[ l(b_0, a_0), l(b_0, a_1), \ldots, l(b_0, a_{i-1}), \ldots, l(b_{n-1}, a_1) >. \] This asymmetry in the sequence of the two vectors contribute as a distinguisher between two multiplications. The following lemma has been introduced to emphasize the dependence of information leakage from the intermediate partial products \( l(a_i, b_j) \) on the input operands \( A, B \) during the computation of LIM(A, B).

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

**Proof.** Let \( Y = (y_1, y_2) \) be the event that the Hamming weight of the output of two multiplications \( y_1 \) and \( y_2 \) collide. Let \( X \) be a random variable such that \((X = a)\) denotes the colliding Hamming weight of the event \((Y = (y_1, y_2))\) is \(a\). If each operand are of size \( w\) bits, then the number of all possible values of one operand is \(2^w\).

**Case 1:** The probability of collision of the pair \((mn, pn)\) is computed as \( P_1 = P(Y = (mn, pn)) = \{P(X = 0) \cup P(X = 1) \cup P(X = 2) \cup \ldots \cup P(X = w)\} = P(X = 0) + P(X = 1) + P(X = 2) + \ldots + P(X = w)\). Now, \( P(X = a) = \frac{n_a}{2^{2w}} \), 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 \( P_1 = \frac{n_0}{2^{2w}} + \frac{n_1}{2^{2w}} + \ldots + \frac{n_w}{2^{2w}} = \frac{n_0 + n_1 + \ldots + n_w}{2^{2w}} \).

**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, \( P(X = a) = \frac{2n_a}{2^{2w}} \), 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 \( P_2 = \frac{2n_0}{2^{2w}} + \frac{2n_1}{2^{2w}} + \ldots + \frac{2n_w}{2^{2w}} = \frac{n_0 + n_1 + \ldots + n_w}{2^{2w}} \).

From the probability values of \(P_1\) and \(P_2\), we observe that \(P_1 > P_2\). Hence proved.

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

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

\[
\rho = \frac{\text{Covariance}(H(AB), H(A'B'))}{\sqrt{\text{Variance}(H(AB))} \sqrt{\text{Variance}(H(A'B'))}}
\] (1)

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

\[
\text{std}(AB) = \text{std}(<H(AB)>)
\] (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)

\[
\rho_1 = \left(\frac{\sum_{i=0}^{t-1} H(a_i b_j) H(c_i b_j)}{\text{std}(AB) \text{std}(CD)}\right)
\] (3)

We denote \(t \leq n\).

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

\[
\rho_2 = \left(\frac{\sum_{i=0}^{l-1} H(a_i b_j) H(b_i c_j)}{\text{std}(AB) \text{std}(BC)}\right)
\] (5)

We denote \(l \leq n\).

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

\[
\rho_3 = \left(\frac{\sum_{i=0}^{m-1} H(a_i b_j) H(c_i d_j)}{\text{std}(AB) \text{std}(CD)}\right)
\] (7)

We denote \(m \leq n\).

Here \(l \leq n\).

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

**Proof.** The vector composed from leakage information of LIM(A, B) can be expanded as \(\langle H(a_0, b_0), H(a_0, b_1), \ldots\)
While, the correlation $\rho_2$ between $(\text{LIM}(A, B), \text{LIM}(B, A))$ can be computed as

$$\rho_2 = \frac{\text{cov}(H(AB), H(BA))}{\sqrt{\text{var}(H(AB))\text{var}(H(BA))}}$$

$$< 1$$

Since from Lemma 3,

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

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

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

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

$$\text{Cov}(H(AB), H(BC)) \neq \text{Cov}(H(AB), H(BC)).$$

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

Observation 2: $\rho_2 \approx \rho_3$ To make a comparison between the values of $\rho_2$ and $\rho_3$, we look at the form of each of the terms present in the two equations take: 1) each term in $\beta$ is of the form $H(a_ib_j)H(c_i d_j)$, where the word multiplications have no common operand. 2) each term in $\gamma$ is of the form $H(a_ib_j)H(c_i d_j)$, where the word multiplications are devoid of any common term. From our conclusion 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 1b, implies same multiplications are being computed. From Lemma 4, we obtain that in such a case $\rho_1$ will always be greater than $\rho_2$ irrespective of the underlying field element values involved. Hence $\rho_1 > \rho_2$ occurs with high probability in such a case.

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

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

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

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

\[
d_{\text{mean}} = \frac{1}{T^2} \sum_{i=1}^{T^2} d_{ij}
\]

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

B. Preventing scenario 1 by choosing safe sequence

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

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

![Fig. 2: Safe sequence transformation of Edward curve formula](image)

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 \((X_1, X_2, X_1Y_1)\), \((X_1X_2, X_2Y_1)\), \((Y_1Y_2, Y_1Y_2)\) 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 \(T_5, T_6\) involve operand \(T_3\) 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 \(T_3\). Now we make a crucial observation that, the operand sharing obtained from the graph considered reveals all the operand sharing multiplications which will be present in the addition operation. But if we consider the graph corresponding to the doubling operation where points \((X_1, Y_1, Z_1)\), and \((X_2, Y_2, Z_2)\) are the same, it can be observed that the previous operand sharing will still be present along with some possible extra operand sharing vertices. So the operand sharing edges obtained from the addition operation graph illustrated above are the edges common to both addition and doubling operations. As a result, they don’t qualify in distinguishing between addition and doubling operations. Evidently, the operand sharing edges which are found only in case of doubling operation may contribute in the distinction. To get a closer look we consider the complements of the islands of our previously constructed graph. Note that we are not interested in the edges between islands in the complement graph because they don’t share operands among themselves. We also replace the vertex values with the respective forms of doubling operation. For e.g. \(X_1Y_2\) will be replaced with \(X_1Y_1\). The complement of the islands are considered here to concentrate on those edges which will be formed only in case of doubling operation. However the complement of the islands will include both essential edges.
(for e.g. edge between two vertices each containing value $X_1Y_1$) as well as redundant edges (for e.g. edge between two vertices with values $X_1X_1$ and $Y_1Y_1$ respectively which do not satisfy the edge property). We remove the redundant edges, and look only at the essential edges because they are the ones which will help in distinguishing and addition operation from a doubling operation. In this case, doubling operation involves $X_1Y_1$, $X_1Y_1$ operated twice, which are satisfying property 1b. On the other hand, addition operation consists of two underlying multiplications $X_2Y_1$, $X_1Y_2$ satisfying property 2 of sharing operands. Thus they successfully depict scenario 1 of HCCA. Based on our observation 2 and observation 3, we rearrange the multiplications as $X_1Y_1$ and $Y_1X_1$, so that the their operand sharing property remains hidden. As was observed in subsection 3, the information leakage for the pair $\text{LIM}(X_1, Y_1)$, $\text{LIM}(Y_1, X_1)$ will be similar to that of the pair $\text{LIM}(X_2, Y_1)$, $\text{LIM}(Y_1, X_2)$, (here we refer to the long integer multiplication routine LIM). So we suggest to swap the order of operands of the second multiplication. From lemma IV.5 stated next we get that the problem of swapping operands of field multiplications can be solved by the problem of two-colorability of a graph. So if the final reduced graph with the islands containing essential edges be two-colorable, then we proceed to color the graph with two colors, and eventually swap the operands of those vertices which belong to the class of one particular color.

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

![Fig. 3: Safe sequence transformation of Brier-Joye unified formula](image)

Before proceeding to state the following lemma IV.5, we give here a rationale behind the operand swapping problem formulation. In our operand swapping problem, we need to identify a set of vertices which need to go through operand swapping, keeping other vertices intact as before so that the overall set reaches a secure form. So it is depictable that the vertex set needs to be partitioned into two sets. The set of vertices which requires operand swapping is called the swap set, while the other set is named as uninterrupted set. Also it can be perceived that in any edge, since the edge has been created due to operand sharing of two vertices, one of the vertex of the edge should be swapped, thus should belong to swap set. While the other vertex should belong to the uninterrupted set. Furthermore, there does not exist an edge such that both of their end vertices belong to the swap set, or the uninterrupted set. Suppose there exists one such edge, then if both vertices belong to the swap set then then it implies in case of both the vertices, the vertex operands have been swapped. But this is equivalent to the state before swapping. For e.g. it means a vertex pair $(X_1Y_1, X_1Y_1)$ has been swapped to $(Y_1X_1, Y_1X_1)$, which does not solve our aim of information masking through operand swapping. This is because the correlation between both the mentioned pairs will be higher with respect to the pair $(X_1Y_1, Y_1X_1)$, as has been proved in lemma IV.4. From this it directly follows why must the vertex ends of any edge belonging to the set $E$ should not belong to the same set (swap set or uninterrupted set). Naturally, it is 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 $\{v \mid$ operands of $v$ should be swapped\}, 2) $V_2$ or uninterrupted set, can be presented as $\{v \mid$ operands of $v$ should not be disturbed\}. 3) the edge set $E$ is of the form $\{(v_1, v_2) \mid (v_i \in V_1, v_j \in V_2)\}$ or $\{(v_i \in V_2, v_j \in V_1)\}$.

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

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

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

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

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

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

Create_Graph(): 
begin
  Initialize Graph G;
  for i ← 1 to n do
    AddVertex(G, S[i]); // create vertices of graph G
  end
  for i ← S[0] to S[n-1] do
    for j ← S[0] to S[n-1] do
      if i ≠ 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 ← 0 to G \ V - 1 do
    Visited[v] = FALSE
  end
  seg_count = 1:
  for v ← 0 to G \ V - 1 do
    if Visited[v] == FALSE then
      Island[seg_count] = Clone_Graph(G, v): // 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

Find_Safeseq_G(): // find safe sequences
begin
  for i ← 0 to seg_count - 1 do
    G_i = Construct_ComplementGraph( Island[i] );
    Remove_redundant_edges(G_i);
    // remove the edges not satisfying the edge property
    if Colorable_2(G_i) == TRUE then
      Color_Graph(G_i, RED, BLACK);
    end
    Swap_Order(G_i);
    for v ← 0 to (G \ V - 1) do
      S'.add(G_i \rightarrow array[v].data);
    end
  end
end
```

C. Preventing scenario 2 by randomizing the Base point

Once the adversary fails to launch the scenario 1, she may exploit the possibility of scenario 2 of HCCA. As was discussed in section III-A, it is based on the observation that during run of left-to-right scalar multiplication (Algorithm 1), 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 $\{m_i \mid m_i \in \text{addition}_i\}$ while $S_2$ denoted as $\{m_j \mid m_j \in \text{addition}_j\}$. 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 $m_1 \in \text{addition}_i$, $m_2 \in \text{addition}_j$ and $(m_1, m_2)$ shares one multiplication operand satisfying property 1a. With this observation the attacker can launch HCCA on a single trace and identify all the addition operations, subsequently also the doubling operations.

We propose here a method based on a randomization scheme which aims at thwarting the scenario 2 of HCCA with minimal timing or area overhead. The technique is based on the idea of randomizing the base point at every execution of addition operation so that any two multiplications chosen from two addition operations become free from the operand sharing property. Based on standard projective coordinate system, the equivalence between two elliptic curve points can be defined as $(X_1, Y_1, Z_1) \sim (X_2, Y_2, Z_2)$ if $X_2 = \lambda X_1, Y_2 = \lambda Y_1$ and $Z_2 = \lambda Z_1$, where $\lambda \in D^*_p$. Any point $(X, Y, Z)$ can be randomized by using a random $\lambda \in D^*_p$ 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 $\{i \mid i \in [1, L]\}$, where $L$ denotes the maximum number of addition operation possible for a key $\in [1, \text{order}(E)]$, $\text{order}(E)$ is the order of the underlying elliptic curve. Every execution of the scalar multiplication algorithm involves one random permutation of the set perm. The $\lambda$ value chosen for consecutive addition operations are chosen from the consecutive elements of the set perm. The addition operation once achieved by using the random point $R_p$ requires derandomization such that the final result is same as that obtained from the fixed base point $B_p$.

The derandomization involves three field multiplications of the form $(\lambda^{k-1}) \times X_3, (\lambda^{k-1}) \times Y_3, (\lambda^{k-1}) \times Z_3$ per addition operation, where $(X_3, Y_3, Z_3)$ represents the intermediate output by using $R_p$ $(k$ is a constant). The computation involving $\lambda X, \lambda Y, \lambda Z$ and $(\lambda^{k-1})$ for varied $\lambda \in \text{perm}$ 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 $\lambda$ chosen. Thus the only extra cost involved in incorporating this countermeasure step includes three field multiplications per addition operation. It should be noted that only step 2 cannot prevent HCCA, since the scenario 1 of HCCA can be exploited within an addition and a doubling operation, although the base point randomization is present. Hence to prevent HCCA completely we require to integrate our countermeasure to the ECC design containing both the steps.

D. Overhead analysis of our countermeasure

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

<table>
<thead>
<tr>
<th>Precomputation</th>
<th>countermeasure step 1</th>
<th>countermeasure step 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithms I with (O(n^2)) time and (O(n^2)) space, where (n) is number of vertices in the graph</td>
<td>computation of ((\lambda X, \lambda Y, \lambda Z)), where (\lambda \in {1, \ldots, L}). This process requires (O(L)) time and (O(L)) space</td>
<td></td>
</tr>
<tr>
<td>Runtime</td>
<td>zero timing and area overhead</td>
<td>timing overhead of three field multiplications per addition, zero area overhead</td>
</tr>
</tbody>
</table>

**V. EXPERIMENTAL RESULTS**

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

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

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

Thus Euclidean distance between two region of interest \(T_{1,R}\) and \(T_{2,R}\) belonging to two EM traces \(T_1\) and \(T_2\) is calculated as follows:

\[
d = \frac{1}{n} \sum_{i=0}^{n-1} |T_{1,R,i} - T_{2,R,i}| \tag{15}
\]

where \(n\) is the length of the region of interest. So for each pair of regions of interest we calculate a single Euclidean distance value.

Similarly to compare Pearson’s correlation coefficient with Euclidean distance as a distinguisher, we have computed Pearson’s correlation coefficient between each region of interest for every clock cycle between a pair of EM traced, where each EM trace denotes a multiplication operation. Thus in this case also for each pair of regions of interest we get a single correlation value.

We will first show our evaluation of a standalone field multiplication operations. As we have already mentioned we have collected around 1000 EM traces for each multiplication operation indicating three different scenario: 1. A pair of multiplication having both of their operand shared 2. A
correlation as shown in figure 6(a). The reason behind this disparity is as follows: correlation is a measure of similarity between two vectors. So if two regions of interest are of same nature but are separated by some distance, they will highly correlate with each other. Now, due to switching activity of the circuit, the draws power from the supply and that is indicated on the EM trace by the dip from the reference level. A high switching activity will induce more dip from the reference level and low switching activity will induce less dip in the EM trace. Hence two regions of interest corresponding to two different Hamming weights will have different dip from the reference level, but will be of same nature. Hence they can be better distinguished by Euclidean distance rather than correlation.

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

Next, we will show results on EM traces of actual ECC scalar multiplication in Edward curve. We have implemented Curve1174 on SASEBO-GII evaluation board and have collected around 400 EM traces of scalar multiplication. As we have already mentioned in the previous sections that in Edward curve unified formula, point doubling involves a pair of field multiplication having none of their operand shared and 3. A pair of multiplication having both operand shared but operand sequence is swapped. Analysis of HCCA and countermeasure step 1 using Euclidean distance and Pearson correlation coefficient distinguisher are presented in figure 5 and 6. In Figure 5(a), the black plot denotes Euclidean distance between a pair of multiplications sharing both their operands, whereas the grey plot denotes Euclidean distance between a pair of multiplications having no operands shared. The disparity between the two plots allows the adversary to clearly distinguish between a pair of multiplications having shared operand and a pair of multiplications having no shared operand. However, this decision can not be taken by using correlation as shown in figure 6(a). The reason behind this disparity is as follows: correlation is a measure of similarity between two vectors. So if two regions of interest are of same nature but are separated by some distance, they will highly correlate with each other. Now, due to switching activity of the circuit, the draws power from the supply and that is indicated on the EM trace by the dip from the reference level. A high switching activity will induce more dip from the reference level and low switching activity will induce less dip in the EM trace. Hence two regions of interest corresponding to two different Hamming weights will have different dip from the reference level, but will be of same nature. Hence they can be better distinguished by Euclidean distance rather than correlation.

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

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

VI. RESISTANCE AGAINST RELATED HORIZONTAL ATTACKS

Here we provide an explanation behind how our countermeasure scheme is protected against related horizontal attacks present in the literature.

A. Resistance against [16], [17]

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

B. Resistance against [25]

In [25] an attack on RSA is demonstrated, where long integer multiplication computation has been exploited. In case of a squaring operation, the long integer multiplication on operands of length \( l \) words involves \( (2^l - 1)/2 \) potential collision pairs of single precision multiplications, which makes the long integer operation vulnerable. However in case of field multiplications of ECC, the number of collision pairs present are less, because of lower bit length of field multiplier operands, hence lower number of collision pairs on the same architecture model. Also the paper does not present any practical results of this collision based attacks. Applicability of [25] in ECC is yet to be exploited.

VII. A NOTE ON OTHER COUNTERMEASURES

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

VIII. CONCLUSION

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

References