TVLA*: Test Vector Leakage Assessment on Hardware Implementations of Asymmetric Cryptography Algorithms

Aruna Jayasena, Emma Andrews, *Student member, IEEE,* and Prabhat Mishra, *Fellow, IEEE,*

***Abstract*—Test Vector Leakage Assessment (TVLA) evaluates the side-channel leakage of sensitive information from the hardware implementation of a design. While TVLA for symmetric cryptography has been well studied, it is not applicable to asymmetric cryptography algorithms. Asymmetric key algorithms involve complex computations in multiple stages that can lead to varying trace lengths depending on input parameters and associated constraints. In this paper, we design an effective TVLA technique for asymmetric key cryptosystems that can compare lengthy trace data with a good statistical resolution and generate valid input (test) patterns to satisfy specific constraints. Specifically, this paper makes the following major contributions. The proposed test generation algorithm can produce valid test patterns to maximize the power signature differences. Our proposed partition-based differential power analysis can significantly improve the TVLA accuracy. Extensive evaluation using elliptic curve cryptography algorithms demonstrates that the proposed TVLA framework can handle type 1 and type 2 statistical errors and evaluate hardware implementations of asymmetric cryptography algorithms with a statistical confidence of 99.999%.**

***Index Terms*—Hardware Security, Side-Channel, Test Vector Leakage Assessment, Asymmetric Key Cryptography**

I. Introduction

Symmetric cryptography, also known as secret-key cryptography, relies on a single key to perform encryption and decryption. It is easy to implement but the key distribution is a major concern. In contrast, asymmetric cryptography, also known as public-key cryptography, uses a pair of keys (public, private) for authentication or authenticated encryption. When encrypting a message with asymmetric cryptography, the public key is used by the sender for encryption. The private key is used by the recipient during decryption. This eliminates the practical limitation of key distribution in symmetric cryptography. There are also hybrid systems that utilize both symmetric and asymmetric cryptography, such as Elliptic Curve Integrated Encryption Scheme (ECIES). There are various efforts to perform side-channel leakage analysis of symmetric-key cryptosystems using Test Vector Leakage Assessment (TVLA) [1], [2]. Unfortunately, the existing TVLA methods are not applicable for evaluating asymmetric key cryptosystems. It is important to evaluate the side-channel vulnerability of asymmetric key algorithms to design trustworthy systems.

A. State-of-the-Art and Limitations

The intuition behind TVLA of hardware implementations is to provide a certain guarantee that the implementation does not reveal secrets through power side-channel signatures during execution. Figure 1 illustrates the major steps involved in TVLA of symmetric key cryptography algorithms [1], [2]. The first step performs hamming distance based test generation to produce differences in power signature [1], [2]. Next, the design is simulated with the generated key pairs and a fixed plaintext. The power signature is constructed based on the simulation's value change dump. Next, the difference between two power signatures is calculated using statistical methods, such as *t-test and KL-divergence* [1]–[3]. Finally, the implementation is categorized as safe or side-channel vulnerable based on a pre-determined threshold. While this method works well on symmetric cryptosystems, it is not applicable on asymmetric cryptosystems due to the following fundamental challenges associated with asymmetric-key algorithms:

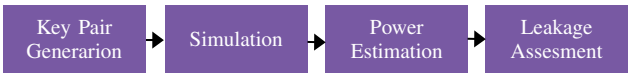


Fig. 1: TVLA steps for symmetric-key cryptosystems.

1) Involves complex computations that lead to significantly longer trace data compared with symmetric cryptography.
2) Implementations without timing mitigation can lead to varying trace lengths, while existing TVLA techniques expect fixed-length traces or manual trace alignments.
3) Timing-specific information such as specific places the power peak has occurred are not captured by applying the standard *Welch t-tests* and *KL-divergence* based methods.
4) Evaluation of asymmetric cryptography needs to consider side-channel leakage of multiple stages independently.
5) Input parameters and associated constraints are significantly different from symmetric cryptography.

B. Research Contributions

In this paper, we address the first and third challenges by evaluating the divergence between two traces with higher resolution by partitioning the traces of each stage and evaluating each partition independently. We resize the traces for each stage over the time axis to the same length using control flags to address the second challenge. We also utilize the control flags to automatically identify each stage of the implementation and perform leakage assessment separately for each stage focusing on security guarantees of the particular stage to address the fourth challenge. Finally, we propose an automated test generation framework to address the fifth challenge. *To the best of our knowledge, our approach is the first attempt in developing a TVLA framework focused on asymmetric key cryptography algorithms.* This work enables

A. Jayasena, E. Andrews, and P. Mishra are with the Department of Computer & Information Science & Engineering, University of Florida, Gainesville, Florida, USA.

a designer to evaluate systems consisting of both symmetric and assymetric components.

The remainder of this paper is organized as follows. Section II provides background and surveys related efforts. Section III describes our proposed *TVLA** methodology. In Section IV, we demonstrate that *TVLA** can reveal more accurate leakage results compared to the state-of-art TVLA approaches. Finally, Section V concludes the paper.

II. Background and Related Work

In this section, we first provide background on elliptic curve cryptography. Next, we outline vulnerabilities in asymmetric cryptography algorithms and review existing attacks. Finally, we survey existing TVLA methods and discuss their limitations.

A. Elliptic Curve Cryptography

Elliptic curves are special plane curves over a field, primarily using Galois fields. All operations are done modulo p, where p is the value of the prime for the defined prime curve [4]. Due to this, elliptic curves make a strong choice for usage in asymmetric cryptography. Elliptic curve cryptography (ECC) is the set of algorithms that use elliptic curves to provide a security guarantee, such as authentication through signatures or encryption and decryption [5]. While the requirement for an elliptic curve is to use a prime number, when making security considerations this should be a large enough number to create certain security guarantees. Therefore, standardized curves from NIST, SEC, and other sources have been deemed secure for standardized usage.

RSA is the oldest and most popular choice for asymmetric key cryptography algorithms due to its simplicity and establishment in legacy programs [6]. In order to have better security over RSA, points on the curve are used as the numbers to perform operations over for ECC [7]. A point is on the curve and thus valid if it satisfies the elliptic curve equation. There are several different coordinate representations for an elliptic curve point. For brevity, we will only consider operating on affine coordinate points and Jacobian coordinate points. The affine representation of a point is (X, Y) and is the general representation of an elliptic curve point. Jacobian representation is (X, Y, Z). An affine point in Jacobian form is $(X, Y, 1)$. The Z coordinate in Jacobian representation stores all the divisions that take place throughout any mathematical operations performed to the point. Algorithm 1 converts Jacobian projective coordinates to the equivalent affine coordinates.

Algorithm 1 Coordinate Conversion

Require: $P : (X_1, Y_1, Z_1), P \neq \mathcal{O}$
Ensure: $R : (X_2, Y_2)$
1: $\lambda \leftarrow Z_1^{-1} \mod p$
2: $X_2 \leftarrow \lambda^2 X_1$
3: $Y_2 \leftarrow \lambda^3 Y_1$

B. Vulnerabilities of Asymmetric Key Cryptography

Asymmetric key cryptography algorithms consist of multiple functions. The vulnerability analysis needs to check for vulnerabilities in each function. For example, ECC module consists of various sub-modules, such as scalar multiplication, coordinate conversion, etc. Let us take the example of scalar multiplication to illustrate the vulnerabilities. With elliptic curves, scalar multiplication is the equivalent of repeated additions of a point on the curve. However, the operation is dependent on the value of a bit in the scalar. ECC uses the private key as the scalar to generate the public key. Therefore, one of the scalar multiplications performed during an ECC algorithm is dependent on the value of the private key [8], [9]. As seen in Algorithm 2, if the current bit of the private key is a value of 1, an extra computation step must be performed. The extra computation step allows an attacker to analyze the power traces for increased use of power to perform this operation, recovering the private key. A similar notion of branching operations with different computational requirements based on the bit value of the private key is also present in RSA and other asymmetric cryptographic algorithms [10].

Algorithm 2 Scalar Multiplication - Double and Add

Require: $P : (X, Y), P \neq \mathcal{O}, k$ positive integer
Ensure: kP
  $R_0 \leftarrow \mathcal{O}$
  $R_1 \leftarrow P$
  for bit **in** k **do**
    if bit $= 1$ **then**
      $R_0 \leftarrow R_0 + R_1$
    end if
    $R_1 \leftarrow 2R_1$
  end for
  return R_0

C. Related Work: Attacks on Asymmetric Key Cryptography

Cryptographic algorithms typically have some control flow dependency as part of their operations. While this is inherently secure, it does pose a security risk if the control flow depends on some secret information, such as the private key. An attacker can then carry out side-channel attacks on the implementation to retrieve a complete or partial private key, effectively exposing secret information [11]–[15]. Minerva is an example of a recent attack on the scalar multiplication implementations of open-source software libraries that used lattices in conjunction with other leakage information to recover the private key [16]. A projective to affine coordinate conversion attack on elliptic curve cryptography is proposed in [17], [18]. Modern ECC software implementations were detected with the vulnerability of projective coordinate leakage [19], showing the feasibility of recovering the private key completely through side-channel analysis. This also illustrated the need for a side-channel leakage assessment of cryptographic implementations.

D. Collisions in Asymmetric Cryptographic Algorithms

In cryptography, collision attacks are primarily used with breaking hashing algorithms. These collision attacks work by making different inputs result in the same output, which can

be used to reveal details of the internal algorithm, effectively reverse-engineering it. For algorithms like ECDSA, secret information can be linked to certain steps of the computation. Since ECC implementations can be attacked using side-channels, combining side-channel attacks with collision attacks creates a new attack vector. This attack vector is known as *horizontal collision correlation analysis* [20]. The classic side-channel attacks are thus distinguished as *vertical attacks*. Power analysis detects spots where collisions occur during the internal computations of point addition and point doubling. These two operations are important due to their usage in scalar multiplication, where the private key is multiplied by a fixed elliptic curve point. After gathering observations on the intermediate registers, Pearson's coefficient is used to derive the secret key.

E. Limitations of Existing Side-Channel Leakage Assessment

The existing TVLA methods are suitable for symmetric key algorithms. However, the existing methods are not applicable on asymmetric key algorithms for the following reasons.

Validity of Inputs: There are multiple parameters as inputs to the asymmetric cryptography algorithms and not all possible inputs are valid inputs to the algorithm. For example, [2] generates key pairs and random plaintext messages, but authenticated encryption algorithms like ECIES do not have inputs for secret keys, instead, it accepts public key, which is not a secret and a specific point on an ECC curve. Moreover, generating random inputs is not an option, since this may lead to invalid states. A set of guidelines for generating test vector pairs for Elliptic Curve Digital Signature Algorithm (ECDSA) with possible collision attacks is discussed in [21].

Computing in Multiple Stages: Asymmetric cryptography is a sequential process, where we need to perform certain steps to complete the encryption/decryption or sign/verify process. These steps are supposed to preserve secrecy guarantee on different input parameters, such as nonce, coordinate points, Enc/Dec_Key, etc. This requirement is also not addressed by existing TVLA techniques.

Long Execution Traces: Statistical techniques such as Welch t-tests and KL divergence are directly used to perform the differential power analysis of power signature traces in [1]–[3]. This works with symmetric key algorithms since these algorithms perform block-wise operations. In fact, Hamming distance-based input key pairs used in [2] can perform well with block-wise operations. Divergent values for the traces are calculated for a small number of clock cycles since the clock cycle depth for block cipher algorithms is significantly lower (order of 100) than asymmetric cryptography algorithms (order of 10000). The direct application of Welch t-tests and KL divergence techniques can hide small but important variations in the traces of asymmetric cryptography algorithms which defeats the purpose of TVLA.

Diversity of Algorithms: There are various types of algorithms proposed for the implementations of asymmetric cryptography with different objectives (security, speed, area, power, etc.). These algorithms have different computation times. In fact, there are algorithms that take different computation times based on the input values [22], [23]. This will lead to incorrect assessment by [1]–[3].

III. TVLA for Asymmetric Cryptography (TVLA*)

Figure 2 provides an overview of our proposed TVLA* framework for side-channel leakage evaluation of asymmetric key algorithms that consists of five major tasks. The first task analyzes the design specification to identify the sequence of steps involved in the algorithm as well as different inputs and associated constraints. The second task generates input (test) vectors focusing on the secrecy guarantee of the algorithm followed by instrumentation of the testbench for simulation. The third task simulates the design to obtain the power traces. The fourth task performs the leakage assessment on generated power profiles using both simple and differential power analysis. The last task computes the divergence factor to identify if the implementation has a side-channel vulnerability. The remainder of this section describes these steps in detail.

A. Design Specification

A major architectural difference between asymmetric and symmetric cryptography algorithms is the sequence of independent stages involved in them. Since the functionality of each stage is different, each stage is expected to have different power signatures during execution. Figure 3 shows the seven stages to encrypt a message with the public key using ECIES: (i) elliptic curve multiplication, (ii) coordinate conversion step from projective to affine, (iii) elliptic curve multiplication, (iv) projective to affine coordinate conversion step, (v) key derivation (KDF) step with ANSI-X9.63, (vi) encryption stage with AES, and (vii) generation of message authentication code with HMAC-SHA1. The ECIES decryption algorithm also follows several stages in order to successfully decrypt the message. We can analyze the secrecy guarantee of each stage based on the feasible vertical and horizontal collisions that can be manipulated with inputs to the algorithm. Next, we need to analyze the input constraints such as supported curves by the algorithm, which we need to consider during the test generation step.

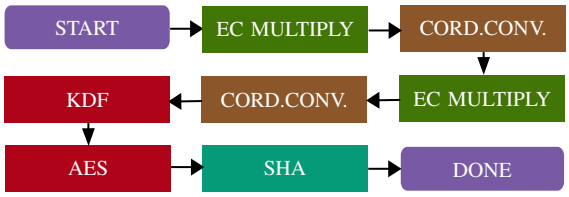


Fig. 3: Different stages during ECIES encryption.

B. Test Generation

The objective of test generation is to produce multiple pairs of input vectors such that it maximizes the difference in the power signature of the same implementation. Depending on the asymmetric cryptography algorithms, the inputs are different. For example, Table I illustrates types of inputs for different ECC algorithms. Since we have diverse input

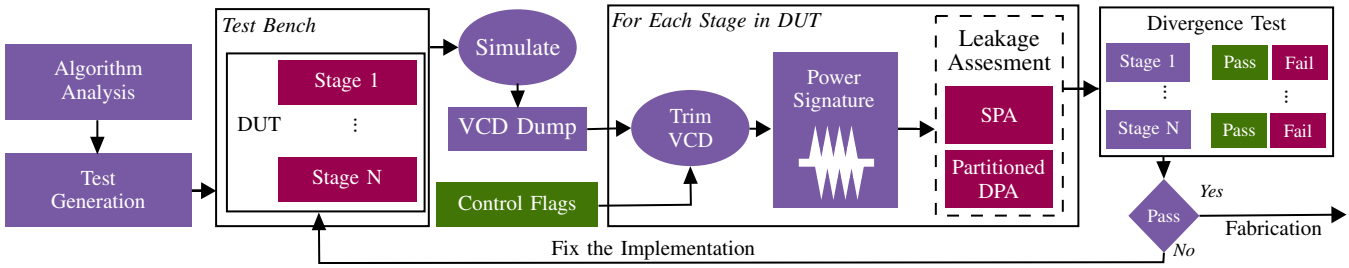


Fig. 2: Overview of TVLA* for leakage assessment of asymmetric key cryptography hardware implementations.

TABLE I: Inputs for different ECC algorithms

Algorithm	Inputs				
	Nonce	Private Key	P(x, y, z)	Msg	Tag
ECIES encrypt	✓		✓	✓	
ECIES decrypt		✓			✓
ECDSA sign	✓	✓		✓	
ECDSA verify			✓	✓	

parameters for different algorithms, we need to consider each and every input parameter during the test generation.

The nonce is a major secrecy feature that we need to evaluate based on the attacks discussed in Section II-C. A potential place the leakage can happen with nonce is the scalar multiplication stage. EC MULTIPLY stage is a bit serialized algorithm over the nonce. Therefore, we focus on generating patterns with multiple blocks of '0's and '1's in the nonce. For this purpose, we use Algorithm 3. First, it generates a nonce with serialized block patterns of '0' and '1's and fills the rest of the requirements with random nonce values. The same algorithm can be used to generate private key pairs.

Algorithm 3 Generation of Nonce Pairs

Require: NonceSize d, NumPairs N
Ensure: NoncePairs $\{\{n_1^1, n_1^2\}..\{n_N^1, n_N^2\}\}$
1: $tests = [\{zeroBin(d), oneBin(d)\}]$
2: **for** $x \in [2^i$ for $i = \log_2 d; i > 0; i$- -] **do**
3: $n2 \leftarrow \emptyset$
4: **while** $len(n2) < d$ **do**
5: $n2 \leftarrow n2 + zeroBin(x) + oneBin(x)$
6: **end while**
7: $tests.append(\{oneBin(d), n2[0:d]\})$
8: **end for**
9: **for** $y \in (N - len(tests))$ **do**
10: $tests.append(\{oneBin(d), randBin(d)\})$
11: **end for**
12: Return $tests$

When providing inputs for the public key, point coordinates $P(x, y, z)$ should be provided. In this case, the points should be valid points on the curve otherwise the algorithm ends up in an undefined state. For this, we generate public keys by solving the polynomial related to the curves identified in Section III-A. Next, we generate multiple random plaintext messages. Finally, we combine each of these parameters into the test vector following the steps outlined in Algorithm 4. First, we iterate through all the generated keys by solving the polynomial. For each key, we generate a random plaintext message. Next, we generate a test vector pair such that the first and second nonce values (generated by Algorithm 3) append to the first and second tests of the test pair. If we have X public keys, Algorithm 3 will produce a total of $X \times N$ pairs of test vectors. In order to obtain more accurate results, we need to synthesize the design. Finally, we create a testbench to simulate the implementation with generated input test vectors.

Algorithm 4 Generation of Test Vector Pairs

Require: PubKeys $\{p_1, .., p_x\}$, NoncePairs $\{\{n_1^1, n_1^2\}..\{n_y^1, n_y^2\}\}$
Ensure: TestVectorPairs $\{\{t_1^1, t_1^2\}, .., \{t_N^1, t_N^2\}\}$
1: $vectors = [\,]$
2: **for** $x \in \{p_1, .., p_x\}$ **do**
3: $pK \leftarrow x$
4: $msg \leftarrow randBin(len(msg))$
5: **for** $\{y_1, y_2\} \in \{\{n_1^1, n_1^2\}, .., \{n_y^1, n_y^2\}\}$ **do**
6: $t1 \leftarrow \{pK, msg, y_1\}$
7: $t2 \leftarrow \{pK, msg, y_2\}$
8: $vectors.append(\{t1, t2\})$
9: **end for**
10: **end for**
11: Return $vectors$

C. Simulation for Generation of Power Signatures

We simulate the testbench obtained in the previous step to generate the Value Change Dump (VCD). In this section, we discuss the process of generating the power signature from the obtained VCD data that corresponds to one test vector. Generally, side-channel footprints related to the power of hardware designs are correlated with the following factors:

- *Switching Activity* of the internal signals of the device. Here transition of a signal from $0 \rightarrow 1$ and $1 \rightarrow 0$ are considered to consume more power and emanate more electromagnetic radiation compared to $0 \rightarrow 0$ and $1 \rightarrow 1$.
- *Hamming Weight* power model correlates the number of signals that are either in value 0 or 1 in an instance to the overall power consumption of the device at that point.

In order to identify the independent stages outlined in Section III-A, we monitor the control flag signals of the implementation. In this way, we can perform the leakage assessment in each stage of the implementation separately. Figure 4 shows the power signature obtained during ECIES encryption divided into seven different stages with different colors. It is clear that ECC-related calculations, such as MULTIPLY (with Double-and-Add) and coordinate conversion (CORD.CONV), occupy the vast majority of the computation. The next two sections perform the leakage assessment on the generated power signatures for each stage of the implementation separately. Our utilization of control flags leads to automatic power signature alignment for the leakage assessment.

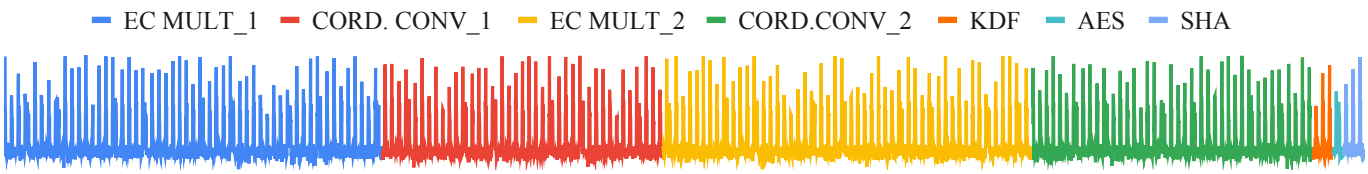
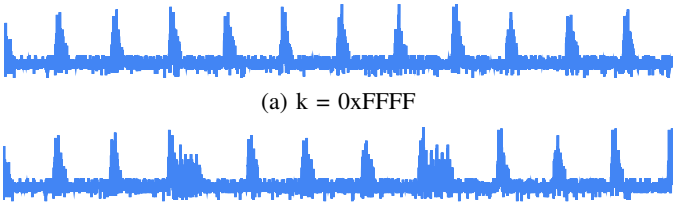


Fig. 4: Switching Activity time-series diagram for ECIES algorithm separated by different colors for each internal stage.

D. Leakage Assessment using Simple Power Analysis (SPA)

Simple Power Analysis (SPA) analyzes execution traces without any pre-processing. SPA can reveal information about the device's internal states, algorithm structure, and input-dependent power variations. Since algorithms are known, attackers can infer the idea about internal operations and secret data by analyzing a single trace or pair of traces. Implementations that appear to be safe against SPA should be further evaluated with differential power analysis.

Example: Let us consider the scalar multiplication, Double-and-Add, illustrated in Algorithm 2. If we analyze the steps involved in the algorithm, it can be observed that the operations performed over the bit value of the scalar (k) is different for *bit = 0* and *bit = 1*. Without having any prior knowledge about the nonce, by looking at the power traces we should be able to see multiple different power levels consumed by the device during the operations. Figure 5 illustrates the two power signatures constructed for an implementation using Algorithm 2. For two key values of k = 0xF0F0F and k = 0xFFFF, it shows a significant difference in the power peaks, which makes the implementation fail the SPA test.



(a) k = 0xFFFF

(b) k = 0xF0F0F

Fig. 5: SPA for Double-and-add Verilog implementation.

E. Leakage Assessment using Differential Power Analysis

Differential Power Analysis (DPA) utilizes statistical-based techniques to identify data-dependent correlations. As the name suggests, DPA requires more than one trace to perform the comparison, and hence we generated test vector pairs in Section III-B. As discussed in Section II-E, for the same stage of the algorithm, two power signature traces can be of different lengths due to the variation of execution time with the inputs. In order to address the problem of variable finish time of algorithms, we resize the traces into the same length with interpolated transformation. To preserve timing information for statistical analysis, we partition both traces into $\boldsymbol{C}$ equal sizes and perform differential analysis on each part separately. This preserves the timing information in the traces across the partitions and generalizes the evaluation technique to all the algorithms. Next, we apply the statistical *Welch t-test* method on each partition to evaluate two partitions of the power traces to assess their differences.

Let us consider a pair of traces $T(v_1^i, S_k)$ and $T(v_2^i, S_k)$, collected over stage S_k for the input test vector pair $V^i = \{v_1^i, v_2^i\}$. Let $n_{v_j^i}$, $\mu_{v_j^i}$, and $s^2_{v_j^i}$ be the size, mean, and variance of the x^{th} partition of trace $T(v_j^i, S_k)$, then *Welch t-test* t for $T(v_1^i, S_k)^x$ and $T(v_2^i, S_k)^x$ trace partitions can be computed by Equation 1 (with a corresponding p value in Equation 2).

$$t = \frac{\mu_{v_1^i} - \mu_{v_2^i}}{\sqrt{\frac{s^2_{V_1^x}}{n_{v_1^i}} + \frac{s^2_{V_2^x}}{n_{v_2^i}}}} \quad (1) \qquad p = 2\int_{|t|}^{\infty} f(t,d)dt \quad (2)$$

where d = degree of freedom

For the t-test, we make the *null hypothesis* as $T(v_1^i, S_k)$ and $T(v_2^i, S_k)$ traces are drawn from the same population, and hence, they are not distinguishable with a significance level of α'. If the condition $p < \alpha'$ satisfies the two trace partitions, we can reject the null hypothesis. After C independent tests for the significance level of α' in each partition, the final probability to reject the null hypothesis becomes the product of individual probabilities $(1-\alpha')^C$. Note that we need to maintain a confidence level of $\alpha = 0.05$ for entire trace width of $T(v_1^i, S_k)$ and $T(v_2^i, S_k)$ [24]. Therefore, confidence α' for each partition can be calculated with Bonferroni correction [25] as in Equation 3.

$$\begin{aligned} (1-\alpha')^C &= (1-\alpha) \\ \alpha' &= 1-(1-\alpha)^{\frac{1}{C}} \\ \alpha' &= \frac{\alpha}{C} \end{aligned} \quad (3)$$

This results in a partition-wise significance level of $\alpha' = \frac{0.05}{C}$. We can execute family-wise rejection if any partition of traces has $p < \frac{0.05}{C}$ and classify the considered stage of the implementation as "Failed".

The use of statistical methods comes with the risk of the miss-classification of results. Two miss-classifications and how the proposed TVLA* technique handles them are as follows:

Type 1 error (False positives): To reduce Type 1 error, we need to have a lower significance level (α) and improved statistical resolution. We perform "Partitioned DPA" analysis to increase the statistical resolution to include timing-related data, and then perform Bonferroni correction to reduce the significance level.

Type 2 error (False negatives): To reduce Type 2 error, we need to capture more sample data and conduct multiple experiments. Compared to TVLA for symmetric cryptography, the chances of type 2 errors are much less in asymmetric cryptography due to two reasons. 1) *Large trace sample size*: As illustrated in Figure 4, stages related to asymmetric key operations are in the order of 10000 (Montgomery multiplication takes 34563 cycles for 192-bit key) while symmetric key operations are in the order of 100 (tinyAES takes 21 cycles for 256-bit key). 2) *Large number of experiments*: Multiple input combinations in Algorithm 4 increase the number of experiments. We combine the nonce pair with different public keys and random messages to increase the number of different scenarios in the experiments. This reduces the Type 2 error.

F. Optimal Partitioning of Traces

The objective of partitioning is to have a good statistical resolution over the long trace data collected from different stages. If we have too few partitions, it will lead to more Type 1 errors while too many partitions will cause more Type 2 errors. Since these longer trace data is over bit serialized algorithms, we dynamically adjust the partition size based on the inputs to the algorithm. This preserves the statistical power of the experiment and provides a more accurate DPA analysis.

Algorithm 5 Trace Partitioning

Require: TestPair $\{t_i^1, t_i^2\}$
Ensure: Partitions $\{c_1, c_2, ..., c_n\}$
1: $t_{XOR} = t_i^1 \oplus t_i^2$
2: $C = \{\ \}$, $prev = 1$, $U = \frac{TraceLength}{NonceLength}$
3: **for** $bit \in t_{XOR}$ **do**
4: **if** $bit = 1$ **then**
5: **if** $prev = 0$ **then**
6: $C.append(ptr)$
7: **end if**
8: $prev = 1$
9: $ptr = ptr + U$
10: $C.append(ptr)$
11: **else**
12: $prev = 0$
13: $ptr = ptr + U$
14: **end if**
15: **end for**
16: Return C

Algorithm 5 presents the steps involved in the dynamic partitioning process. First, we perform an XOR operation on the input values. We partition XOR result such that all consecutive zeros become one partition while each ones separately partitioned into different partitions. For this purpose, we calculate the unit length to process a single bit by taking the ratio of the trace length to the nonce length. Then each partition point is generated with Algorithm 5. Figure 7 illustrates an example scenario for calculating partitions dynamically. After the dynamic partitioning, the DPA technique is applied to each partition separately and proceeds with the steps discussed in Section III-E.

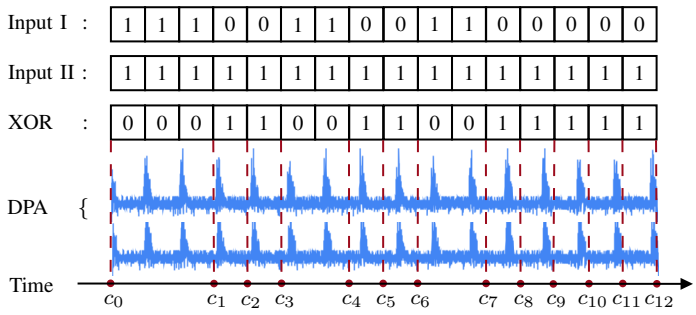


Fig. 7: An example scenario for dynamic partitioning of traces based on the bitwise difference of the input nonce.

Complexity Analysis: Let the size of two input distributions used for the differential analysis be M (after scaling to the same length). If we do not partition (traditional TVLA), the time complexity of applying the Welch t-test is $\mathcal{O}(M)$ since there are $\mathcal{O}(M)$ elements in the two distributions. Partitioning the traces with input XOR operation can be performed in a constant time ($\mathcal{O}(1)$). Let n ($1 \leq n \leq key\_size$) be the number of partitions and m_i be the number of elements in each partition such that $n \times \sum_{i=1}^{n} m_i = M$. Therefore, the time complexity of applying Welch t-test on all the partitions is also bounded by $\mathcal{O}(M)$ since the total number of elements in the two distributions did not change. The last step of the algorithm (family-wise rejection) can be performed in a constant time of $\mathcal{O}(1)$. Therefore, the partitioned DPA (TVLA*) does not increase the complexity compared to traditional TVLA.

G. Classification using Divergence Test

The leakage analysis discussed in the previous section analyzes a pair of traces for only one stage in the algorithm. In this section, we first discuss how to classify each stage as side-channel vulnerable or safe. Next, we make a decision on the entire implementation. In Section III-B, we generated test vector pairs, which results in $X \times N$ trace pairs. We perform the Welch t-test followed by Bonferroni correction for each stage of implementation and classify each stage as "Pass" or "Fail". If a particular stage of the implementation "Fail" during DPA, the implementation is classified as "Fail". We classify the implementation as "Pass" if and only if all the stages of the implementation pass the divergence test.

H. Fixing the TVLA* Failing Implementations

If a design fails the divergence test, the design should be fixed against the side channel leakage. This step will involve different techniques in different instances to prevent information leakage. For different stages, there can be different implementation improvements. For example, a popular mitigation is to use the Montgomery ladder for scalar multiplication over double and add since it severely reduces the information leakage through timing. Other mitigations involve randomizing some value being used in the computation to blind it.

The authors of [20] proposed several mitigation techniques against horizontal collisions. These mitigations specifically target the modular multiplication of two field values, which are integers. Since these field values can be represented in word form, multiplication takes the form of a matrix. A representation of the matrix is given below, where a and b are the two field values being multiplied together and the entry number corresponds to its word location.

$$\begin{bmatrix} a_1b_1 & \dots & a_1b_n \\ \vdots & \ddots & \vdots \\ a_nb_1 & \dots & a_nb_n \end{bmatrix}$$

Operands Blinding: Blind each operand with a random value. After multiplication, the resulting blinded value is equivalent to the result of the non-blinded value. This mitigation reduces the efficiency of the horizontal collision correlation analysis attack, however, it does not remove all sources of the leakage. Blinding is memory efficient since it will not take additional memory, however, an additional operation is added for each value that needs blinding.

Shuffling Rows and Columns: Shuffling leads to different permutations of row and column configurations. However, shuffling the rows and columns does not have any impact on the computed values of the matrix entries. It does add to the computational time that the attacker needs to perform the attack by the cost of searching for a permutation. This search is $O(n)$. Additional memory may be required while swapping values.

Shuffling and Blinding: This mitigation combines ideas from the two previous mitigations together. It achieves this by first blinding one value, shuffling the rows with permutations, blinding the second value, and then shuffling the columns with permutations. This technique does prevent the attack but allows for a zero-value attack to be possible that was previously not.

Global Shuffling: This is a variation of Shuffling and Blinding where the permutations of the rows and columns are done at the same time, instead of sequentially. Therefore, the attacker needs to now search through a permutation of size n^2 instead of two permutations of size n, which is computationally more expensive. Shuffling both the rows and columns at the same time will take more memory than if done sequentially. While the authors note that this does prevent the attack, more research is needed to demonstrate the efficiency of the implementation in using this mitigation.

IV. Experiments

In this section, we first describe the experimental setup. Next, we present results for side-channel leakage assessment.

A. Experimental Setup

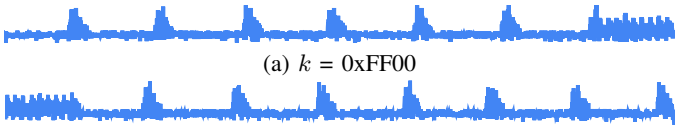
We have implemented ECDSA and ECIES in Verilog with the algorithms outlined in [22] and [23] including three different EC MULTIPLY algorithms of Double-and-Add, BinaryNAF, and Montgomery. Where appropriate, these algorithms used open-sourced Verilog implementations for SHA and AES. SEC's SHA1 implementation in Verilog was used for hashing operations. tinyAES was used in ECIES for AES encryption and decryption. For the key derivation function in ECIES, the standard ANSI X9.63 was used. For the creation and authentication of tags, HMAC-SHA1 was used, with the SHA1 functionality being provided by SEC's SHA1 module. After testing the implementation with NIST prime field curves, we instrumented the design with the steps discussed in Section III-B. We used *Synopsys Design_Compiler* with *SAED90nm* CMOS technology for the synthesis of the design. We used *Synopsys VCS* for the simulations of the designs and to obtain the signal dumps. For test generation, power signature construction, and leakage assessment, we created appropriate scripts with Python. For partitioned DPA, we dynamically partitioned the traces based on the input nonce with the algorithm proposed in Section III-F.

B. Leakage Assessment of Asymmetric Key Algorithms

We evaluate ECIES and ECDSA implementations using existing and proposed (TVLA*) methods. To generate private keys or nonce (as appropriate), we used Algorithm 3. All other steps in TVLA* remain the same for the rest of the evaluation. Since direct evaluation with [1]–[3] is not possible due to input test generation difference, we compare the results with the standard Welch t-test. Table II presents the final evaluation performed on different modes of ECC implementations with the standard Welch t-test and the proposed TVLA* methodology. Each column shows the percentage of experiments "failed" the DPA analysis out of 1000 experiments. The green color cells represent true positive and true negative results while the brown color cells represent false positive and false negative results. It can be observed that the Binary NAF algorithm has been subjected to type 1 error due to not having timing-related information on the standard Welch t-test. The tiny_AES implementation was also detected as side-channel vulnerable with TVLA*, which is consistent with [2].

C. Effect of Partitioned t-test

We created this experiment to demonstrate the effectiveness of partitioned DPA (preserved the timing information of the traces) over standard divergence measuring techniques. We have taken a test case that fails the SPA. For this, we used the EC MULTIPLY algorithm Double-and-Add with two nonce values of "0xFF00" and "0x00FF". As illustrated in Figure 8, power traces are inverted over the time axis and hence divergence test should fail.



(a) k = 0xFF00

(b) k = 0x00FF

Fig. 8: SPA for Double-and-add two *nonce (k)* values.

As illustrated in Table III, standard Welch t-test and KL divergence do not take timing information into account and hence provided false positive results. Our proposed partition-based differential power analysis distinguishes the two traces since the analysis is done on multiple different partitions (16 partitions in this specific example).

TABLE III: Divergence test on traces of 0xFF00 and 0x00FF

Method	Welch t-test	KL divergence	TVLA*
Result	False Positive	False Positive	True Negative

D. Dynamic Partitioning of Trace Data

We have created a separate experiment to demonstrate the effectiveness of dynamic partitioning. For this purpose, we generated 1000 trace data pairs with the scalar multiplication implementation with the Montgomery algorithm (with shuffling and blinding mitigation). Then we introduced modifications to the Montgomery implementation to have a subtle imbalance in the switching activity over the nonce multiplication and generated a separate data set that consists of 1000 trace data pairs. These two data sets serve as the known labeled data set for the evaluation of the results of the experiment.

In the next step, we conducted experiments with both data sets with different partition sizes ranging from 1 to nonce size

TABLE II: Evaluation on ECC Verilog implementations with Welch t-test and partitioned Welch t-test in TVLA*

Implementation	Welch t-test in TVLA						Partitioned Welch t-test in TVLA*					
	EC MULTIPLY			CORD. CONV	SHA1	tiny AES	EC MULTIPLY			CORD. CONV	SHA1	tiny AES
	D. Add	B. NAF	Mont				D. Add	B. NAF	Mont.			
ECIES encryption	4.6%	0%	0%	0%	0%	0.012%	18.9%	19.2%	0%	0%	0%	0.012%
ECIES decryption	6.2%	0%	0%	0%	0%	0.008%	17.6%	21.1%	0%	0%	0%	0.008%
ECDSA sign	5.9%	0%	0%	0%	0%		20.9%	18.3%	0%	0%	0%	
ECDSA verify	4.2%	0%	0%	0%	0%		16.7%	16.9%	0%	0%	0%	

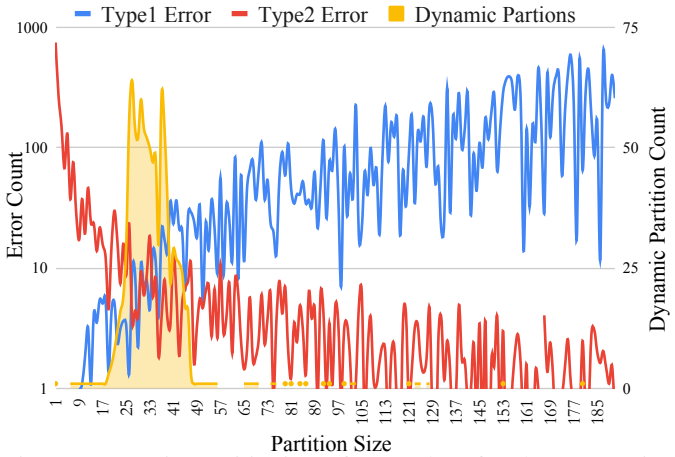


Fig. 9: Dynamic partitioning of trace data for the nonce size of 192 on Montgomery scalar multiplication algorithm.

(in this case 192). Next, we deployed dynamic partitioning on both data sets to observe the number of partitions that resulted in the comparisons. Figure 9 presents the relationship between Type1 error and Type2 error with the number of partitions in the trace data. The yellow curve represents the number of partitions generated by the dynamic partitioning algorithm proposed in Section III-F which sums up to the total number of test cases (1000). This illustrated that the proposed dynamic partitioning technique selects the best partition for the statistical comparison.

TABLE IV: Minimum p-values observed with Montgomery multiplication algorithm with different mitigation techniques with the experiment conducted for four iterations each containing 1000 test pairs. Brown-colored cells indicate rejected implementations from the divergence test

#	Without Mitigation	Blinding	Shuffling	Shuffling and Blinding	Global Shuffling
1	0.0042	0.0258	0.1842	0.4947	0.6854
2	0.0059	0.0009	0.2874	0.7398	0.7458
3	0.0048	0.0085	0.1879	0.5009	0.5748
4	0.0106	0.0147	0.2935	0.8456	0.6875

E. TVLA on Different EC MULTIPLY Algorithms*

As discussed in Section II-C, EC MULTIPLY is the victim for most of the side-channel attacks. Therefore, we evaluated the three algorithms that we have implemented in Verilog. BinaryNAF and Double-and-Add should fail the experiment since these algorithms contain imbalanced finite-state machine (FSM) operations that depend on inputs. For the Montgomery algorithm, we have implemented multiple variations with and without mitigation techniques discussed in Section III-H and applied TVLA*. Table IV presents the results of the experiment with minimum observed p-value in each experiment with different mitigation techniques. Brown-colored cells indicate rejected implementations from the divergence test while green-colored cells represent implementations that are classified as safe against side channel leakage by TVLA* framework.

Figure 10 presents the minimum p-value (y axis in log scale) observed among each partition against the generated test vectors with three algorithms of Montgomery (with blinding and shuffling), Binary NAF, and Double-and-Add. Here α' represented the minimum p-value to accept the *null hypothesis*, which is calculated with dynamic partitioning ($\alpha' = \frac{0.05}{C}$, where C is the number of partitions). As expected, it can be observed that for Binary NAF and Double-and-Add, TVLA* has rejected the *null hypothesis* with the confidence of 99.999% (with $p < \alpha$).

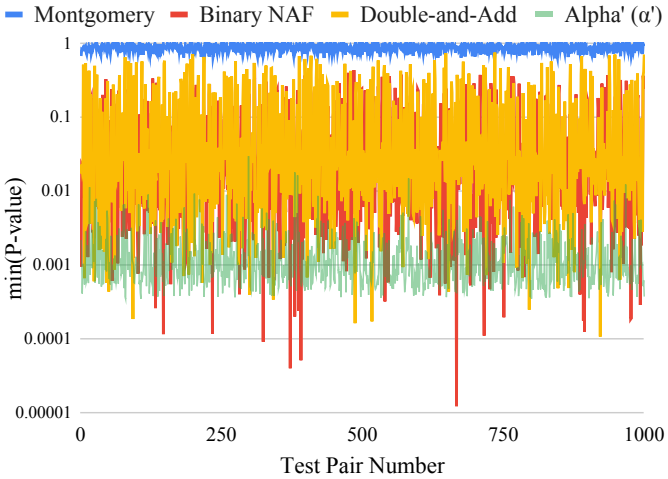


Fig. 10: Minimum p-value observed for partitioned Welch t-test DPA analysis for 1000 pairs of input vectors over different scalar multiplication algorithms.

TABLE V: Divergence test on different EC Multiply Algorithms for curve P-192, 192 bit key size

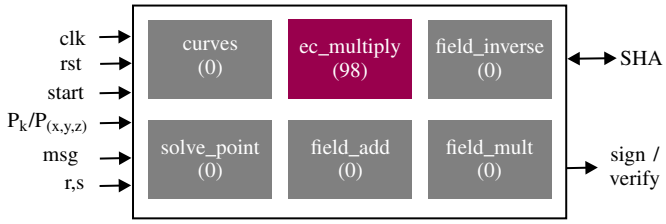
EC MULTIPLY		TVLA*		
Algorithm	# Clk Cycles	SPA	min(p)	Divergence Test
Double-and-Add	20606	Fail	0.00010	Fail
Binary NAF	3837	Pass	0.00006	Fail
Montgomery	34563	Pass	0.61739	Pass

Table V illustrates the divergence test results for the three implementations. TVLA* methodology classifies Double-and-ADD and Binary NAF implementations as side-channel vulnerable, while the Montgomery implementation is classified as side-channel resistant.

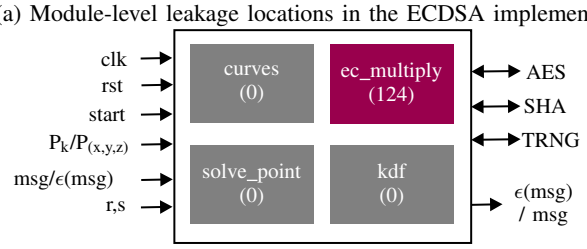
F. Physical Leakage Locations in ECDSA and ECIES

TVLA* framework analyzes the power usage of a cryptographic implementation at the pre-silicon stage. It divides

implementation into stages by analyzing the control flags. Therefore, the evaluation (detection) results provide us the information about all the stages. A designer is mostly interested in the stages (components) that failed TVLA analysis, which would be a potential leakage location in the fabricated chip. Figure 11a illustrates the leakage locations in the ECDSA implementation. In this evaluation, ec_multiply module failed for 98 partitioned-DPA tests. None of the other ECDSA submodules failed during the evaluation consisting of 3000 tests.



(a) Module-level leakage locations in the ECDSA implementation



(b) Module-level leakage locations in the ECIES implementation

Fig. 11: Leakage locations (highlighted in ■) and the corresponding number of failed tests (in brackets) based on partitioned-DPA analysis on ECDSA and ECIES Verilog implementations during TVLA* evaluation.

Figure 11b illustrates the leakage locations in the ECIES implementation. As we can see from the figure, ec_multiply module failed for 124 partitioned-DPA tests. None of the other ECIES modules failed during the evaluation consisting of 3000 tests. It can be observed from Figure 11 that the scalar multiplication module (ec_multiply) is the most leaky location in both ECIES and ECDSA implementations.

G. Testing TVLA* on Bare Metal ECC Implementation

In this experiment, we highlight the possibility of using the proposed TVLA* technique on firmware-level implementations of public key cryptography modules on embedded systems. For this process, we have used the OpenRiscV32 [26] processor implemented in Verilog as the host processor for the embedded system. Next, we implemented ECC cryptography modules in C and compiled them with the RISC-V toolchain. Figure 12 illustrates the module level block diagram of the hardware setup. Then we evaluated different scalar multiplication algorithms against their side-channel leakage.

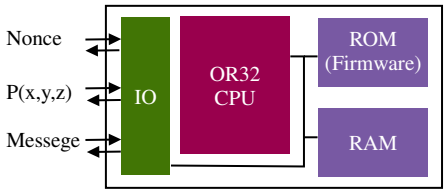


Fig. 12: Firmware-level evaluation of ECC implementation.

Figure 13 illustrates the minimum p-values observed on firmware level implementation of scalar multiplication algorithms of Montgomery (with shuffling and blinding), Binary NAF and Double-and-Add. Here α' represents the minimum p-value to accept the *null hypothesis*, which is calculated with dynamic partitioning ($\alpha' = \frac{0.05}{C}$, where C is the number of partitions). As expected, it can be observed that for firmware implementations of Binary NAF and Double-and-Add, TVLA* has rejected the *null hypothesis* with the confidence of 99.999% (with $p < \alpha$).

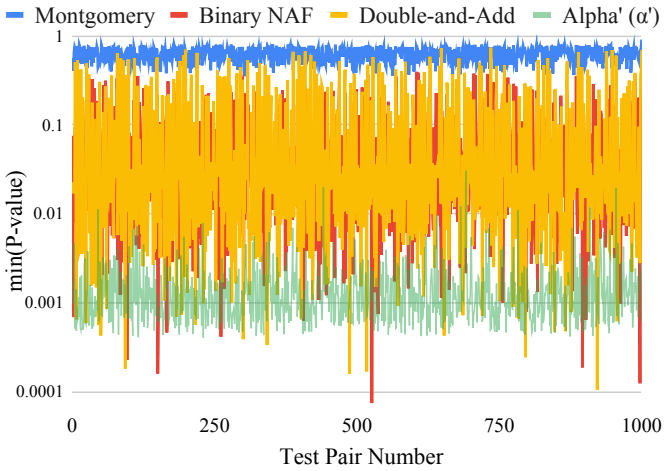


Fig. 13: Minimum p-value for partitioned Welch t-test DPA analysis on firmware-level ECC implementation for 1000 pairs of input vectors over various scalar multiplication algorithms.

H. Applicability and Limitations

In this paper, we focused on the Elliptic Curve Cryptography (ECC) family of algorithms due to their improved security, and performance. Further, the implementation of ECC-based hardware algorithms can be considered more complex compared to the other asymmetric key cryptography algorithms. However, the applicability of TVLA* is not limited to asymmetric cryptosystems based on the ECC family. In order to demonstrate the applicability of TVLA* on other asymmetric cryptography algorithms, we first discuss three popular RSA implementations based on three different algorithms: Chinese Remainder Theorem (CRT), Montgomery Multiplication, and Square-and-Multiply method. Then we evaluate three designs of RSA that implement the above algorithms with TVLA*. Finally, we discuss the applicability of TVLA* on hybrid cryptosystems.

RSA with CRT: CRT optimization is applied during the decryption process to speed up the modular exponentiation. Instead of performing a single modular exponentiation operation using the private exponent, the CRT method breaks it down into multiple smaller modular exponentiation using the prime factors of the modulus. This smaller modular exponentiation can be computed separately and combined using the CRT formulas. However, the CRT method can introduce vulnerabilities to power side-channel attacks. For example, the power consumption during the modular exponentiation steps might

vary depending on the value of the corresponding prime factor. An attacker can exploit these variations to extract information about the secret key.

RSA with Sqaure-and-Multiply: In the Square-and-Multiply algorithm, the exponent is typically represented in binary form, and the algorithm performs repeated squaring and multiplications based on the bits of the exponent. These operations can result in different power consumption patterns depending on the value of each bit.

RSA with Montgomery Multiplication: In the case of Montgomery multiplication, the power consumption patterns can vary depending on the intermediate values and operations performed during the algorithm. For example, the number of shifts and additions involved in Montgomery multiplication can introduce variations in power consumption. An attacker can analyze these power consumption patterns to potentially deduce information about the secret key or other sensitive data.

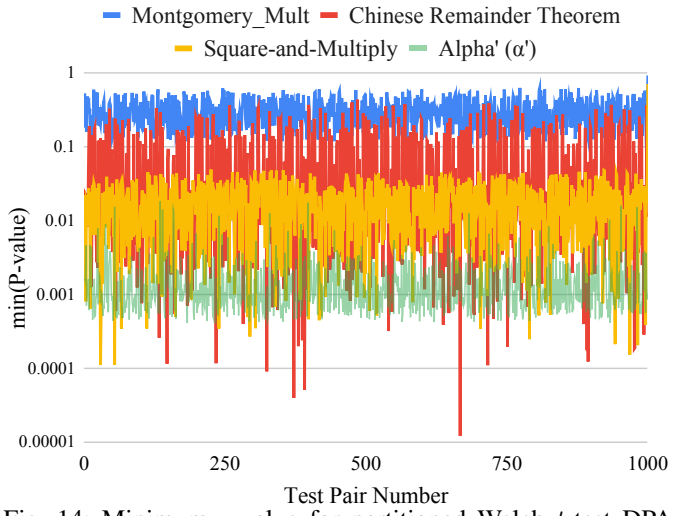


Fig. 14: Minimum p-value for partitioned Welch t-test DPA analysis on RSA implementation for 1000 pairs of input vectors over various RSA implementation algorithms.

TVLA on RSA:* We have used three different Verilog implementations of RSA with three different algorithms (Chinese Remainder Theorem, Square-and-Multiply, and Montgomery Multiplication with shuffling and blinding mitigation). Figure 14 illustrates the minimum p-values observed after performing TVLA* methodology. As expected, power side-channel vulnerability of mitigated RSA implementation based on the Montgomery Multiplication algorithm is classified by TVLA* as safe against side-channel leakage while implementations based on the Chinese Remainder Theorem and Square-and-Multiply are classified as susceptible to power side-channel leakage.

TVLA on Hybrid Cryptosystems:* TVLA* enables the evaluation of hybrid cryptosystems which combines both asymmetric and symmetric components in the implementations. Since the components are evaluated separately, asymmetric components of the system can be evaluated with TVLA*, while symmetric components can be evaluated with existing symmetric TVLA [1], [2] techniques. The composition of results is trivial since we consider it a system failure if any of the components fail.

V. Conclusion

In this paper, we proposed a test vector leakage assessment (TVLA) technique for asymmetric key algorithms. We analyzed the applicability of existing TVLA techniques on asymmetric algorithms and identified the fundamental limitations. We proposed a systematic test generation technique to generate valid test cases that can maximize the switching difference in side-channel vulnerable implementations. We developed a differential power analysis technique for asymmetric key cryptography algorithms. Specifically, we presented a partition-based t-test evaluation technique to evaluate with higher statistical accuracy while preserving timing information over the traces. Experimental evaluation on diverse elliptic curve cryptography algorithms demonstrated that our proposed technique has better accuracy than statistical techniques used in TVLA for symmetric key cryptography algorithms.

Acknowledgments

This work was partially supported by the National Science Foundation grant CCF-1908131.

References

[1] M. He, J. Park, A. Nahiyan, A. Vassilev, Y. Jin, and M. Tehranipoor, "RTL-PSC: Automated power side-channel leakage assessment at register-transfer level," in *IEEE VLSI Test Symposium (VTS)*, 2019, pp. 1–6.

[2] N. Pundir, J. Park, F. Farahmandi, and M. Tehranipoor, "Power side-channel leakage assessment framework at register-transfer level," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2022.

[3] T. Zhang, J. Park, M. Tehranipoor, and F. Farahmandi, "PSC-TG: RTL power side-channel leakage assessment with test pattern generation," in *ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 709–714.

[4] J. Silverman, *The arithmetic of elliptic curves*. Springer, 2009, vol. 106.

[5] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[6] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[7] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and rsa on 8-bit cpus," in *Cryptographic Hardware and Embedded Systems*, 2004, pp. 119–132.

[8] M. Adalier and A. Teknik, "Efficient and secure elliptic curve cryptography implementation of curve p-256," in *Workshop on elliptic curve cryptography standards*, vol. 66, no. 446, 2015, pp. 2014–2017.

[9] D. F. Aranha, F. R. Novaes, A. Takahashi, M. Tibouchi, and Y. Yarom, "Ladderleak: Breaking ecdsa with less than one bit of nonce leakage," in *ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 225–242.

[10] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Advances in Cryptology—CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings 16*. Springer, 1996, pp. 104–113.

[11] J.-F. Dhem, F. Koeune, P.-A. Leroux, P. Mestré, J.-J. Quisquater, and J.-L. Willems, "A practical implementation of the timing attack," in *International Conference on Smart Card Research and Advanced Applications*. Springer, 1998, pp. 167–182.

[12] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," *Proceedings of the IEEE*, vol. 100, no. 11, pp. 3056–3076, 2012.

[13] A. Barenghi, G. Bertoni, A. Palomba, and R. Susella, "A novel fault attack against ecdsa," in *2011 IEEE International Symposium on Hardware-Oriented Security and Trust*. IEEE, 2011, pp. 161–166.

[14] Y. Yarom and N. Benger, "Recovering openssl ecdsa nonces using the flush+ reload cache side-channel attack," *Cryptology ePrint Archive*, 2014.

[15] S. Bhattacharya, C. Maurice, S. Bhasin, and D. Mukhopadhyay, "Branch prediction attack on blinded scalar multiplication," *IEEE Transactions on Computers*, vol. 69, no. 5, pp. 633–648, 2019.

[16] J. Jancar, V. Sedlacek, P. Svenda, and M. Sys, "Minerva: The curse of ecdsa nonces: Systematic analysis of lattice attacks on noisy leakage of bit-length of ecdsa nonces," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 281–308, 2020.

[17] D. Naccache, N. P. Smart, and J. Stern, "Projective coordinates leak," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2004, pp. 257–267.

[18] D. Maimuţ, C. Murdica, D. Naccache, and M. Tibouchi, "Fault attacks on projective-to-affine coordinates conversion," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2013, pp. 46–61.

[19] A. C. Aldaya, C. P. García, and B. B. Brumley, "From a to z: Projective coordinates leakage in the wild," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 428–453, 2020.

[20] A. Bauer, É. Jaulmes, E. Prouff, J.-R. Reinhard, and J. Wild, "Horizontal collision correlation attack on elliptic curves," *Cryptography and Communications*, vol. 7, no. 1, pp. 91–119, 2015.

[21] M. Tunstall and G. Goodwill, "Applying tvla to public key cryptographic algorithms," *Cryptology ePrint Archive*, 2016.

[22] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International journal of information security*, vol. 1, no. 1, pp. 36–63, 2001.

[23] D. R. Brown, "Sec 1: Elliptic curve cryptography," *Certicom Research*, vol. 2, 2009.

[24] T. Schneider and A. Moradi, "Leakage assessment methodology," in *Cryptographic Hardware and Embedded Systems*, 2015, pp. 495–513.

[25] E. Weisstein, "Bonferroni correction," *https://mathworld.wolfram.com/*, 2004.

[26] YosysHQ, "Yosyshq/picorv32: Picorv32 - a size-optimized risc-v cpu." [Online]. Available: https://github.com/YosysHQ/picorv32

Aruna Jayasena is a Ph.D student in the Department of Computer & Information Science & Engineering at the University of Florida. He received his B.S. in the Department of Computer Science and Engineering at the University of Moratuwa, Sri Lanka, in 2019. His research focuses on systems security, test generation, trusted execution, side-channel analysis, and system-on-chip debug.

Emma Andrews is a Ph.D student in the Department of Computer & Information Science & Engineering at the University of Florida. She received her B.S. and M.S. in the Department of Computer & Information Science & Engineering at the University of Florida in 2021 and 2023, respectively. Her research focuses on hardware security and trustworthy cryptography.

Prabhat Mishra is a Professor in the Department of Computer and Information Science and Engineering and a UF Research Foundation Professor at the University of Florida. He received his Ph.D. in Computer Science from the University of California at Irvine in 2004. His research interests include embedded systems, design automation, hardware security, energy-aware computing, formal verification, system-on-chip validation, machine learning, and quantum computing. He currently serves as an Associate Editor of IEEE Transactions on VLSI Systems and ACM Transactions on Embedded Computing Systems. He is an IEEE Fellow, an AAAS Fellow, and an ACM Distinguished Scientist.