IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2021, No. 4, pp. 1–26. DOI:10.46586/tches.v2021.i4.1-26

Novel Key Recovery Attack on Secure ECDSA Implementation by Exploiting Collisions between Unknown Entries

Sunghyun Jin[1,2], Sangyub Lee[3], Sung Min Cho[4], HeeSeok Kim[5†] and Seokhie Hong[1,2]

[1] School of Cyber Security, Korea University, Seoul, Republic of Korea
[2] Center for Information Security Technologies, Institute of Cyber Security and Privacy, Korea University, Seoul, Republic of Korea,
{sunghyunjin,shhong}@korea.ac.kr
[3] National Institute for Mathematical Sciences, Daejeon, Republic of Korea,
sylee@nims.re.kr
[4] CIOT, Seongnam, Republic of Korea,
smcho@ciotsecurity.com
[5] Department of Cyber Security, College of Science and Technology, Korea University, Sejong, Republic of Korea,
80khs@korea.ac.kr

Abstract. In this paper, we propose a novel key recovery attack against secure ECDSA signature generation employing regular table-based scalar multiplication. Our attack exploits novel leakage, denoted by *collision information*, which can be constructed by iteratively determining whether two entries loaded from the table are the same or not through side-channel collision analysis. Without knowing the actual value of the table entries, an adversary can recover the private key of ECDSA by finding the condition for which several nonces are linearly dependent by exploiting only the collision information. We show that this condition can be satisfied practically with a reasonable number of digital signatures and corresponding traces. Furthermore, we also show that all entries in the pre-computation table can be recovered using the recovered private key and a sufficient number of digital signatures based on the collision information. As case studies, we find that fixed-base comb and T_SM scalar multiplication are vulnerable to our attack. Finally, we verify that our attack is a real threat by conducting an experiment with power consumption traces acquired during T_SM scalar multiplication operations on an ARM Cortex-M based microcontroller. We also provide the details for validation process.

Keywords: Public-key cryptography · Digital signature · ECDSA · Scalar multiplication · Side-channel analysis · Collision attack

1 Introduction

A digital signature has an important role as an authentication mechanism in modern security. The Elliptic Curve Digital Signature Algorithm (ECDSA) [KR13, RHAL92, JMV01] is an elliptic curve cryptography-based digital signature scheme that is used in a wide variety of security services. ECDSA has speed and memory usage advantages with a shorter

†Corresponding Author. This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT under Grant NRF-2019R1A2C2088960.

Received: 2021-04-15 Accepted: 2021-06-15 Published: 2021-08-11

key length while providing a level of security equivalent to that of RSA, a common representative public-key cryptosystem. Thus, it is preferred in constrained environments such as smart cards.

On the other hand, side-channel analysis (SCA) takes advantage of various forms of leakage (e.g., execution time, power consumption, electromagnetic emission, and acoustic emanation) occurring when cryptosystems are executed in devices to retrieve secret information because the instructions and data processed by the device are correlated with these leakages [Koc96, KJJ99, GMO01, GST14, MOP08]. Because it is well-known that SCA can be practically employed to break cryptosystems, many works on SCA against ECDSA have been reported to investigate its practical security.

In this paper, we propose a novel key recovery attack against secure ECDSA signature generation employing regular table-based scalar multiplication by exploiting side-channel collisions between unknown entries. Because ECDSA uses a fixed base, table-based scalar multiplication is commonly employed in ECDSA due to its efficiency and security. Although table-based scalar multiplication can be easily implemented to be practically secure against known SCAs, our attack can still be successful against it. Our attack exploits a form of leakage to determine whether two entries loaded from the table are the same or not, i.e., whether there is a collision or not between two corresponding traces. Without knowing the actual value of the table entries, an adversary can recover the private key for ECDSA exploiting only the collision information by finding the condition for which several nonces are linearly dependent. We show that this condition can be satisfied practically with a reasonable number of digital signatures and corresponding traces. The required number is at most one more than the total number of entries in the table. Furthermore, we also show that all entries of the pre-computation table can be recovered using the recovered private key and a sufficient number of digital signatures based on the collision information. We first explain the detail of our attack against ECDSA signature generation employing regular table-based scalar multiplication. We then show that fixed-base comb and T_SM scalar multiplication are vulnerable to our attack as case studies. Finally, we prove that our attack is a real threat by conducting an experiment with power consumption traces acquired during T_SM scalar multiplication operations on an ARM Cortex-M based microcontroller. We also provide details on how to conduct the validation process.

This paper is organized as follows. In Section 2, we briefly explain ECDSA signature generation and provide an overview of side-channel attacks on ECDSA. We also generalize regular table-based scalar multiplication for the description of our attack. In Section 3, we describe our novel key recover attack, which can recover the private key for ECDSA by identifying the situation in which several nonces are linearly dependent using collision information from the pre-computation table. Section 4 presents case studies on fixed-base comb and T_SM scalar multiplication. In Section 5, we validate the feasibility of the proposed attack by describing an experiment using real traces of T_SM scalar multiplication acquired from an ARM Cortex-M based microcontroller. Finally, we conclude this paper in Section 6.

2 Preliminaries

2.1 Notations

Let $t \in \mathbb{R}^{D\times 1}$ denote side-channel traces of length D. Let $x[j]$ denote the j-th entry of vector x. Let $\mathbb{Z}_a := \{x \ mod \ a | x \in \mathbb{Z}\}$ and $\mathbb{Z}_a^b := \{(z_1, z_2, ..., z_b) | z_1, z_2, ..., z_b \in \mathbb{Z}_a\}$. Let $A \parallel B$ be the concatenation of vectors (or matrices) A and B. Let $(\cdot)^T$ be a transpose of the vector (or matrix), and $Mat_{X\times Y}(Z)$ be the set of all X-by-Y matrices with all entries in Z. Let I_N is a $N \times N$ identity matrix.

2.2 Overview on Side-Channel Attacks against ECDSA Signature Generation

The ECDSA signature generation algorithm, described in Algorithm 1, generates a signature (r, s) for message m using private key d obtained from the key generation process. It consists of a scalar multiplication stage over an elliptic curve using a randomly selected secret nonce k (also known as the ephemeral key or ephemeral scalar), and a subsequent stage in which r and s are computed using the value obtained from the scalar multiplication.

Algorithm 1 ECDSA signature generation [HMV04]

```
Require: Message $m$, private key $d$, base point $P$ of order $ord$, hash function $\mathcal{H}$
Ensure: Signature $(r, s)$
1: Choose nonce $k$ uniformly at random from $[1, ord - 1]$
2: Compute scalar multiplication $Q = (x_Q, y_Q) = k \cdot P$
3: $r = x_Q \ mod\ ord$; If $r = 0$ then go to step 1
4: Compute hash $h = \mathcal{H}(m)$
5: $s = k^{-1} \cdot (h + d \cdot r) \ mod\ ord$
6: If $s = 0$ then go to step 1
7: Return $(r, s)$
```

Naive implementation of Algorithm 1 can be easily broken by side-channel attacks. Side-channel attacks against ECDSA signature generation can be categorized as multiple-trace attack (MTA) and single-trace attack (STA) depending on the required number of traces.

Representative of MTA are differential power analysis (DPA) [KJJ99] and correlation power analysis (CPA) [BCO04] against ECDSA signature generation [Cor99, AFV07, HMHW09]. An adversary can conduct a DPA-like attack against intermediate data from the multiplication operation $d \cdot r$, performed in line 5 of Algorithm 1, by partially guessing private key d because the value of r is known as a signature. The adversary can then test whether the guess is correct by evaluating the correlation between multiple traces and hypothetical values for the intermediate data with a statistical tool and consequently recover the secret d. Hence, to counter this attack, it is required to adjust the operations of $k^{-1} \cdot (h + d \cdot r)$ in line 5 of Algorithm 1 such that the inverse of random secret k is involved in every prior operation, i.e., $k^{-1} \cdot h + (k^{-1} \cdot d) \cdot r$, to prevent the guessing of the intermediate data.

Another form of MTA is lattice attacks using partial nonces. Partial information for nonces with signatures can be reconstructed as a system of inequalities for private key, which is known as the hidden number problem (HNP) [HGS01, NS02]. The system of inequalities is a closest vector problem (CVP) and can be solved efficiently using the LLL lattice basis reduction algorithm and Babai's nearest plane algorithm. For the lattice attack, partial nonces can be retrieved in the following ways. First, if biased nonces are generated in Algorithm 1, then partial nonces can be easily guessed [AFG$^+$14, BH19]. Second, if some part of Algorithm 1 is implemented to consume a non-constant execution time and has different times according to the message or nonce, then it can leak partial information [BH09, BT11, YB14, DHMP14, BvSY14, ABF$^+$15, vSY15, FWC16, GPP$^+$16, BFMT16, ASS17, GB17, Rya18, DPP20, RSBD20, ANT$^+$20, MH20, CPB20, JSSS20, GuHT$^+$20, MSEH20, WSBS20]. Third, it can be possible through template-based attacks [MO09, BCP$^+$14, BCP$^+$19] or STA. These attacks, which will be described later, can be used to find only partial information because errors may exist.

Here, we provide an overview of STA. Similar to MTA, the naive implementation of scalar multiplication allows simple power analysis (SPA) [KJJ99], which is a basic form of STA, because the patterns of doubling and addition in a single trace of scalar multiplication are different. This distinguishability is generally caused by two factors. First, differences in the implementation of doubling and addition operations, which results in different execution times and computation methods, are used. Second, an irregular scalar multiplication algorithm is employed, e.g., double-and-add scalar multiplication where point addition is computed depending on the bits of the scalar. Thus, there are two approaches to counteracting SPA.

The first approach is introducing regular scalar multiplication algorithms such as double-and-add always [Cor99] and the Montgomery ladder algorithm [IT02], which always employ an identical sequence of operations consisting of the same number of doubling and addition. The second approach is removing the difference in unit instructions for doubling and addition, such as side-channel atomicity [CCJ04] and unified point addition [BJ02]. Although these approaches are effective in counteracting SPA because they focus on removing the operational leakages of scalar multiplication algorithms, leakages from registers and data are still exploitable and STA using these leakages have been proposed [Wal01, HMH+12, CFG+12, BJPW14, HIM+14, DGH+16, HKT15, SHKS15, SH17].

Because register- or data-dependent leakages, i.e., collision characteristics, can be determined to be highly correlated or not depending on the bit value of the scalar, all forms of attack using collision characteristics can be categorized as a collision attack (CA). Walter [Wal01] firstly proposed the idea of a collision attack with a single trace, referred to as the Big Mac attack, which can determine the bits of the secret exponent by exploiting the collision characteristics between sliding window exponentiation and its pre-computation process utilizing the difference of means. ROSETTA (Recovery of Secret Exponent by Triangular Trace Analysis) [CFG+12] distinguishes squaring from multiplication operations by exploiting collision characteristics caused by the same input single precision multiplications in a squaring operation. Unlike ROSETTA, which determines inner-collision in one operation, i.e., squaring, HCCA (Horizontal Collision Correlation Attack) [BJPW14] exploits collision characteristics between two (or more) operations, particularly in the same input operands manipulated in target operations. Hanley et al. [HKT15] extended the idea of HCCA by exploiting the collision characteristics between the input and output operands of the target operations.

Whereas the collision attacks listed above focus on data-dependant leakages in target operations, e.g., field multiplications in the doubling or addition of scalar multiplication, several studies have proposed approaches to exploiting collision characteristics caused by different register behavior dependant on the value of the scalar by measuring location-dependent electromagnetic emissions [HMH+12, SHKS15]. In addition, these attacks employ clustering algorithms for the better discrimination of register-dependant leakages [HIM+14, JB17].

An adversary against ECDSA can obtain nonces through STA against scalar multiplication and then can recover the private key as follows:

$$s = k^{-1} \cdot (h + d \cdot r) \;\Leftrightarrow\; d = (k \cdot s - h) \cdot r^{-1} \tag{1}$$

2.3 Table-based Scalar Multiplication for ECDSA Signature Generation

In this section, we generalize regular table-based scalar multiplication for ECDSA signature generation, as shown in Definition 1, Algorithm 2, and Algorithm 3, to represent a target for the proposed attack. Regular table-based scalar multiplication involves a consistent number of point doubling and point addition during the table-based scalar multiplication phase.

Definition 1. Given regular table-based scalar multiplication and precomputation table $\mathcal{T}$ for the scalar multiplication, there are sequence $ks(k)$ for the row index of the table reference and function ws for $k \cdot P$ such that

$$ks(k) = (ks_0, ks_1, ..., ks_{c-1}), \quad ks_j \in \{0, 1, ..., r-1\}, \tag{2}$$

$$ws : ks_j \mapsto ws(ks_j) \in \mathbb{Z}_{>0}, \tag{3}$$

where ks_j is the row index for referencing the j-th column of the table, and the result of Algorithm 3 with sequence $ks(k)$ and ws is the same as with $k \cdot P$. $ks(k)$ and ws are determined depending on the scalar multiplication algorithm.

It is worth noting that our attack targets Algorithm 3 with function ws, which outputs a value that depends on only j, not ks_j. In other words, methods such as the sliding window method are excepted.

Algorithm 2 Preparation of pre-computed tables for regular table-based scalar multiplication

```
Require: base point P over $\mathcal{E}(\mathbb{F}_p)$ of order $ord$
Ensure: $r \times c$ pre-computation table $\mathcal{T}$
1: Initialize $r \times c$ table $\mathcal{T}$
2: for $j \in 0$ up to $c-1$ do
3:     for $i \in 0$ up to $r-1$ do
4:         Choose $\alpha_{i,j} \in \mathbb{Z}_{ord}$ that is appropriate for the current table-based scalar multiplication
5:         $\mathcal{T}[i, j] \leftarrow \alpha_{i,j} \cdot P$
6:     end for
7: end for
8: Return $\mathcal{T}$
```

Algorithm 3 Regular table-based scalar multiplication

```
Require: $k$ and $r \times c$ pre-computation table $\mathcal{T}$
Ensure: $Q = k \cdot P$
1: Set $ks(k) = (ks_0, ks_1, ..., ks_{c-1})$ and $ws$ to be appropriate for the current table-based scalar multiplication
2: $Q \leftarrow \infty$
3: for $j \in 0$ up to $c-1$ do
4:     $Q \leftarrow ws(ks_j) \cdot Q$
5:     $Q \leftarrow Q + \mathcal{T}[ks_j, j]$
6: end for
7: $k \leftarrow k \bmod ord$
8: Return $k, Q$
```

3 Proposed attack

This section proposes a novel key recovery attack on ECDSA signature generation employing regular table-based scalar multiplication. We set two assumptions for the threat model of our attack:

Assumption 1 We assume that ws is a function of index j in the loop in Algorithm 3. i.e., we consider only cases of ws, the output of which depends only on index j, not $ks(j)$.

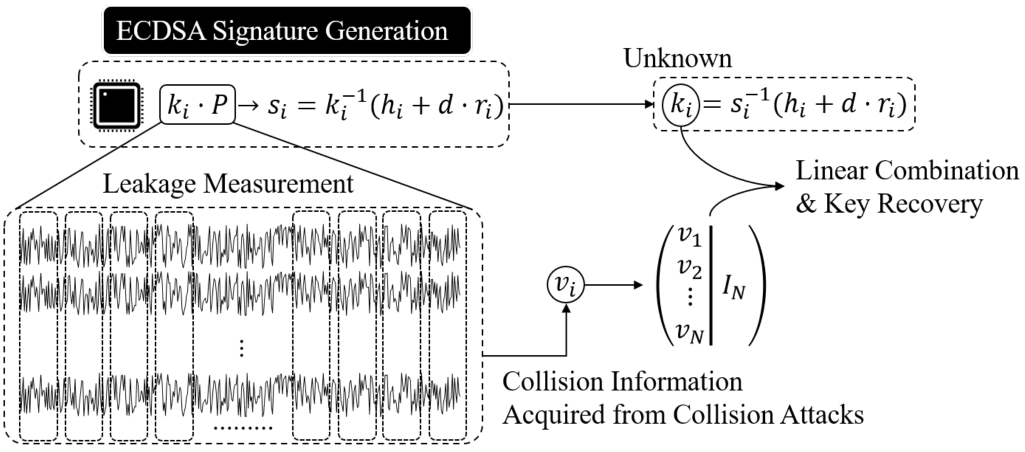
ECDSA Signature Generation
$k_i \cdot P \rightarrow s_i = k_i^{-1}(h_i + d \cdot r_i)$
Unknown
$k_i = s_i^{-1}(h_i + d \cdot r_i)$
Linear Combination & Key Recovery
Leakage Measurement
v_i
$\left(\begin{array}{c|c} v_1 & \\ v_2 & I_N \\ \vdots & \\ v_N & \end{array}\right)$
Collision Information Acquired from Collision Attacks

Figure 1: Overall flow of the proposed attack.

Assumption 2 We assume that an adversary can perform multiple ECDSA signature generation operations where the private key is fixed and acquire corresponding signatures and traces of the table-based scalar multiplication with fixed pre-computation table $\mathcal{T}$.

We emphasize that our attack can be performed even when the adversary has no knowledge of the pre-computed table entries. Because our attack only uses collisions between entries, the pre-computation phase is independent of our attack. We illustrate the details of the proposed attack in two steps: the preparation of the collision information and the recovery of the private key for the ECDSA signature. Figure 1 presents the overall flow of the proposed attack.

3.1 Preparation of Collision Information

First, we prepare collision information through side-channel analysis. Regular table-based scalar multiplication repeats the operations in lines 4-5 in Algorithm 3 c times, where c is defined according to the security parameter of the scalar multiplication. In each loop, point doubling and point addition are conducted respectively based on j and ks_j, i.e., iteration and the index of the row of the table. Note that, although security parameter c might be confidential, it can be easily obtained by visually inspecting the power consumption of the overall process of scalar multiplication. In addition, row size r of the table is guessable from c using a guessing scalar multiplication algorithm.

We can now consider a attack scenario where an adversary can obtain N signatures (r_i, s_i) generated with the fixed private key and different messages and their corresponding traces T_i, where $i = \{1, ..., N\}$, on the table-based scalar multiplications with the fixed pre-computation table which are operated during the signature generations [1]. Because lines 4-5 in Algorithm 3 are regularly iterated, the adversary can determine and extract samples corresponding to these operations, denoted by a subtrace, for each iteration and reconstruct the trace T_i as $T_i = (t_{i,0}, t_{i,1}, ..., t_{i,c-1})$, where $t_{i,j}$ represents a subtrace of the j-th loop.

Let us consider a specific iteration, e.g., $0 \le j \le c-1$, and every subtrace of each trace corresponding to the iteration, i.e., $t_{1,j}$, ... $t_{N,j}$. In this situation, we can perform collision attacks that determine whether a row index of the table corresponding to a subtrace is

[1]If it is not deterministic ECDSA signature, it is possible with the same messages.

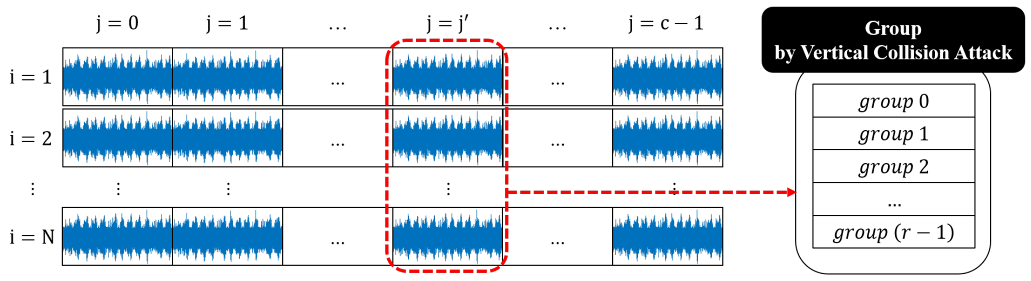

Figure 2: Flow for grouping of the subtraces for the j'-th loop. An adversary can determine whether each subtrace is related to the same data or not through collision attacks among vertical subtraces on the j'-th loop. With this information, the adversary can group vertical subtraces.

the same or not as the indexes of the table corresponding to the other subtraces. These attacks are possible because the data-dependent leakages of any two entries from $\mathcal{T}$ are highly correlated if the selected row index of the entries is the same. Because there are at most r different row entries in $\mathcal{T}$ for each iteration, we can classify all of the subtraces for the iteration in r different groups by repeating the process (the process described above is presented in Figure 2 and a more detailed version is provided in Algorithm 9 in Section 5). At this time, we assume that this grouping for all iterations is possible without any error[2]. For the rest of iterations, the grouping of the subtraces can be achieved using the same technique. After the grouping of the traces is completed, we can label each $t_{i,j}$ with $g_{i,j} \in \{0, 1, ..., r-1\}$. Consequently, we can define the *collision information vector* corresponding to (r_i, s_i) as $G_i = (g_{i,0}, g_{i,1}, ..., g_{i,c-1})$. In the next section, we describe how the private key can be recovered by utilizing the collision information vectors and the acquired signatures.

3.2 Key Recovery by Identifying Linearly Dependent Nonces

In the second step, we find the condition in which nonces are linearly dependent by utilizing the collision information and then recover the private key on the basis of this linear dependency. The values of each component $g_{i,j}$ of G_i are assigned arbitrarily as labels in the elements of the set $\{0, 1, ..., r-1\}$ only to classify the selected entries from a column of pre-computation table $\mathcal{T}$ into r groups. If we consider any two columns, the same label does not mean that the same row index of the entry is selected in the actual scalar multiplication operation. In other words, the labels of the collision information are independent both vertically and horizontally. Hence, in order to indicate independent information between different labels, each component $g_{i,j}$ of G_i is converted to one-hot representation E_{oh} as follows:

$$\begin{aligned} E_{oh} : \mathbb{Z}_r &\rightarrow \mathbb{Z}_{ord}^r \\ g_{i,j} &\mapsto e_{(g_{i,j}+1)} \end{aligned} \tag{4}$$

where e_i is a unit vector in which only the i-th component is 1 and all of the other components are 0. The collision information vector $G_i = (g_{i,0}, g_{i,1}, ..., g_{i,c-1})$ for nonce k_i used in the i-th signature generation can then be converted into vector $v_i = E_{oh}(g_{i,0}) \parallel E_{oh}(g_{i,1}) \parallel ... \parallel E_{oh}(g_{i,c-1}) \in \mathbb{Z}_{ord}^{r \cdot c}$. Hence, each component of v_i corresponds bijectively to each entry of pre-computation table $\mathcal{T}$ although we still do not know how to map this.

Now, we can consider v_i to be a coefficient vector representing a linear combination to construct nonce k_i with unknown variables, i.e., the table entries. Note that we set

[2] At the end of this section, we discuss how to deal with errors.

the codomain of E_{oh} as $\mathbb{Z}_{ord}^r$, not as $\mathbb{Z}_2^r$, because all of the values in the computations in the later process are defined in $\mathbb{Z}_{ord}$. Because v_i is a finite-dimensional vector, if there are enough vectors, then a specific vector can be expressed as a linear combination of the other vectors, i.e., it is linearly dependent, via Corollary 1.

Corollary 1. *Let V be a finite-dimensional vector space and let the dimension of V be $n = dim\, V$. Then, any subset of V that contains more than n vectors is linearly dependent.*

As a result, if there are N $(> r \cdot c)$ vectors, there exists at least one vector that can be expressed as a linear combination of the remaining vectors because v_i is in the $(r \cdot c)$-dimensional vector space, i.e., one of N different vectors can be expressed as a linear combination of the remaining vectors. However, it is possible that these linear dependencies also occur in $N <= r \cdot c$ different vectors because v_is are sparse vectors.

To obtain a linear dependency, we construct matrix A with N different vectors v_i as rows of the matrix and then construct M as follows:

$$A = \left[v_1^T \parallel v_2^T \parallel \ldots \parallel v_N^T\ \right]^T \in Mat_{N\times(r\cdot c)}(\mathbb{Z}_{ord}) \tag{5}$$

$$M = \left[A \parallel I_N\right] \in Mat_{N\times(r\cdot c+N)}(\mathbb{Z}_{ord}) \tag{6}$$

By performing Gaussian elimination on matrix M as shown in (7), the row echelon matrix, denoted by A', can be calculated from A. Then, $A' = B \cdot A$ is established for A, B, and A' in (7). Note that, because the components of v_i represent coefficients of a linear combination of all entries of the pre-computation table, Gaussian elimination is calculated with modulus ord.

$$\begin{gathered} M = \left[A \parallel I_N\right] \in Mat_{N\times(r\cdot c+N)}(\mathbb{Z}_{ord}) \\ \downarrow \textbf{Gaussian elimination} \\ M' = \left[A' \parallel B\right] \in Mat_{N\times(r\cdot c+N)}(\mathbb{Z}_{ord}) \end{gathered} \tag{7}$$

where $A' = [{v'}_1^T \parallel {v'}_2^T \parallel \ldots \parallel {v'}_N^T]^T \in Mat_{N\times(r\cdot c)}(\mathbb{Z}_{ord})$, and $B = [b_1^T \parallel b_2^T \parallel \ldots \parallel b_N^T]^T \in Mat_{N\times N}(\mathbb{Z}_{ord})$.

If there is a linearly dependent relationship between the used vectors, there is at least one row with all components at zero in the row echelon matrix A'. Let us define the corresponding row index as $g \in \{2, 3, \ldots, N\}$. Then, (8) is satisfied, where b_g represents the g-th row vector of B.

$$b_g \cdot A = \vec{0} \ \textit{and}\ b_g \neq \vec{0} \tag{8}$$

where $\vec{0}$ is a vector with all components at zero. This means that all components of $b_g = (b_{g,1}, b_{g,2}, ..., b_{g,N})$ are coefficients for the linear combination of nonces k_i resulting 0. In other words, the corresponding nonces with non-zero components in b_g are linearly dependent. Finally, from (9) and the linearly dependent nonces, the private key d can be recovered as shown in (10). Note that finishing the Gaussian elimination in our attack is not required. Because the attacker needs only one row with all entries at zero, the attacker can stop Gaussian elimination early as soon as this row appears.

$$\begin{aligned} s_i &= k_i^{-1} \cdot (h_i + r_i \cdot d) \\ \Leftrightarrow k_i &= s_i^{-1} \cdot (h_i + r_i \cdot d) \\ \Leftrightarrow k_i &= s_i^{-1} \cdot h_i + s_i^{-1} \cdot r_i \cdot d \end{aligned} \tag{9}$$

$$\begin{aligned}
0 &= \sum_i b_{g,i} \cdot k_i \\
&= \sum_i b_{g,i} \cdot (s_i^{-1} \cdot h_i + s_i^{-1} \cdot r_i \cdot d) \\
&= \sum_i (b_{g,i} \cdot s_i^{-1} \cdot h_i) + (\sum_i b_{g,i} \cdot s_i^{-1} \cdot r_i) \cdot d \\
\Leftrightarrow d &= \left(\sum_i b_{g,i} \cdot s_i^{-1} \cdot r_i \right)^{-1} \cdot \left(- \sum_i b_{g,i} \cdot s_i^{-1} \cdot h_i \right)
\end{aligned} \tag{10}$$

3.3 Recovery of the Entries in the Pre-computation Table

Trivially, all entries in the pre-computation table of $\mathcal{T}$ can be found after the private key is recovered. If private key d is recovered by the proposed attack, we can calculate all k_i using (9). Then, we have $N \geq r \cdot c$ linear equations where the left side of the equation is a linear combination between coefficients of v_i and the unknown entries of $\mathcal{T}$ and the right side of the equation is k_i. Hence, if there are sufficient digital signatures and traces capable of finding the basis of the vector space of v_is (i.e., if N is sufficiently large), the value of all table entries can be recovered by solving the linear equations. However, the actual order of entries in each column might be known or unknown according to the scalar multiplication algorithm used and the use of countermeasures.

3.4 Discussion of Issues regarding the Proposed Attack

There are some issues with our attack. First, computing the collision information accurately is required. i.e., the collision information should have no errors. However, it is possible for errors to occur in the grouping stage because trace noise can induce errors in the collision attacks. Currently, only a trial-and-error solution is possible. From a sufficient pool of $N_p(> N)$ traces, the attacker chooses N traces randomly and conducts an attack until the recovery of the secret key because our attack assumes that there is no error in the grouping stage. Other solutions can be the focus of future work.

Secondly, for the same reason, our attack may be limited for relatively large row size r. The number of groups for each column grows with r. However, because clustering errors are dependent on the signal-to-noise ratio (SNR) of the traces, the limitation associated with the row size is changed according to the SNR of traces. Thus, we skip formulating an accurate limitation for r.

We investigate the probability of incorrect clustering between two different entries assuming an ideal environment, i.e., only the Hamming weight as a leakage model and without noise. In the same way, as Section 5 will describe, we can target points of interest about loading two multi-precision integers 32-bit-wisely from the pre-computation table. i.e., we can target the leakage on loading the coordinates X, Y of $k \cdot P$, where P is a base point. These coordinates X and Y of $k \cdot P$ are determined by the corresponding 256-bit scalar. Thus, to simplify the calculation of probability, we assume here that we can target 8 points of interest corresponding to the load scalar for each entry instead of the coordinates. i.e., there are 8 points of interest here. Note that we assume that each entry from the pre-computation table is different. This assumption is reasonable because the probability of two random 256-bit values being the same is extremely low if random extraction is valid. When two different random 32-bit words are chosen, the probability of two Hamming weights of the two words being equal is as follows:

$$p = \frac{1}{2^{64}} \sum_{i=1}^{31} \binom{32}{i} \cdot \left(\binom{32}{i} - 1 \right). \tag{11}$$

Thus, the probability of two leakages being equal when two different entries are chosen is as follows:

$$P = \sum_{i=1}^{8} \binom{8}{i} \cdot p^i \cdot \left(\frac{1}{2^{32}} \right)^{8-i} \tag{12}$$

We claim that the trial-and-error method can be successful because the probability P is very low. We also empirically verify this through 100,000 simulations, with no incorrect clustering for each $r \in \{2^2, 2^4, 2^8\}$ [3].

4 Applications: Case studies

In this section, we explore two types of representative regular table-based scalar multiplication for ECDSA signature generation that are considered to be practically or perfectly resistant to side-channel analysis as the targets of our attack.

4.1 Case Study: Fixed-base Comb Scalar Multiplication

Table-based scalar multiplication such as the fixed-base NAF windowing method and the fixed-base comb method is widely used for efficiency in many crypto libraries including OpenSSL, Mbed TLS, Bouncy Castle [Libc, Libb, Liba]. Of these general table-based scalar multiplication approaches, we take the well-known fixed-base comb method as an example to demonstrate that our attack can be easily applicable to general table-based scalar multiplication deployed in ECDSA signature generation. In this section, for the sake of simplicity, we select the original version of the fixed-base comb method here, as described in Algorithm 4, although there are many variations to improve the efficiency and counteract side-channel analysis.

Algorithm 4 is a version of the algorithm combining Definition 1, Algorithm 2, and Algorithm 3 for fixed-base comb scalar multiplication. For the fixed-base comb method,

$$ws(x) = 2 \text{ for any } x, \;\; r = 2^w, \;\; c = d \tag{13}$$

holds. Note that, although the pre-computed table has only one column, we assume that there is a $r \times c$ pre-computed table, which is the original table repeated c times, for the sake of simplicity. With this setting, our attack can be applied easily to fixed-base comb scalar multiplication. Table 1 shows how many signatures and traces are required for our attack depending on the security parameter of the fixed-base comb method.

4.2 Case Study: T_SM Scalar Multiplication

T_SM scalar multiplication is proposed to be resistant against STA [SCM+18]. To achieve security against STA, it is designed to differ from other types of scalar multiplication. It can be employed only in specific settings such as ECDSA signature generation because it outputs both random nonce k and corresponding result point $k \cdot P$ for scalar multiplication using randomly pre-computed tables whereas conventional scalar multiplication algorithms calculate $k \cdot P$ based on the inputted k, as shown in Algorithm 5 and Algorithm 6.

T_SM scalar multiplication, as described in Algorithm 6, is performed by selecting a random *row* per column, which is represented by j, and accumulating the scalar and

[3]Except $r = 2^{16}$ because it is unrealistic that this would be employed in a real environment.

Algorithm 4 Fixed-base comb scalar multiplication [LL94, HMV04]

```
Require: base point $P \in \mathcal{E}(\mathbb{F}_p)$ of order $ord$, scalar $k = (k_{t-1}, ..., k_1, k_0)_2 \in \mathbb{Z}_{ord}$, security parameter $w$, $d = \lceil t/w \rceil$
Ensure: $Q = k \cdot P$
    Pre-computation
 1: for $j = (j_{w-1}, ...j_1, j_0)_2 \in 0$ up to $2^w - 1$ do
 2:     $\mathcal{T}[j] \leftarrow \sum_{i=0}^{w-1} (2^{i \cdot d} \cdot j_i) \cdot P$
 3: end for
    Scalar recoding
 4: Let $k = (k_{wd-1}, ..., k_1, k_0)_2$ be a $wd$-bit integer by padding $k$ on the left with 0s if necessary
 5: Prepare $k' \leftarrow (k'_{d-1}, ..., k'_1, k'_0)_{2^w}$        ▷ $k' = ks(k)$
 6: for $i \in 0$ up to $d - 1$ do
 7:     $k'_i \leftarrow (k_{(w-1) \cdot d+i}, ..., k_{2 \cdot d+i}, k_{1 \cdot d+i}, k_{0 \cdot d+i})_2$
 8: end for
    Calculation
 9: $Q \leftarrow \infty$
10: for $i \in d - 1$ down to 0 do
11:     $Q \leftarrow 2 \cdot Q$
12:     $Q \leftarrow Q + \mathcal{T}[k'_i]$
13: end for
14: Return $Q$
```

Table 1: For the fixed-base comb method, the size, the number of entries in the pre-computation table, and attack parameter according to security parameter.

secp256r1 [Bro10] (128-bit security)		Pre-computation table		Attack phase	
w	d	Size	# of entries	# of group per loop	Dimension of v_i
2	128	128 B	4	4	512
4	64	512 B	16	16	1,024
8	32	8 KB	256	128	8,192
16	16	2 MB	65,536	65,536	1,048,576

Algorithm 5 Preparation of pre-computed tables for T_SM scalar multiplication [SCM+18]

```
Require: security parameter $\lambda = m \cdot n \in \mathbb{Z}^+$, base point $P$ on group $\mathcal{E}(\mathbb{F}_q)$ of order $ord$
Ensure: $\mathcal{T}_k, \mathcal{T}_P$
1: for $i \in 0$ up to $2^m - 1$ do
2:     for $j \in 0$ up to $n - 1$ do
3:         $\mathcal{T}_k[i, j] \overset{R}{\leftarrow} \mathbb{Z}_{2^{ord}}$        // Choose a $ord$-bit random integer
4:         $\mathcal{T}_P[i, j] \leftarrow \mathcal{T}_k[i, j] \cdot P$
5:     end for
6: end for
7: Return $\mathcal{T}_k, \mathcal{T}_P$
```

Algorithm 6 T_SM scalar multiplication [SCM+18]

```
Require: security parameter $\lambda = m \cdot n \in \mathbb{Z}^+$, the order $ord$ of base point $P \in \mathcal{E}(\mathbb{F}_q)$,
    pre-computation tables $\mathcal{T}_k$ and $\mathcal{T}_P$
Ensure: $k, Q = k \cdot P$
1: $k \leftarrow 0, Q \leftarrow \infty$
2: for $j \in 0$ up to $n - 1$ do
3:     $row \overset{R}{\leftarrow} \mathbb{Z}_{2^m}$        // Choose an $m$-bit random integer
4:     $k \leftarrow k + \mathcal{T}_k[row, j]$
5:     $Q \leftarrow Q + \mathcal{T}_P[row, j]$
6: end for
7: $k \leftarrow k \ mod \ ord$
8: Return $k, Q$
```

corresponding point for each table corresponding to the rows and columns, respectively. This accumulation is iterated n times, hence, n long integer additions for k (line 4 in Algorithm 6) and n elliptic curve point additions for $k \cdot P$ (line 5 in Algorithm 6) are conducted. Consequently, the accumulated results for the scalar and point are returned as a nonce k and point $k \cdot P$. With this property, T_SM scalar multiplication can be employed for ECDSA signature generation by replacing step 1 and 2 in Algorithm 1 because it outputs k and corresponding $k \cdot P$.

Note that it is not possible to know whether the output scalar k from T_SM scalar multiplication is uniformly random in $[1, ord - 1]$, as described in step 1 in Algorithm 1, because k is an accumulated result of random entries. However, we can evaluate the security of T_SM scalar multiplication by calculating the number of cases selecting random scalars from table $\mathcal{T}_k$ to produce a result k of $(2^m)^n = 2^{mn} = 2^\lambda$, which is the same as the complexity of the λ-bit scalar for conventional scalar multiplication. Although it is not yet investigated whether the T_SM method outputs biased nonces or not, this is not of interest in this work, and remains a goal for future work.

It is worth noting that every entry in the pre-computed table of T_SM scalar multiplication is independent and this is an extreme case of table-based scalar multiplication. Due to this, we choose to explore the vulnerability of T_SM against our attack, although the threat model for the T_SM method is not considered against MTA. Note also that it is difficult to reveal the table entries through only STA against the pre-computation phase, which is executed once and occupies many execution times, because the digital oscilloscope has a restricted sampling memory [DLO+19].

Although the notation in Section 2 is not completely consistent with T_SM scalar multiplication, T_SM scalar multiplication also repeatably uses two pre-computed tables and uses an entry of a column per iteration. Thus, T_SM scalar multiplication can be represented as a case of Algorithm 3 with (14) below for security parameter $\lambda = m \cdot n$,

$$ws(x) = 1 \text{ for any } x, \quad r = 2^m, \quad c = n. \tag{14}$$

In other words, step 4 in Algorithm 3 is skipped. With this setting, our attack can also be applied to T_SM scalar multiplication. Table 2 shows how many signatures and traces are required for our attack depending on the security parameter of the T_SM method.

5 Experimental Results

In this section, we validate our attack by conducting a practical experiment of our attack on 256-bit T_SM scalar multiplication for security parameter $\lambda = m \cdot n = 2 \times 128$ as a proof of work.

Table 2: The size, the number of entries of pre-computation table, and attack parameter according to security parameter of T_SM method.

$\lambda = m \cdot n$ $= 256$		Pre-computation table		Attack phase	
m	n	Size	# of entries	# of groups per loop	Dimension of v_i
2	128	48 KB	512	4	512
4	64	96 KB	1,024	16	1,024
8	32	768 KB	8,192	128	8,192
16	16	96 MB	1,048,576	65536	1,048,576

5.1 Experimental Setup and Trace Acquisition

We implement T_SM calculation, where $\lambda = 256$, $m = 2$, and $n = 128$, operating on ARM Cortex-M4-based STM32F405 microcontroller [Devb] which is embedded on ChipWhisperer [OC14] CW308T-STM32F [Deva] target board. T_SM calculation (Algorithm 6) is implemented in C and Thumb-2 assembly language. In detail, line 4 for scalar addition is implemented as 256-bit long integer addition and line 5 for point addition is implemented as "madd-2004-hmv" point addition [BL], described in Algorithm 7, where randomly selected entries from $\mathcal{T}_P$ are inputted as the operand P_2. For practical reasons, the first iteration of Algorithm 6 where $j = 0$ is implemented as simple loading, i.e., copying corresponding data from the tables to the variables of k and Q from $\mathcal{T}_k$ and $\mathcal{T}_P$ according to the first *row* value.

Algorithm 7 Point addition ($y^2 = x^3 - 3x + b$, Jacobian Coordinates) [BL]

```
Require: $P_1 = (X_1, Y_1, Z_1)$, $P_2 = (X_2, Y_2, 1)$ in Jacobian coordinates
Ensure: $P_1 + P_2 = (X_3, Y_3, Z_3)$ in Jacobian coordinates

1: $T_1 = Z_1^2$
2: $T_2 = T_1 \cdot Z_1$
3: $T_1 = T_1 \cdot X_2$
4: $T_2 = T_2 \cdot Y_2$
5: $T_1 = T_1 - X_1$
6: $T_2 = T_2 - Y_1$
7: $Z_3 = Z_1 \cdot T_1$
8: $T_3 = T_1^2$
9: $T_4 = T_3 \cdot T_1$
10: $T_3 = T_3 \cdot X_1$
11: $T_1 = 2 \cdot T_3$
12: $X_3 = T_2^2$
13: $X_3 = X_3 - T_1$
14: $X_3 = X_3 - T_4$
15: $T_3 = T_3 - X_3$
16: $T_3 = T_3 \cdot T_2$
17: $T_4 = T_4 \cdot Y_1$
18: $Y_3 = T_3 - T_4$
19: Return $(X_3, Y_3, Z_3)$
```

A power consumption trace of our implementation operating at 5 MHz acquired with a 250 MS/s sampling rate using a LeCroy HDO6104A oscilloscope [Devc] is presented in Figure 3. We collected 513 ($= 2^m \cdot n + 1$) traces because this number guarantees the successful recovery of the private key, as outlined in the previous section. Note that, although we already know $m = 2$ and $n = 128$, we can obtain n easily by inspecting the power consumption through common means in side-channel analysis such as visual inspection and cross-correlation. By knowing $n = 128$, we can guess $m = 2$. After acquisition, a 5 MHz low-pass filter is applied to every trace.

5.2 Identification and Extraction of Target Operation Traces

In the next step, we perform visual inspection to identify and extract the target operation traces to exploit collision characteristics and then construct the collision information by

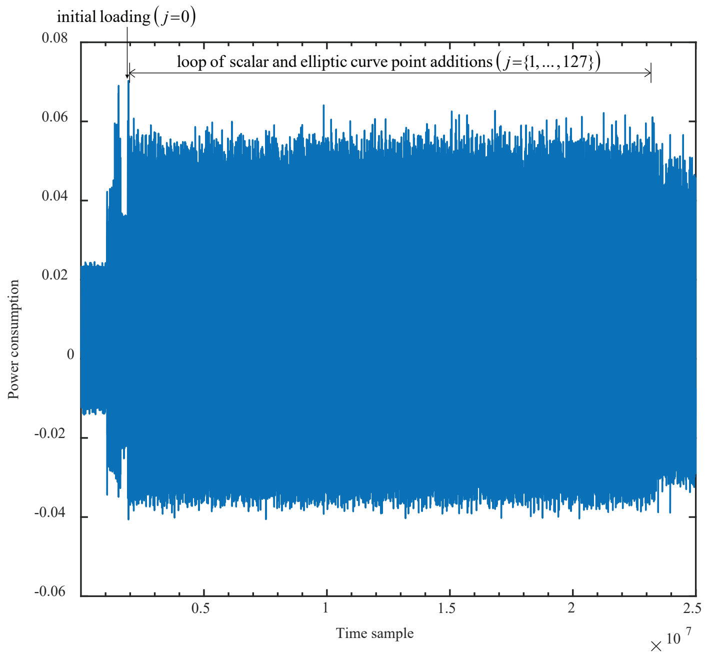

Figure 3: Power consumption trace for T_SM scalar multiplication.

grouping. To this end, we apply a low-pass filter with an arbitrary frequency lower than the operating frequency, e.g., 1 MHz, to easily identify operations as presented in Figure 4.

First, we determine which are the power consumption traces corresponding to the long integer additions and the point additions. As represented in Figure 4 (a), each point addition operation can be identified by eleven peaks because it consists of eight multiplications and three squarings of long integers. The long integer additions can also be determined because it is located before the point addition operations. We then identify the loading operation, which is located before the first iteration of the long integer addition and the point addition.

Now our target operations for the attack are the loading operation for $j = 0$ and one long integer addition and two long integer multiplications for $j \in \{1, \ldots 127\}$, i.e., line 3 and 4 in Algorithm 7, which manipulates X_2 and Y_2 selected from $\mathcal{T}_P$ according to *row* (these operations are highlighted with gray boxes in Figure 4 (a) and (c)). Note that, in a real attack scenario, because an adversary may not know when the target long integer multiplication algorithm is operated, it should guess the location of the target operation in this case. However, because this location is necessarily located in the early stages of the iteration, the cost of guessing the location may not be great, thus it does not seriously degrade the feasibility of the attack.

With the power consumption samples of the target operations as references, we can determine the locations of the target operations on the other 512 power consumption traces from T_SM scalar multiplication by utilizing cross-correlation. After that, we reconstruct the target operation traces $C_{i,j}$ where $i \in \{1, ..., 513\}$ indicates the trace acquisition number. Hence $C_{i,0}$ consists of samples of the loading operation and $C_{i,j}$ where $j \in \{1, ..., 127\}$ consists of samples of one long integer addition and two long integer multiplications sequentially concatenated.

Note that 1 MHz low-pass-filtered traces are used only for the identification of target operations. In the rest of the attack process, the original 5 MHz-low-pass-filtered traces are used for trace cutting exploiting cross-correlation and the reconstruction of $C_{i,j}$. After the

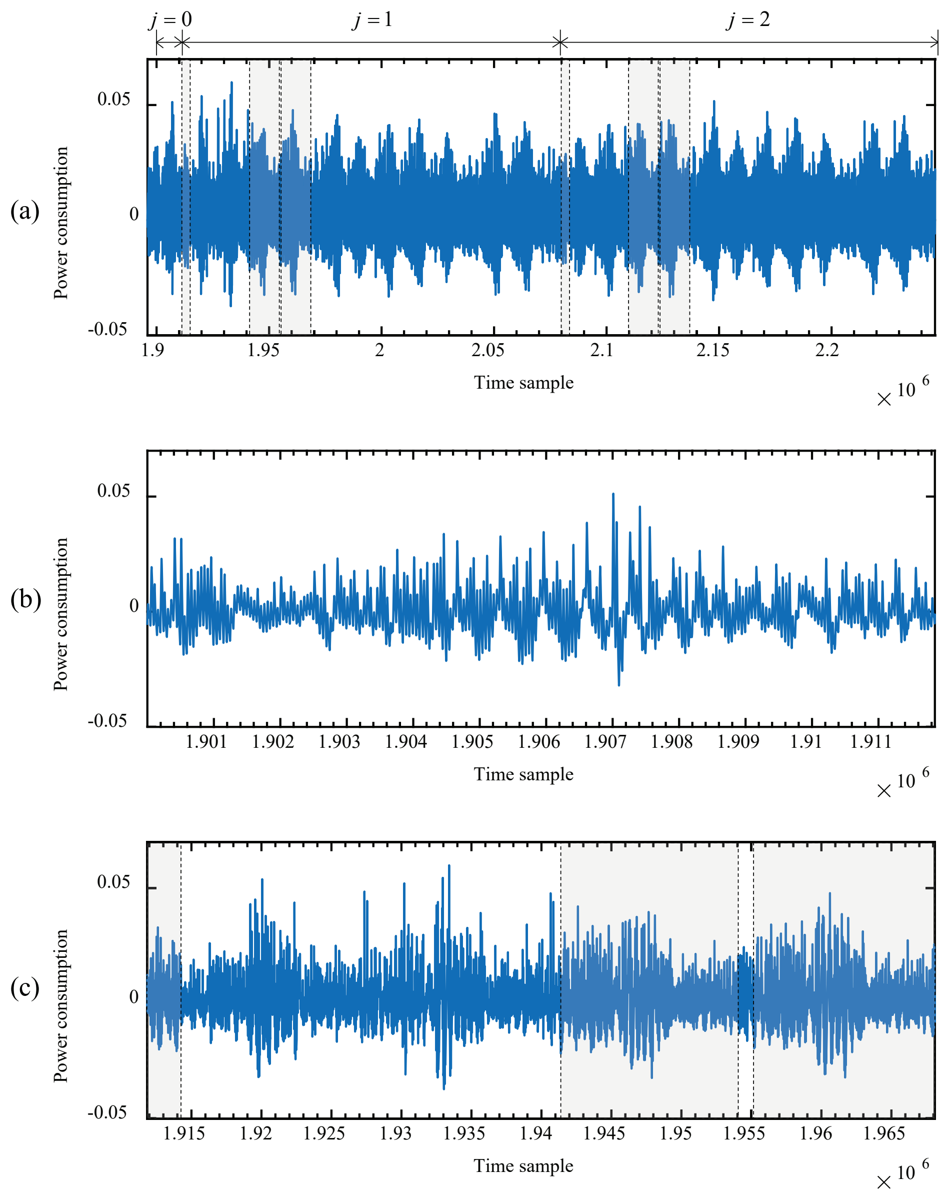

Figure 4: Example of 1 MHz low-pass-filtered power consumption trace for (a) the first part of T_SM scalar multiplication where $j = 0$ to $j = 2$, (b) initial loading operation where $j = 0$, and (c) the group of operations corresponding to one iteration where $j = 1$. In each iteration, gray boxes indicates intervals of three target operations according to one long integer addition, which manipulates an entry from table $\mathcal{T}_k$, and two long integer multiplications, which manipulate an entry from the table $\mathcal{T}_P$ consisting of two coordinate values of an elliptic curve point.

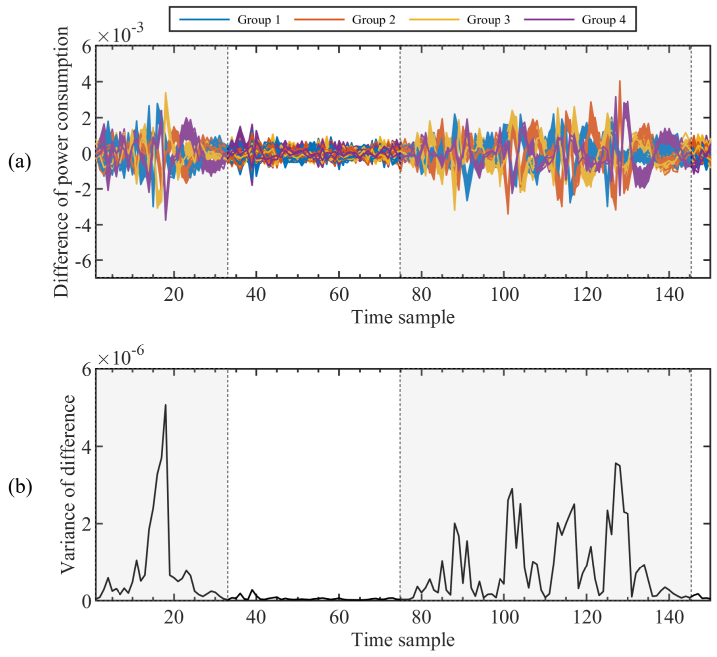

Figure 5: Example of (a) power consumption difference traces and (b) a variance of power consumption difference trace corresponding to initial loading operations where $j = 0$. Gray boxes represent selected PoIs.

extraction of all $C_{i,j}$, we apply the integration compression technique [MOP08] on each $C_{i,j}$ for noise reduction in which 50 samples are integrated into a single sample because we sampled 50 samples per clock cycle. We treat these compressed traces as the same $C_{i,j}$ because these are used for the rest of the attack.

5.3 Making Difference Traces and Finding Points of Interest

In this step, we make difference traces $D_{i,j}$, as described in Algorithm 8, by calculating average trace $\tilde{C}_j$ and subtracting it from each $C_{i,j}$ for $j \in \{0, ..., 127\}$ independently to exploit collision characteristics followed by finding points of interest (PoIs). We illustrate examples of difference traces for $j = 0$ and $j = 127$ in Figure 5 and Figures 6, 7, and 8, respectively, as these represent two types of target operations. In the corresponding figures, all $D_{i,j}$ are plotted in different colors according to the actual *row* value used during the trace acquisition process to visually represent exploitable collision characteristics. Of course, in a real attack scenario, an adversary cannot acquire differently colored figures at this stage because $D_{i,j}$ has not been grouped yet. Hence to find PoIs, it should utilize a variance trace, which represents the variance of samples corresponding to each index in the time domain.

For $j = 0$, because the loading operation solely loads data from each table $\mathcal{T}_k$ and $\mathcal{T}_P$ and stores the same data for some variables, every peak in the variance trace can be selected as PoIs. On the other hand, for $j \in \{1, ..., 127\}$, we heuristically choose the moment only when some data are loaded from the tables to exploit collision characteristics because there are always collision characteristics caused by load operation independent of how to compute long integer operation. Hence, these PoIs should be carefully selected because it is difficult to identify PoIs as shown in Figures 6, 7, and 8. As highlighted in Figures 6, 7, and 8 in gray boxes, PoIs for iterations of one long integer addition (Figure 6) and

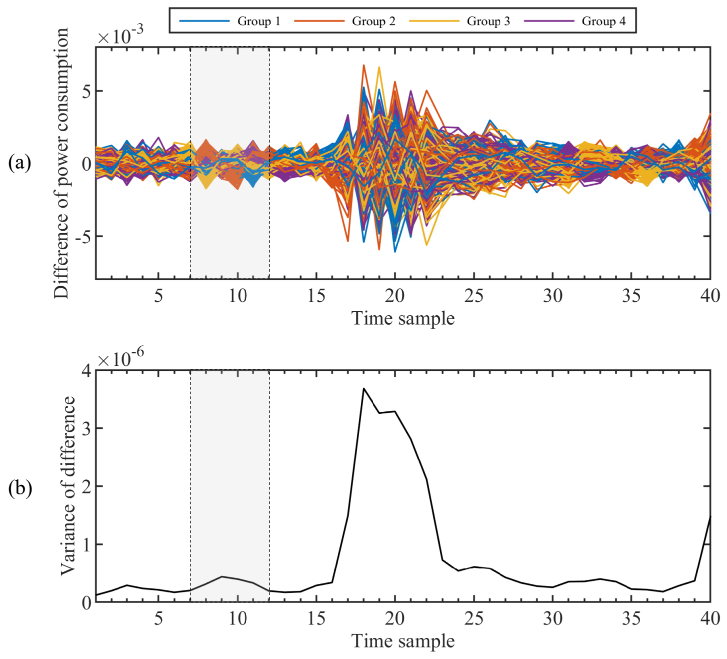

Figure 6: Example of (a) power consumption difference traces and (b) a variance of power consumption difference trace corresponding to long integer addition operations where $j = 127$. Gray boxes represent selected PoIs.

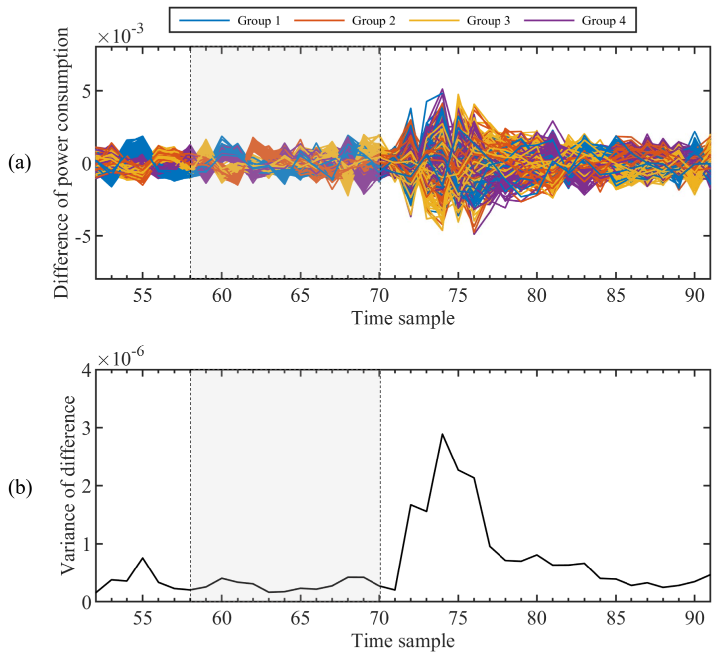

Figure 7: Example of (a) power consumption difference traces and (b) a variance of power consumption difference trace corresponding to the first long integer multiplication operation where $j = 127$. Gray boxes represent selected PoIs.

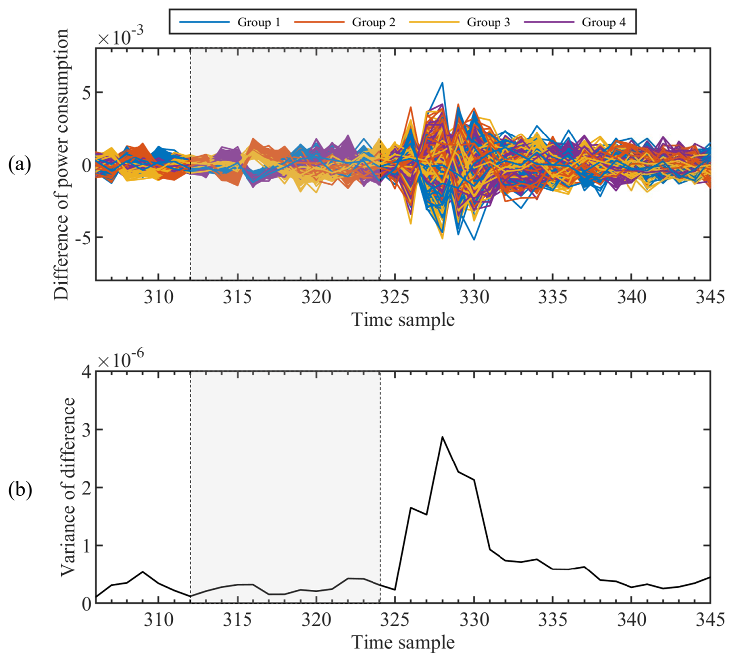

Figure 8: Example of (a) power consumption difference traces and (b) a variance of power consumption difference trace corresponding to the second long integer multiplication operation where $j = 127$. Gray boxes represent selected PoIs.

Algorithm 8 Making difference traces

```
Require: $C_{i,j}$ with $i \in \{1, ..., N\}$ and $j \in \{0, ..., n-1\}$
Ensure: $D_{i,j}$ with $i \in \{1, ..., N\}$ and $j \in \{0, ..., n-1\}$
    Calculate mean traces
1: for $j = 0$ up to $n-1$ do
2:     $\tilde{C}_j = \frac{1}{N}\sum_{i=1}^{N} C_{i,j}$
3: end for
    Calculate difference traces
4: for $j = 0$ up to $n-1$ do
5:     for $i = 1$ up to $N$ do
6:         $D_{i,j} = C_{i,j} - \tilde{C}_j$
7:     end for
8: end for
9: Return $D_{i,j}$
```

two long integer multiplications (Figure 7 and Figure 8) can be determined by relatively low peaks located before large peaks in the variance trace. Note that, in this case, the variance trace is generated by utilizing all subtraces corresponding to the same operation, i.e., every $D_{i,j}$ with $j \in \{1, ..., 126\}$. Finally, with the selected PoIs, which are represented by indexes of samples, we reconstruct each $D'_{i,j}$ only consisting of samples corresponding to PoIs from each $D_{i,j}$.

5.4 Grouping Traces with Correlation Coefficients

Now, we can determine the group label for each $D'_{i,j}$ using Algorithm 9 on $N = 513$ traces where $i \in \{1, ..., N\}$ and j is fixed with a certain value, e.g., $j = 0$. First, we prepare two $N \times 2^m$ matrices ρ and λ, one for the Pearson correlation coefficients and another to determine the group label, respectively. Then, we calculate the correlation coefficients $\rho_{i,1}$ between trace $D'_{1,0}$ and N traces $D'_{i,0}$. By comparing each $\rho_{i,1}$ and the mean value of every $\rho_{i,1}$ where $i \in \{1, ..., N\}$, we set entries of the first column of λ as one if $\rho_{i,1}$ is larger than the mean or zero otherwise. For the next step, we find $D'_{i,0}$ with the lowest $\rho_{i,1}$, which means the least similar trace to the first group and calculate $\rho_{i\in\{1,...,N\},2}$. Now we set the entries of the second column of λ by comparing each $\rho_{i,2}$ and the mean of all $\rho_{i,2}$ similarly as described above. The rest of the process for determining the entries of λ can be generalized as finding the least similar coefficients between every traces and setting the next column of λ. The least similar trace to the former groups can be determined by finding row indices with all zero entries for λ and the minimum of the sum of the correlation coefficients for the former groups corresponding to these indices.

After all of the entries of ρ and λ are calculated, we can determine vector $(l_1, ..., l_N)$ corresponding to the j-th components of collision information vectors where j is fixed, i.e., $g_{i,j} = l_i$. First, we set the $D'_{1,0}$ as the first group, i.e., $l_1 = 0$. For the remaining $D'_{i,0}$, we find the column index of λ with only one non-zero row entry and all others at zero and determine the group label according to that column index. If there is a row entry with more than one, the group label can be determined by finding the column index with the largest correlation coefficient corresponding to the row index on ρ. By iterating Algorithm 9 with the remaining $D'_{i,j}$ where $j \in 1, ..., 127$, we successfully determine the entire collision information vector $G_{i\in\{1,...,N\}}$ without error.

5.5 Recovery of the Private Key

The final step is the recovery of the private key by exploiting the collision information acquired in the previous step. We transform the collision information vectors in one-hot representation, i.e., v_i, to find linearly dependent nonces. We can generate matrix M in (7). Next, we calculate M' through Gaussian elimination on M. Because the matrix consists of 513 transformed vectors more than the total number of entries in the pre-computation table, there must exists at least one row in which all components are zero in part A' [4]. Then, the row vector in B corresponding to the zero row vector of A' is a vector resulting that the linear combination of nonces with its components is zero. Finally, we can recover the private key of ECDSA using (10).

5.6 Discussion of Clustering Algorithms

Note that we provided a correlation-based clustering algorithm (as shown in Algorithm 9) and confirmed that the clustering algorithm works well by real experiment for $m = 2$, showing that our attack is valid. This does not guarantee that the clustering algorithm works for a larger class. Because the correlation of fewer PoIs is susceptible to noise,

[4]In an our experiment, the row with all zero components in A' appeared in the 390th out of 513 vectors because the v_i's are sparse vectors.

Algorithm 9 Grouping traces with correlation coefficients between traces

```
Require: $D'_{i,j}$ with $i \in \{1, ..., N\}$ and fixed $j = c$
Ensure: $N \times 1$ group label vector $\{g_{1,c}, ..., g_{N,c}\}^T$
 1: Prepare $N \times 1$ vector $L = \{l_1, ..., l_N\}^T$ for the output
 2: Prepare $N \times 2^m$ matrix $\rho$ with zero entries
 3: Prepare $N \times 2^m$ matrix $\lambda$ with zero entries
 4: for $i = 1$ up to $N$ do
 5:     $\rho[i, 1] \leftarrow corr(D'_{1,j}, D'_{i,j})$
 6: end for
 7: for $i = 1$ up to $N$ do
 8:     if $\rho[i, 1] > mean(\{\rho[i, 1], \rho[2, 1], ..., \rho[N, 1]\})$ then
 9:         $\lambda_{i,1} \leftarrow 1$
10:     end if
11: end for
12: for $q = 2$ up to $2^m$ do
13:     $vert\_cand\_set \leftarrow \{i | \arg\min_i (\sum_{r=1}^{q-1} \lambda_{i,r})\}$
14:     $next\_index \leftarrow \arg\min_{i \in vert\_cand\_set} (\sum_{r=1}^{q-1} \rho_{i,r})$
15:     for $i = 1$ up to $N$ do
16:         $\rho[i, q] \leftarrow corr(D'_{next\_index,j}, D'_{i,j})$
17:     end for
18:     for $i = 1$ up to $N$ do
19:         if $\rho[i, q] > mean(\{\rho[1, q], \rho[2, q], ..., \rho[N, q]\})$ then
20:             $\lambda_{i,q} \leftarrow 1$
21:         end if
22:     end for
23: end for
    Determine final label results
24: $l_1 = 0$
25: for $i = 2$ up to $N$ do
26:     if $mean(\lambda_{i,q \in \{1,...,2^m\}}) = 1/2^m$ then
27:         $l_i \leftarrow (\arg\max_{r \in \{1,...,2^m\}} (\lambda_{i,r})) - 1$
28:     else
29:         $hori\_cand\_set \leftarrow \{q | \lambda_{i,q} \neq 0\}$
30:         $l_i \leftarrow (\arg\max_{r \in hori\_cand\_set} (\rho_{i,r})) - 1$
31:     end if
32: end for
33: Return $L$
```

the success of the clustering algorithm for a larger class is unclear. Thus, improving the accuracy of the clustering algorithm or developing other clustering algorithms should be considered in future work.

6 Conclusion

We proposed a novel key recovery attack against ECDSA signature generation employing regular table-based scalar multiplication by exploiting side-channel collisions between unknown entries. Without knowing the actual values of the table entries, an attacker can extract collision information indicating which group each entry belongs to for every individual column of a pre-computation table through vertical collision attacks. Next, the private key can be recovered by finding the condition in which several nonces are linearly dependent exploiting only the collision information. Additionally, we explained that all of the unknown entries in the pre-computation table can be recovered using the recovered private key with sufficiently more digital signatures and traces. We then presented case studies for our attack against ECDSA employing fixed-base comb and T_SM scalar multiplication to illustrate that our attack can be easily applicable to other forms of table-based scalar multiplication. Finally, we validated that our attack is a real threat by conducting a practical experiment with power consumption traces acquired during the operations of T_SM scalar multiplication on an ARM Cortex-M based microcontroller. We also detailed the validation process.

References

[ABF$^+$15] Thomas Allan, Billy Bob Brumley, Katrina Falkner, Joop van de Pol, and Yuval Yarom. Amplifying side channels through performance degradation. Cryptology ePrint Archive, Report 2015/1141, 2015. https://eprint.iacr.org/2015/1141.

[AFG$^+$14] Diego F. Aranha, Pierre-Alain Fouque, Benoît Gérard, Jean-Gabriel Kammerer, Mehdi Tibouchi, and Jean-Christophe Zapalowicz. GLV/GLS decomposition, power analysis, and attacks on ECDSA signatures with single-bit nonce bias. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 262–281. Springer, Heidelberg, December 2014.

[AFV07] Frederic Amiel, Benoit Feix, and Karine Villegas. Power analysis for secret recovering and reverse engineering of public key algorithms. In Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors, *SAC 2007*, volume 4876 of *LNCS*, pages 110–125. Springer, Heidelberg, August 2007.

[ANT$^+$20] Diego F. Aranha, Felipe Rodrigues Novaes, Akira Takahashi, Mehdi Tibouchi, and Yuval Yarom. LadderLeak: Breaking ECDSA with less than one bit of nonce leakage. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 20*, pages 225–242. ACM Press, November 2020.

[ASS17] Alejandro Cabrera Aldaya, Alejandro Cabrera Sarmiento, and Santiago Sánchez-Solano. SPA vulnerabilities of the binary extended euclidean algorithm. *Journal of Cryptographic Engineering*, 7(4):273–285, November 2017.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *CHES 2004*, volume 3156 of *LNCS*, pages 16–29. Springer, Heidelberg, August 2004.

[BCP$^+$14] Lejla Batina, Lukasz Chmielewski, Louiza Papachristodoulou, Peter Schwabe, and Michael Tunstall. Online template attacks. In Willi Meier and Debdeep Mukhopadhyay, editors, *INDOCRYPT 2014*, volume 8885 of *LNCS*, pages 21–36. Springer, Heidelberg, December 2014.

[BCP$^+$19] Lejla Batina, Lukasz Chmielewski, Louiza Papachristodoulou, Peter Schwabe, and Michael Tunstall. Online template attacks. *Journal of Cryptographic Engineering*, 9(1):21–36, April 2019.

[BFMT16] Pierre Belgarric, Pierre-Alain Fouque, Gilles Macario-Rat, and Mehdi Tibouchi. Side-channel analysis of Weierstrass and Koblitz curve ECDSA on android smartphones. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 236–252. Springer, Heidelberg, February / March 2016.

[BH09] Billy Bob Brumley and Risto M. Hakala. Cache-timing template attacks. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 667–684. Springer, Heidelberg, December 2009.

[BH19] Joachim Breitner and Nadia Heninger. Biased nonce sense: Lattice attacks against weak ECDSA signatures in cryptocurrencies. In Ian Goldberg and Tyler Moore, editors, *FC 2019*, volume 11598 of *LNCS*, pages 3–20. Springer, Heidelberg, February 2019.

[BJ02] Eric Brier and Marc Joye. Weierstraß elliptic curves and side-channel attacks. In David Naccache and Pascal Paillier, editors, *PKC 2002*, volume 2274 of *LNCS*, pages 335–345. Springer, Heidelberg, February 2002.

[BJPW14] Aurélie Bauer, Éliane Jaulmes, Emmanuel Prouff, and Justine Wild. Horizontal collision correlation attack on elliptic curves. In Tanja Lange, Kristin Lauter, and Petr Lisonek, editors, *SAC 2013*, volume 8282 of *LNCS*, pages 553–570. Springer, Heidelberg, August 2014.

[BL] D. J. Bernstein and T. Lange. Explicit-Formulas Database. http://hyperelliptic.org/EFD/index.html Accessed on: Jun 02, 2020.

[Bro10] Daniel R. L. Brown. Standards for efficient cryptography (sec) 2: Recommended elliptic curve domain parameters (version 2.0). *Certicom Research, Certicom Corp*, 2010.

[BT11] Billy Bob Brumley and Nicola Tuveri. Remote timing attacks are still practical. Cryptology ePrint Archive, Report 2011/232, 2011. https://eprint.iacr.org/2011/232.

[BvSY14] Naomi Benger, Joop van de Pol, Nigel P. Smart, and Yuval Yarom. "ooh aah... just a little bit": A small amount of side channel can go a long way. In Lejla Batina and Matthew Robshaw, editors, *CHES 2014*, volume 8731 of *LNCS*, pages 75–92. Springer, Heidelberg, September 2014.

[CCJ04] Benoît Chevallier-Mames, Mathieu Ciet, and Marc Joye. Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity. *IEEE Transactions on computers*, 53(6):760–768, 2004.

[CFG$^+$12] Christophe Clavier, Benoit Feix, Georges Gagnerot, Christophe Giraud, Mylène Roussellet, and Vincent Verneuil. ROSETTA for single trace analysis. In Steven D. Galbraith and Mridul Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 140–155. Springer, Heidelberg, December 2012.

[Cor99] Jean-Sébastien Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In Çetin Kaya Koç and Christof Paar, editors, *CHES'99*, volume 1717 of *LNCS*, pages 292–302. Springer, Heidelberg, August 1999.

[CPB20] Alejandro Cabrera Aldaya, Cesar Pereida García, and Billy Bob Brumley. From A to Z: Projective coordinates leakage in the wild. *IACR TCHES*, 2020(3):428–453, 2020. https://tches.iacr.org/index.php/TCHES/article/view/8596.

[Deva] NewAE Technology, CW308T-STM32F. http://wiki.newae.com/CW308T-STM32F Accessed on: Jun 02, 2020.

[Devb] STMicroelectronics, STM32F405/415. https://www.st.com/en/microcontrollers/stm32f405-415.html Accessed on: Jun 02, 2020.

[Devc] Teledyne LeCroy, HDO6104A. http://teledynelecroy.com/oscilloscope/hdo6000a-high-definition-oscilloscopes/hdo6104a Accessed on: Jun 02, 2020.

[DGH$^+$16] Jean-Luc Danger, Sylvain Guilley, Philippe Hoogvorst, Cédric Murdica, and David Naccache. Improving the big mac attack on elliptic curve cryptography. In *The New Codebreakers*, pages 374–386. Springer, 2016.

[DHMP14] Elke De Mulder, Michael Hutter, Mark E. Marson, and Peter Pearson. Using Bleichenbacher's solution to the hidden number problem to attack nonce leaks in 384-bit ECDSA: extended version. *Journal of Cryptographic Engineering*, 4(1):33–45, April 2014.

[DLO$^+$19] Ibrahima Diop, Yanis Linge, Thomas Ordas, Pierre-Yvan Liardet, and Philippe Maurine. From theory to practice: horizontal attacks on protected implementations of modular exponentiations. *Journal of Cryptographic Engineering*, 9(1):37–52, April 2019.

[DPP20] Gabrielle De Micheli, Rémi Piau, and Cécile Pierrot. A tale of three signatures: Practical attack of ECDSA with wNAF. In Abderrahmane Nitaj and Amr M. Youssef, editors, *AFRICACRYPT 20*, volume 12174 of *LNCS*, pages 361–381. Springer, Heidelberg, July 2020.

[FWC16] Shuqin Fan, Wenbo Wang, and Qingfeng Cheng. Attacking OpenSSL implementation of ECDSA with a few signatures. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1505–1515. ACM Press, October 2016.

[GB17] Cesar Pereida García and Billy Bob Brumley. Constant-time callees with variable-time callers. In Engin Kirda and Thomas Ristenpart, editors, *USENIX Security 2017*, pages 83–98. USENIX Association, August 2017.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *CHES 2001*, volume 2162 of *LNCS*, pages 251–261. Springer, Heidelberg, May 2001.

[GPP$^+$16] Daniel Genkin, Lev Pachmanov, Itamar Pipman, Eran Tromer, and Yuval Yarom. ECDSA key extraction from mobile devices via nonintrusive physical side channels. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1626–1638. ACM Press, October 2016.

[GST14] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 444–461. Springer, Heidelberg, August 2014.

[GuHT$^+$20] Cesar Pereida García, Sohaib ul Hassan, Nicola Tuveri, Iaroslav Gridin, Alejandro Cabrera Aldaya, and Billy Bob Brumley. Certified side channels. In Srdjan Capkun and Franziska Roesner, editors, *USENIX Security 2020*, pages 2021–2038. USENIX Association, August 2020.

[HGS01] Nick A Howgrave-Graham and Nigel P. Smart. Lattice attacks on digital signature schemes. *Designs, codes and cryptography*, 23(3):283–290, 2001.

[HIM$^+$14] Johann Heyszl, Andreas Ibing, Stefan Mangard, Fabrizio De Santis, and Georg Sigl. Clustering algorithms for non-profiled single-execution attacks on exponentiations. In *Proc. International Conference on Smart Card Research and Advanced Applications – CARDIS 2013*, pages 79–93, Berlin, Germany, November 27–29, 2014.

[HKT15] Neil Hanley, HeeSeok Kim, and Michael Tunstall. Exploiting collisions in addition chain-based exponentiation algorithms using a single trace. In Kaisa Nyberg, editor, *CT-RSA 2015*, volume 9048 of *LNCS*, pages 431–448. Springer, Heidelberg, April 2015.

[HMH$^+$12] Johann Heyszl, Stefan Mangard, Benedikt Heinz, Frederic Stumpf, and Georg Sigl. Localized electromagnetic analysis of cryptographic implementations. In Orr Dunkelman, editor, *CT-RSA 2012*, volume 7178 of *LNCS*, pages 231–244. Springer, Heidelberg, February / March 2012.

[HMHW09] Michael Hutter, Marcel Medwed, Daniel Hein, and Johannes Wolkerstorfer. Attacking ECDSA-enabled RFID devices. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *ACNS 09*, volume 5536 of *LNCS*, pages 519–534. Springer, Heidelberg, June 2009.

[HMV04] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography.* Springer New York, 2004.

[IT02] Tetsuya Izu and Tsuyoshi Takagi. A fast parallel elliptic curve multiplication resistant against side channel attacks. In David Naccache and Pascal Paillier, editors, *PKC 2002*, volume 2274 of *LNCS*, pages 280–296. Springer, Heidelberg, February 2002.

[JB17] Kimmo Järvinen and Josep Balasch. Single-trace side-channel attacks on scalar multiplications with precomputations. In *Proc. International Conference on Smart Card Research and Advanced Applications – CARDIS 2016*, pages 137–155, Cannes, France, November 7–9, 2017.

[JMV01] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ecdsa). *International journal of information security*, 1(1):36–63, 2001.

[JSSS20] Jan Jancar, Vladimir Sedlacek, Petr Svenda, and Marek Sys. Minerva: The curse of ECDSA nonces. *IACR TCHES*, 2020(4):281–308, 2020. https://tches.iacr.org/index.php/TCHES/article/view/8684.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer, Heidelberg, August 1999.

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 104–113. Springer, Heidelberg, August 1996.

[KR13] Cameron F. Kerry and Charles Romine. Fips pub 186-4 federal information processing standards publication digital signature standard (dss). National Institute of Standards and Technology, NIST FIPS PUB 186-4, U.S. Department of Commerce, 2013.

[Liba] Bouncy Castle. https://www.bouncycastle.org/ Accessed on: Jun 02, 2020.

[Libb] Mbed TLS. https://tls.mbed.org/ Accessed on: Jun 02, 2020.

[Libc] OpenSSL. https://www.openssl.org/ Accessed on: Jun 02, 2020.

[LL94] Chae Hoon Lim and Pil Joong Lee. More flexible exponentiation with precomputation. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 95–107. Springer, Heidelberg, August 1994.

[MH20] Gabrielle De Micheli and Nadia Heninger. Recovering cryptographic keys from partial information, by example. Cryptology ePrint Archive, Report 2020/1506, 2020. https://eprint.iacr.org/2020/1506.

[MO09] Marcel Medwed and Elisabeth Oswald. Template attacks on ECDSA. In Kyo-Il Chung, Kiwook Sohn, and Moti Yung, editors, *WISA 08*, volume 5379 of *LNCS*, pages 14–27. Springer, Heidelberg, September 2009.

[MOP08] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks: Revealing the secrets of smart cards.* Springer Science & Business Media, 2008.

[MSEH20] Daniel Moghimi, Berk Sunar, Thomas Eisenbarth, and Nadia Heninger. TPM-FAIL: TPM meets timing and lattice attacks. In Srdjan Capkun and Franziska Roesner, editors, *USENIX Security 2020*, pages 2057–2073. USENIX Association, August 2020.

[NS02] Phong Q. Nguyen and Igor Shparlinski. The insecurity of the digital signature algorithm with partially known nonces. *Journal of Cryptology*, 15(3):151–176, June 2002.

[OC14] Colin O'Flynn and Zhizhang (David) Chen. Chipwhisperer: An open-source platform for hardware embedded security research. In *Proc. Constructive Side-Channel Analysis and Secure Design – COSADE 2014*, pages 243–260, Paris, France, April 13–15, 2014.

[RHAL92] Ronald L Rivest, Martin E Hellman, John C Anderson, and John W Lyons. Responses to nist's proposal, 1992.

[RSBD20] Niels Roelofs, Niels Samwel, Lejla Batina, and Joan Daemen. Online template attack on ECDSA: - extracting keys via the other side. In Abderrahmane Nitaj and Amr M. Youssef, editors, *AFRICACRYPT 20*, volume 12174 of *LNCS*, pages 323–336. Springer, Heidelberg, July 2020.

[Rya18] Keegan Ryan. Return of the hidden number problem. *IACR TCHES*, 2019(1):146–168, 2018. https://tches.iacr.org/index.php/TCHES/article/view/7337.

[SCM+18] Bo-Yeon Sim, Kyu Young Choi, Dukjae Moon, Hyo Jin Yoon, Jihoon Cho, and Dong-Guk Han. T_SM: Elliptic curve scalar multiplication algorithm secure against single-trace attacks. In *Proc. International Conference on Information Security Practice and Experience – ISPEC 2018*, pages 407–423, Tokyo, Japan, September25–27 2018.

[SH17] Bo-Yeon Sim and Dong-Guk Han. Key bit-dependent attack on protected PKC using a single trace. In *Proc. International Conference on Information Security Practice and Experience – ISPEC 2017*, pages 168–185, Melbourne, Australia, December13–15 2017.

[SHKS15] Robert Specht, Johann Heyszl, Martin Kleinsteuber, and Georg Sigl. Improving non-profiled attacks on exponentiations based on clustering and extracting leakage from multi-channel high-resolution EM measurements. In Stefan Mangard and Axel Y. Poschmann:, editors, *COSADE 2015*, volume 9064 of *LNCS*, pages 3–19. Springer, Heidelberg, April 2015.

[vSY15] Joop van de Pol, Nigel P. Smart, and Yuval Yarom. Just a little bit more. In Kaisa Nyberg, editor, *CT-RSA 2015*, volume 9048 of *LNCS*, pages 3–21. Springer, Heidelberg, April 2015.

[Wal01] Colin D. Walter. Sliding windows succumbs to big mac attack. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *CHES 2001*, volume 2162 of *LNCS*, pages 286–299. Springer, Heidelberg, May 2001.

[WSBS20] Samuel Weiser, David Schrammel, Lukas Bodner, and Raphael Spreitzer. Big numbers - big troubles: Systematically analyzing nonce leakage in (EC)DSA implementations. In Srdjan Capkun and Franziska Roesner, editors, *USENIX Security 2020*, pages 1767–1784. USENIX Association, August 2020.

[YB14] Yuval Yarom and Naomi Benger. Recovering OpenSSL ECDSA nonces using the FLUSH+RELOAD cache side-channel attack. Cryptology ePrint Archive, Report 2014/140, 2014. https://eprint.iacr.org/2014/140.