Reading It like an Open Book: Single-trace Blind Side-channel Attacks on Garbled Circuit Frameworks

Sirui Shen, Chenglu Jin
Centrum Wiskunde & Informatica, The Netherlands
{*sirui.shen, chenglu.jin*}*@cwi.nl*

***Abstract*—Garbled circuits (GC) are a secure multiparty computation protocol that enables two parties to jointly compute a function using their private data without revealing it to each other. While garbled circuits are proven secure at the protocol level, implementations can still be vulnerable to side-channel attacks. Recently, side-channel analysis of GC implementations has garnered significant interest from researchers.**

We investigate popular open-source GC frameworks and discover that the AES encryption used in the garbling process follows a secret-dependent sequence. This vulnerability allows private inputs to be exposed through side-channel analysis. Based on this finding, we propose a side-channel attack on garbled circuits to recover the private inputs of both parties. Our attack does not require access to any plaintexts or ciphertexts in the protocol and is single-trace, adhering to the constraint that a garbled circuit can be executed only once. Furthermore, unlike existing attacks that can only target input non-XOR gates, our method applies to both input and internal non-XOR gates. Consequently, the secrets associated with every non-XOR gate are fully exposed as in an open book. Fundamentally, this work challenges the standard non-collusion assumption in multi-party computation, arguing for the necessity of extending it to the physical layer.

We comprehensively evaluate our attack in various scenarios. First, we perform the attack on single-platform software implementations of standard AES and interleaved AES on a 32-bit ARM processor, achieving a 100% success rate in both cases. Next, we target a hardware implementation on a Xilinx Artix-7 FPGA, where the resolution of power consumption measurements and the number of samples are significantly limited. In this scenario, our attack achieves a success rate of 79.58%. Finally, we perform a cross-platform attack on two processors with different microarchitectures representing the two parties. The differing execution cycles and power sensors across the platforms increase the difficulty of side-channel analysis. Despite these challenges, our point-of-interest (POI) selection method allows our attack to achieve a 100% success rate in this scenario as well. We also discuss effective countermeasures that can be readily applied to GC frameworks to mitigate this vulnerability.

This publication is part of the project CiCS of the research programme Gravitation which is (partly) financed by the Dutch Research Council (NWO) under the grant 024.006.037.

***Index Terms*—Secure multi-party computation, Garbled Circuits, Side-channel analysis, Single-trace attack, Cross-platform attack, FPGA.**

1. Introduction

Secure Multi-Party Computation (MPC), a trending research area in modern cryptography, allows multiple parties to jointly compute a function f while safeguarding the privacy of their respective inputs. MPC protocols enable multiple parties to evaluate f by exchanging their information without reliance on additional trusted parties. The goal of MPC can be achieved through techniques like secret sharing [1], [2], homomorphic encryption [3], [4], or garbled circuits [5]–[7], etc. MPC protocols have a wide range of privacy-preserving applications, such as data analysis [8]–[10], outsourcing computation [11], and federated learning [12], [13].

The two-party setting of MPC, known as *secure two-party computation* (2PC), has received vast attention from researchers. *Garbled circuit* (GC) protocol [14], [15] is used in the first 2PC implementation [5]. The GC protocol can evaluate any functions that can be represented by a Boolean circuit while providing security against semi-honest adversaries. There are two parties in the GC protocol: a garbler and an evaluator. The garbler encodes a Boolean circuit into a garbled circuit, where the bit value 0 or 1 of each wire in the original circuit is represented by a random label (e.g., a 128-bit random bit-string). Then, the garbled circuit and the labels corresponding to the garbler's input are sent to the evaluator. The evaluator gets the labels corresponding to its own input through oblivious transfer and uses all the available input labels to evaluate the garbled circuit till getting the final output value. By encoding the real values as random labels for circuit evaluation, the evaluator cannot infer the real value of a wire label observed during the evaluation, and thus, the privacy of each party's private input is preserved. Due to the flexibility of implementing any functions as Boolean circuits, the GC protocol is widely adopted in real-world applications [16]–[19].

To further reduce the computation, memory, and communication costs of the GC protocol, researchers have introduced a variety of optimizations, such as [6], [20]–[24]. These optimizations make the GC more efficient and practical while not sacrificing security. Hence they have been

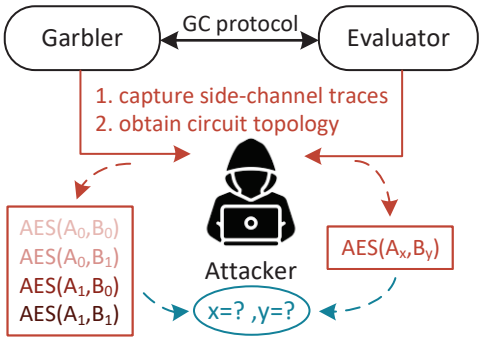

Figure 1. Adversarial model.

implemented in a series of GC frameworks [6], [7], [25]–[31].

Although GC and their related optimizations are theoretically secure at both the primitive and protocol levels, practical implementations may introduce additional vulnerabilities. A class of attacks known as *Side-channel attack* (SCA) exploits side-channel information, such as power consumption [32], timing [33], or electromagnetic emissions [34], to recover secrets from a target system. Two studies [35] and [36] published at TCHES 2023 and ACNS 2024 demonstrated that GC implementations can suffer from SCA. The former uses power analysis to attack a GC implementation to get the global offset R, which is used in a popular optimization, FreeXOR; the latter utilizes timing information to extract the garbler's input directly. Despite the considerable research on the theoretical security and performance of GC implementations, their side-channel resistance still requires thorough investigation.

In this paper, we investigate and review the code of seven open-source GC frameworks and find that the order of AES execution in each gate's garbling process is secret-dependent. Also, in popular GC constructions, the evaluator needs to perform one of the four/two AES operations performed by the garbler with exactly the same key and plaintext in order to evaluate a garbled gate. Hence, by figuring out which AES operation among the four/two performed by the garbler is repeated on the evaluator side, we can interpret the position of the operation in a sequence as private input of the gate due to the secret-dependent AES sequencing. We propose an attack that uses side-channel information to distinguish which plaintext in the garbling process is also used by the evaluator. Fig. 1 illustrates how an attacker can obtain private information x and y from a side channel on a garbled circuit implementation. It is worth noting that, unlike the former SCA attack [35], our attack targets the gate encryption scheme widely used in popular GC frameworks.

We assess the effectiveness of our attack on multiple implementations. First, as a proof of concept, we attack a software implementation on a 32-bit ARM processor, achieving a 100% success rate. Next, we evaluate our attack on a hardware accelerator implementation of AES; we deploy AES hardware implementations and self-built on-chip sensors on two FPGA boards, mimicking a remote SCA that attacks a hardware accelerator within a cloud FPGA [37]. In this scenario, the number of samples per trace is extremely low due to the short execution time of an AES hardware accelerator and the low sampling rate of a self-built FPGA sensor. Still, our attack achieves a success rate of 79.58% under these challenging conditions. Finally, we demonstrate our attack in a more general setting where the garbler and the evaluator are implemented on processors with different microarchitectures. In this cross-platform scenario, different AES execution times and architectural leakage models make it more challenging for attackers to discover similarities between the traces of the two parties. However, under these tough conditions, our attack still manages to achieve a 100% success rate, leveraging our point-of-interest (POI) selection method. The high success rate of the cross-platform attack shows the wider applicability and severity of our attack in practical scenarios.

The main contributions of our work include:

- We propose a single-trace blind SCA on widely used GC frameworks. This method can extract the private input values of all non-XOR gates, including both input gates and internal gates within a GC, irrespective of whether they are the garbler's inputs or the evaluator's inputs. To the best of our knowledge, our attack is the first demonstration of breaking the non-collusion assumption in secure multi-party computation by a malicious third party who does not need to actively engage with the protocol participants.
- We evaluate our attack on both software and hardware implementations, achieving high success rates of 100% and 79.58%, respectively.
- To show the wide applicability of our attack in practice, we develop a novel cross-platform attack between two software implementations on two processors with different microarchitectures and still achieve a 100% success rate.
- We review the code of seven open-source GC frameworks and find that the vulnerability exists in all of them.
- We make the code and dataset used in our experiments open source on GitHub[1][2] and Figshare[3], respectively.
- We discuss potential countermeasures against the proposed attack with almost no performance overhead.

The paper is organized as follows. Sec. 2 introduces the background knowledge about GC and SCA. Sec. 3 defines the adversarial model and provides a detailed description of the proposed attack. In Sec. 4, we evaluate the proposed attack in three different scenarios. Sec. 5 discusses some potential countermeasures. Finally, Sec. 6 delves into the related works, and Sec. 7 concludes this work and suggests

1. https://github.com/NomadShen/chipwhisperer
2. https://github.com/NomadShen/fpga_based_attack
3. https://doi.org/10.6084/m9.figshare.27284547.v1

future research. Appendix A provides an additional evaluation of the attack on an interleaved AES implementation.

2. Background

2.1. Garbled Circuits

2.1.1. Overview. We recall the abstraction of a garbling scheme introduced in [38]. A garbling scheme is defined as a tuple of algorithms $\mathsf{GS} = (\mathsf{Gb}, \mathsf{En}, \mathsf{Ev}, \mathsf{De})$ where Gb is probabilistic and the rest are deterministic. The garbling algorithm Gb takes input 1^k and a Boolean circuit f and outputs a tuple (F, e, d), where F is a garbled circuit, and e and d are encoding and decoding information, respectively. The encoding algorithm En uses e to encode a suitable input x for f into a garbled input X. The evaluation algorithm Ev takes (F, X) as above and outputs a garbled output Y. Finally, the decoding algorithm De takes (d, Y) to obtain the plaintext output y. The correctness condition of the garbling scheme is that if $(F, e, d) \leftarrow \mathsf{Gb}(1^k, f)$, then $\mathsf{De}(d, \mathsf{Ev}(F, \mathsf{En}(e, x))) = f(x)$. For the security analysis of GC, readers can refer to [38], [39].

The classic setting of the garbling scheme for 2PC has two parties: a garbler and an evaluator. The garbler executes Gb and En algorithm. Two random bit-strings called *wire labels* are chosen to represent the semantic 0 or 1 on each wire in the Boolean circuit. Then, the truth table of each gate is garbled, i.e., each output wire label is encrypted by the combination of two corresponding input wire labels, according to the input-output relationship in the truth table. Hence, a two-input Boolean gate will be converted to a garbled truth table with four entries. The concrete encryption scheme can be realized with pseudo-random functions (PRF), pseudo-random permutation (PRP), or correlation robust hash function [23]. The evaluator runs the Ev algorithm. It receives the wire labels corresponding to the garbler's private input and uses oblivious transfer to get the wire labels corresponding to its own input, which means the evaluator only knows one label per wire. To evaluate a gate in GC, only one correct entry of the garbled truth table is decrypted using the only combination of input labels known by the evaluator. Then, the evaluator uses the generated output labels as input labels of the following gates and evaluates the gates one by one according to the circuit topology. In the end, the De algorithm decodes the final output wire label into its plaintext. The De can be run by either the garbler or the evaluator, depending on the application, and the result is typically shared with the other party. **Informally, the privacy guarantee provided by a GC protocol is that the collection (f, F, X, d) does not reveal any more information about x than the final result $f(x)$.** Later, we will demonstrate how we attack the privacy guarantee of GC protocols and recover x from side-channel information directly.

2.1.2. Gate Encryption Schemes. The encryption scheme in a garbling process hides an output wire label C in a ciphertext produced by two input wire labels A and B.[4] The gate encryption scheme keeps evolving in the prior works [6], [21], [40], [41]. In [21], the encryption is realized as $PRF_A(*) \oplus PRF_B(*) \oplus C$. Lindell, Pinkas, and Smart reduced the number of calls to the PRF per gate from two to one [40] as $H(A \,\|\, B, C) \oplus C$, where H is a hash function. [41] proposed a scheme $H(A \,\|\, B \,\|\, T) \oplus C$ where a publicly known tweak T is introduced, and the hash function is instantiated as $H(A \parallel B \parallel T) = AES256_{A\|B}(T)$. Inspired by previous work, Bellare et al. [6] proposed the following scheme:

$$\begin{aligned} E^{\pi}(A, B, T, C) &= \rho(2A \oplus 4B \oplus T) \oplus C \\ \rho(x) &= \pi(x) \oplus x \end{aligned} \tag{1}$$

where a fixed-key AES acts as the primitive π as a pseudo-random permutation. A fixed-key AES means that the AES uses the same key for garbling and evaluating every gate, and the key is publicly known by both parties and attackers. This scheme's security is proved in [6], and since then, it has been widely adopted in popular GC frameworks [6], [7], [25]–[27], [30]. In the gate encryption scheme, each combination of values A and B corresponds to one entry in the truth table; thus, the term "entry" is used in this paper to denote a single ciphertext in the garbling process.

2.1.3. Popular Optimizations. To improve the performance of GC implementations, researchers proposed a series of optimization techniques to reduce the computation or communication cost while not sacrificing the security of the protocol.

The *point-and-permute* (P&P) optimization, proposed in [20], introduced a select bit to each wire label. The value of a select bit is independent of the wire's semantic value, so the secret is not revealed by the select bit. The select bit can point out the correct entry of the garbled truth table, allowing the evaluator to decrypt one correct entry at each gate instead of decrypting all four entries one by one and looking for the correct one.

The *garbled row reduction* (GRR3) optimization reduces the size of transferred ciphertext per gate. The idea of GRR3 is to set one of the four output encrypted entries of a gate to be all-zero, meaning set $C = H(A \parallel B)$ [21]. Thus, the garbler only needs to send three ciphertexts instead of four per gate.

In the *FreeXOR* optimization, a global offset R is introduced, and the two complementary labels of a wire can be written as A and $A \oplus R$ [22]. Hence, an XOR gate can be computed as the XOR result of the two input labels and does not use a garbled truth table.

The *half-gate* optimization further reduces the transferred ciphertext size of the non-XOR gates from three to two [23]. Taking an AND gate as an example, an AND

4. This process is also called Garbling Scheme in some literature. In this paper, we follow the naming convention in [23] and call it Encryption Scheme.

gate is split into two "half gates." The garbler generates two ciphertexts for one half gate as:

$$\begin{aligned} &PRF(B) \oplus C \\ &PRF(B \oplus R) \oplus C \oplus aR \end{aligned} \tag{2}$$

Similarly, the garbler generates two ciphertexts for the other half gate:

$$\begin{aligned} &PRF(A) \oplus C \\ &PRF(A \oplus R) \oplus C \oplus B \end{aligned} \tag{3}$$

Applying GRR3 on each half gate, we can reduce the number of transferred ciphertexts per AND gate to just two.

Some optimizations further improved the performance of GC, like *fleXor* [42] and *three halves* [24]. However, they are less common in popular GC frameworks.

The effects of optimizations on our attack. The optimizations described above, except FreeXOR and half-gate, reduce the amount of ciphertext transmitted or the computational load on the evaluator. Our attack focuses on the secret-dependent order of operations issue when the garbler generates garbled truth tables, so these optimizations do not affect our attack. The FreeXOR optimization method eliminates the need for a garbled truth table for XOR gates, so our attack cannot work on XOR gates if FreeXOR is used. The half-gate optimization transforms every non-XOR gate into two half gates, and thus the encryption scheme becomes Eq. 2 and 3, instead of Eq. 1. However, within each half gate, the vulnerability still exists, so our attack can adapt and still work.

2.1.4. Popular GC Frameworks. Various open-source GC frameworks are developed to realize the GC protocol and incorporate the above optimizations. **JustGarble** is the first GC framework that adopts a fixed-key AES as PRF and demonstrates high efficiency [6]. Based on JustGarble, **TinyGarble** adds the half-gate optimization and proposes a flow from circuits described in Verilog HDL to garbled circuits, which facilitates its adoption in practice [25]. **Obliv-C** is a GCC wrapper that allows people to embed MPC inside regular C programs [30]. **ABY** combines secret sharing and GC together to achieve a solution in the 2PC scenario [27], and **ABY**^{3} extends the framework to the three parties setting [29]. **EMP-toolkit** is an efficient multi-party computation toolkit that contains GC and zero-knowledge proof protocols [7]. **TinyGarble2** is a C++ framework privacy-preserving computation based on EMP-toolkit [26]. **MOTION** is a generic open-source framework for mixed-protocol MPC that achieves excellent communication efficiency [31]. Table. 1 summarizes the PRF instantiation and optimization techniques relevant to our attack in these popular GC frameworks. Since all frameworks use AES as the PRF, the targets of the attacks in this paper also employ AES as the PRF primitive. For each gate, the key for the AES is the same on both the garbler's and the evaluator's sides, regardless of whether it is a fixed-key AES or a re-keying AES.

TABLE 1. THE PRF AND OPTIMIZATIONS USED IN THE LATEST VERSION OF POPULAR GC FRAMEWORKS AS OF MAY 28, 2024.

Framework	PRF	FreeXOR	Half-gate
JustGarble [6]	fixed-key AES	✓	×
TinyGarble [25]	fixed-key AES	✓	✓
ABY [27]	fixed-key AES	✓	×
ABY^3 [29]	fixed-key AES	✓	✓
Obliv-C [30]	fixed-key AES	✓	✓
EMP-toolkit [7]	re-keying AES[a]	✓	✓
MOTION [31]	fixed-key AES	✓	×[b]

[a] EMP-Toolkit uses a re-keying AES for better concrete security [43], i.e., the key is different per gate. This does not affect the effectiveness of our attack because all the keys are publicly known, and each gate is attacked independently.

[b] MOTION adopts the *three halves* variance in [24], but the secret-dependent order of operations issue also exists.

2.2. Power Side Channel Attacks

2.2.1. Overview. *Side-channel attack* is an effective and non-invasive attack method against implementations of mathematically strong cryptographic algorithms. Power analysis is an important class of SCA where attackers can derive secret information by analyzing the power consumption of hardware devices. In a power SCA, attackers usually use a sensor, e.g., an oscilloscope, to capture the power information that varies over time called *power trace* and then statistically analyze the trace to extract the leakage.

The idea of power analysis was first introduced to cryptographic researchers publicly as differential power analysis (DPA) in 1998 [44]. Following DPA, Eric et al. proposed correlation power analysis (CPA) [45], which is more stable and efficient. The CPA scheme calculates the Pearson correlation coefficient between the actual and the hypothesized values as the likelihood metric. The Pearson correlation coefficient $corr(x, y)$ between two traces x and y with the same length is defined as

$$corr(x,y) = \frac{\sum_{i=0}^{n-1}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=0}^{n-1}(x_i - \bar{x})^2}\sqrt{\sum_{i=0}^{n-1}(y_i - \bar{y})^2}} \tag{4}$$

where x_i and y_i are the i-th samples in the trace x and trace y, respectively. DPA and CPA are model-based attacks where the attacker is required to have some basic knowledge of the implementation. Typically, the attacker uses a leakage model, e.g., hamming weight, to estimate the power consumption of the hypothesized intermediate value and compares it with the collected traces. Finally, the key is extracted from the noisy trace via a distinguisher.

The power SCA has been shown feasible to attack various cryptographic implementations [46]–[48]. Given the limited amount of work on SCA on GC implementations, research in this area deserves broader attention to enhance the implementation security.

3. Proposed Attack

3.1. Observation

Power side-channel leakage is generally due to data-dependent power consumption patterns. Most existing power SCAs focus on discovering the relationship between the secret key and the power traces, but we focus on the data-dependent execution patterns in the implementation and discover the following insights:

- When the P&P optimization is implemented, the evaluator decrypts only one entry per gate, as indicated by the P&P select bits, out of the four entries (or two entries in the case of half-gate optimization) encrypted by the garbler.
- As shown in Eq. 1, 2, and 3, decrypting the output label C requires calling the same PRF (AES encryption) with the exact same key and data (A and/or B) as in its corresponding encryption.

Hence, by checking which PRF call happens on both the garbler's and the evaluator's sides with the same data, we can discover the semantic values of the two input wires of the gate. This is because the order of the four/two PRF calls for a (half-)gate is determined by the semantic values of the input wires in the garbling process. Hence, one just needs to use side-channel information to find the same PRF call that happens on both sides. Our attack is applicable to all gates implemented as garbled truth tables. This also means that our attack only works on non-XOR gates if XOR gates are not implemented as garbled truth tables in the case of FreeXOR.

Based on the insights above, we review the code of seven popular open-source GC frameworks [6], [7], [25], [27], [29]–[31] to investigate their vulnerability to our attack. All the frameworks use the same gate encryption scheme (Eq. 1) with P&P, GRR3, and FreeXOR optimization. Half-gate optimization is absent in JustGarble and ABY, though it is included in other frameworks. We notice that the garbler's encryption of entries is performed in a secret-dependent order within every gate. Take the following code snippet from the garbling procedure in ABY^3 framework as an example:

```
1 hash[0] = (a << 1) ^ tweaks[0];
2 hash[1] = ((a ^ mGlobalOffset) << 1) ^ tweaks[0];
3 hash[2] = (b << 1) ^ tweaks[1];
4 hash[3] = ((bNot)<< 1) ^ tweaks[1];
5 mAesFixedKey.ecbEncBlocks<4>(hash, temp);
6 hash[0] = hash[0] ^ temp[0]; // H( a0 )
7 hash[1] = hash[1] ^ temp[1]; // H( a1 )
8 hash[2] = hash[2] ^ temp[2]; // H( b0 )
9 hash[3] = hash[3] ^ temp[3]; // H( b1 )
```

The ABY^3 framework employs half-gate optimization, two complementary wire labels `a` and `a^mGlobalOffset` are encrypted in a "half gate," and a full gate consists of two half gates. The wire labels are copied into an array `hash` after multiplying with constants and adding tweaks [6]. Then, a fixed-key AES cipher is called to encrypt all the labels sequentially in line 5. If the attacker can determine which label's AES operation was also performed by the evaluator in the evaluation process, he/she can determine which item in the array `hash` is used in the circuit, thereby outputting the corresponding index as the private input information of the gate. **Essentially, the attack boils down to whether one can distinguish a repeated AES execution from three/one AES executions with random inputs only based on their side-channel information.**

Another observation is that in all frameworks, the garbling and evaluation processes are executed in the order of gate indexes, meaning that attackers can easily divide the power trace of the whole garbling or evaluation process into sub-traces associated with each gate using public information about the circuit.

We examine all seven open-source GC frameworks in Table 1 and find that the secret-dependent order of operations issue exists in all the frameworks[5].

3.2. Adversarial Model

Fig. 1 illustrates the adversarial model in our attack. We assume that the adversary $\mathcal{A}$ has the following capabilities:

- $\mathcal{A}$ has access to both the garbler's and the evaluator's side-channel traces during the garbling and evaluation process either physically [49] or remotely [50].[6]
- $\mathcal{A}$ knows the function f computed in the garbled circuit, which is public in a GC protocol. This allows $\mathcal{A}$ to partition a long trace into small traces corresponding to the AES encryptions for each gate.
- $\mathcal{A}$ is supposed to have basic knowledge of the two parties' devices and get hold of some devices with the same architecture to profile the POIs. Note that the profiling process does not need the victim's devices and can be done completely offline.

Due to the properties of the GC protocol, $\mathcal{A}$ is also subject to the following constraints:

- $\mathcal{A}$ can obtain only one trace for each garbled circuit since classical garbled circuits cannot be reused.
- $\mathcal{A}$ can not eavesdrop on or manipulate any data and, therefore, can only launch a blind attack.

In practical scenarios, $\mathcal{A}$ could be a cloud service provider with physical access to both parties's servers, allowing for the acquisition of side-channel information of the two parties. As another example, in an application where both parties of GC run on the same processor to provide a confidential Function-as-a-Service platform [51], the platform owner can easily get the side-channel traces of both

5. We found the *yaoGenerateGate* function in Obliv-C does not have the secret-dependent order of operations, which will be discussed later in Sec. 5. The more frequently used *yaoGenerateHalfGatePair* function still suffers from this issue.

6. The side-channel information here can be power, electromagnetic emission, cache access, etc., as long as the attacker can distinguish AES executions with different plaintexts. In this paper, we demonstrate our attack using power side-channel leakage.

Attacking Phase 1 Profiling the POIs.

1: **Prepare:** collect two groups of traces pg and pe of AES execution with the same inputs from the attacker's devices that have the same architecture as the victims'. pg_i or pe_i represents a vector composed of the points with the same index i on all the traces in pg or pe, arranged in the order of the traces.
2: **Choose POIs of the evaluator:** divide the traces into small segments of length l_s, select k_{POI} points with the highest variance in each segment.
3: **Choose POIs of the garbler:** for each POI of the evaluator, calculate the correlations between it and every pg_i in the garbler's trace and choose the index i with the highest absolute value of the correlation as the corresponding POI on the garbler's trace.
4: **Filter POIs:** only keep the POIs with absolute correlation values greater than a threshold t_{POI}.

Attacking Phase 2 Attacking N non-XOR gates.

1: **Capture traces:** obtain one power trace each from the garbler and the evaluator running a GC protocol together using their private inputs.
2: **Partition:** divide the traces into sub-traces t_garb_i and t_eval_i associated with each gate, where i denotes the gate index.
3: **Condense the traces:** according to the index of the POIs, select feature points to create a feature vector from each sub-trace. For every gate i, the garbler has four (or two in the case of half-gate) feature vectors of AES execution $f\_garb_{i,j}$ and the evaluator has only one feature vector of AES execution f_eval_i.
4: **Calculate the means:** calculate the mean of $f\_garb_{i,j}$ for all i, j and get avg_garb. Similarly, calculate the mean of f_eval_i for all i and get avg_eval.
5: **for each** $i \in [0, N)$ **do**
6: **Compare the traces:** calculate $corr_{i,j} = corr(f\_garb_{i,j} - avg\_garb, f\_eval_i - avg\_eval)$ for every j and find the index of the maxima $k_{i,max}$.
7: **Extract the input value:** based on the execution order of AES in the garbling process, convert $k_{i,max}$ into the input values x_i and y_i of gate i.
8: **end for**

parties because the garbler and the evaluator are effectively running on the same processor. Another interesting example is presented in [52], where garbled circuit protocols are executed by three parties: a garbled circuit generator, several garbled input providers, and a garbled circuit evaluator. Our attack can steal the private input providers' data solely by gaining side-channel access to the circuit generator and the evaluator, neither of whom knows the private inputs.

In addition to the traditional method of obtaining side-channel information through physical access, recent studies have shown that remote SCAs are also feasible in practice [37], [53]–[56]. Thus, $\mathcal{A}$ can capture side-channel information from both parties remotely if it gains a foothold on both servers. This remote measurement capability significantly expands the applicable scenarios for our attack. We will also demonstrate one cloud FPGA-based remote attack in our experimental evaluation.

3.3. Detailed Description of the Attack

In our attack, the attacker aims to obtain the input value of all non-XOR gates in a garbled circuit. We assume the victim GC implementation executes AES encryption according to the order of the actual values on the input wires and using P&P optimization,[7] as in all the reviewed seven GC frameworks.

The steps of the offline profiling stage and online attacking stage of our attack are listed in Attack Phase (AP) 1 and 2, respectively. They target a garbled circuit with N non-XOR gates.

The first part is offline profiling of POIs. As shown in AP 1, the attacker generates k_{pre} random inputs for a fixed-key AES and captures two groups of power traces, pg and pe, from AES executions with these random inputs on devices that share the same architecture as the garbler's and the evaluator's, respectively. Then, the attacker first selects the POIs on the evaluator's trace. It can be done either manually or by dividing the traces into small segments of length l_s and choosing the k_{POI} points with the highest variance in each segment. Afterward, for each vector, pe_p of a POI index p, calculate its Pearson correlation with all pg_i vectors and take the index i with the highest absolute correlation value as the corresponding POI on the garbler's trace. The computational complexity of profiling for two groups of n traces, each of length m, is $O(nm)$. As we will show later, the profiling stage makes our attack more efficient and adaptable to more general scenarios, including the cross-platform scenario.

Next, in AP 2, the adversary performs the online attack and captures two power traces from the garbler's and evaluator's devices, respectively, when they are running GC with their private inputs. The traces are aligned and divided into sub-traces associated with each gate. The sub-traces are further condensed into feature vectors based on the selected POIs. If the sign of the correlation of a pair of POIs on both sides is negative, the sign of the garbler's value at the POI is inverted. This guarantees that all pairs of points at the same positions on the two feature vectors are positively correlated. For each gate i, the garbler has four (or two in the case of half-gate) feature vectors of AES execution $f\_garb_{i,j}$, and the evaluator only has one vector f_eval_i.

Finally, the attacker uses the feature vectors from two parties to extract private information. For gate i, the Pearson correlation coefficient is calculated between every $f\_garb_{i,j}$ and f_eval_i after each of them subtracting the average trace, and the index $k_{i,max}$ of the largest correlation is found. The computational complexity of attacking power traces of

7. Note that even without P&P optimization, our attack may still work if the attacker can figure out when the evaluator has found the correct output label. However, since P&P is adopted by all GC frameworks, we do not investigate the scenario without P&P.

TABLE 2. COMPARISON BETWEEN OUR ATTACK AND TWO RELATED ATTACKS.

Attack	Side-channel	Data required	Process being attacked	Target	Vulnerability in	Avg. SR
[35]	Power	Garbler's labels	Garbling	Global offset R	Input non-XOR gates	$\approx 70\%$
Goblin [36]	Timing	None	Wire label generation	Garbler's input	Input wire labels	$> 90\%$
Our attack	Power	None	Garbling & Evaluation	Both parties's input	All non-XOR gates	100%/79.58%

length m is $O(m)$ per gate. Based on the secret-dependent execution order of AES, the two input values x_i and y_i of the gate can be inferred from $k_{i,max}$. When the gate is an input gate, the leaked input values tell both parties' input. Additionally, a confidence value α_i for each gate can be calculated as $\alpha_i = max\_corr_i - second\_max\_corr_i$, where the max_corr_i and $second\_max\_corr_i$ are the largest and the second largest correlation between the garbler's trace and the evaluator's trace for gate i. α_i can help the attacker evaluate the reliability of the attacking result on a particular gate.

Possible Improvements. Because our attack can attack both input gates and internal gates, the adversary can exploit the known circuit and the confidence values α_i of every gate to cross-validate and correct the wrong guesses. For example, the attacker can start with the guesses with a high confidence value α_i and use an SAT solver to help cross-validate the low confidence guesses. We will show a strong correlation between high confidence values and high success rates in the experimental evaluation.

Comparison with related attacks. To the best of our knowledge, there are only two prior works on SCA on GC [35], [36]. For more description of them, readers can refer to Sec. 6. Table 2 compares the two attacks and our attack. The two existing attacks can only attack the input non-XOR gates or the input wire label generations, whereas our attack can target all non-XOR gates within the circuit, including the internal ones. When the success rate cannot achieve 100%, only our attack can validate and correct the results by inferring the circuit inputs from the internal values and the circuit.

3.4. Performance Metric

The success rate metric is used to measure the performance of the proposed attack. We adapt the success rate definition in [57] to our setting. In an attack against a garbled circuit with n_{entry} AES executions per gate, we assume the actual input of a gate as s, the sorted guess of the adversary from the highest to the lowest as $g = [g_1, g_2, .., g_{n_{entry}}]$. The successful rate of order o, SR^o, is defined as:

$$SR^o = Pr(s \in [g_1, g_2, ..., g_o]) \tag{5}$$

In a simple term, SR^o is the likelihood of the correct result among the top o most likely guesses. SR^1 is assumed when o is not specified.

Furthermore, we also calculate the average significance avg_sig of the attack results, which is defined as:

$$avg\_sig = \frac{\sum_{i=1}^{n_{gate}} (corr_{correct} - max\_corr_{incorrect})}{n_{gate}} \tag{6}$$

where $corr_{correct}$ denotes the correlation corresponding to the correct pair of traces and $max\_corr_{incorrect}$ represents the maximum correlation of the other incorrect pairs of traces. This quantifies the difficulties of distinguishing a correct pair from all the other incorrect ones.

4. Implementation and Evaluation

4.1. Attack Modeling

The essence of the proposed attack relies on a secret-dependent order of AES executions in the garbler. By matching the AES execution in the evaluator with that in the garbler, we can determine which entry is being evaluated, obtain the inputs to the gate, and finally infer the private information of both the garbler and the evaluator. This indicates that the core of the attack is to find the same AES encryption on the two parties based on power consumption.

We use real power traces measured on devices to verify the feasibility of the proposed attack. The two encryption schemes we will attack include a standard implementation with four entries per gate (as in Eq. 1) and an implementation of half-gate optimization with two entries per half-gate respectively referred to as the *circuit with full gates* and the *circuit with half gates*, which covers all popular encryption schemes except three halves in one framework.

Using public information about the circuit, attackers can divide the obtained power traces into segments corresponding to each AES operation in each gate. As the same in [35], we assume that the trace segments corresponding to each gate and each AES operation are synchronized and aligned for simplicity.

For every gate, we first generate n_{entry} random plaintexts as the input wire labels, where $n_{entry} = 4$ in the standard implementation and $n_{entry} = 2$ in the half-gate implementation. Then, we use a fixed-key AES to encrypt all the plaintexts on the garbler's device and encrypt a randomly selected plaintext out of the n_{entry} plaintexts on the evaluator's device. During the execution of AES, we measure the power consumption of both parties. Next, we find the POIs and distinguish which one of n_{entry} garbler's trace corresponds to the evaluator's trace.

4.2. Evaluation on Software Implementation

We first consider a scenario where both the garbler and the evaluator run a software implementation of AES on processors with the same architecture. This allows the attacker to easily align the collected power traces and use the same POIs for both parties.

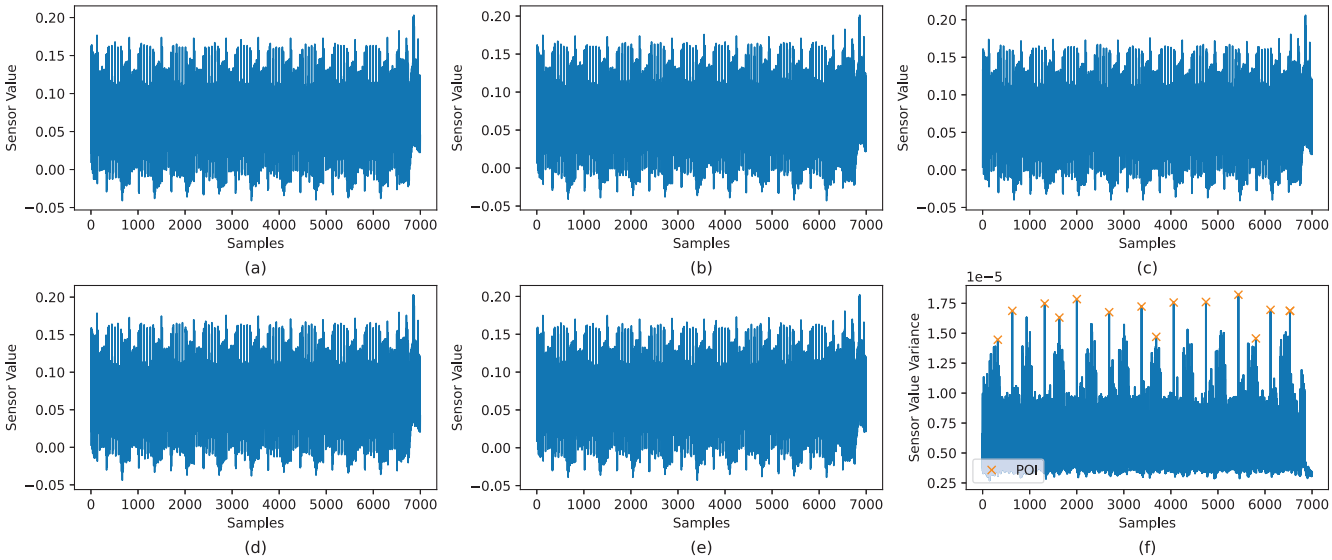

Figure 2. The power consumption and variance of the garbler's and evaluator's fixed-key AES executions at one gate on a 32-bit ARM processor: (a) Power consumption of the garbler's entry0 (b) Power consumption of the garbler's entry1 (c) Power consumption of the garbler's entry2 (d) Power consumption of the garbler's entry3 (e) Power consumption of the evaluator (f) Variance of the power consumption and the chosen POIs.

Experimental Setup. The attack is first performed on a software implementation on the 32-bit ARM processor. As a proof of concept, we use the ChipWhisperer-Lite platform to conduct our experiments.[8] A Tiny-AES C program is loaded on an STM32F303RBT6 processor with a randomly selected fixed key. On the same board, an ADC chip samples the voltage once every clock cycle.

Attack Details. We first collect 200 power traces of the fixed-key AES executed by the two parties. Note that the traces for profiling are independent of the traces for the subsequent attack. Fig. 2a, 2b, 2c, 2d, and 2e show five traces corresponding to four AES calls on the garbler and one AES call on the evaluator, respectively. Despite having different AES plaintexts among Fig. 2a, 2b, 2c, and 2d, the matched trace with that in Fig. 2e cannot be directly found by observing the trace as in *simple power analysis* (SPA).

A straightforward comparing method that uses all 7000 points across the entire trace turns out to be redundant and inefficient; therefore, we calculate the variance of each point in the collected traces and select POIs based on step 2 in AP 1. The variance is shown in Fig. 2f. In this setting, the garbler's and the evaluator's devices have the same architecture, so we can attack both parties with the same set of POIs rather than choosing POIs separately for the garbler as step 3 in AP 1. If we set $l_s = 500$ and $k_{POI} = 1$, the chosen POIs are represented as the orange $\times$ symbols in Fig. 2f.

Experimental Result. We generate data for 2000 full gates and collect the power consumption data from the ChipWhisperer-Lite. Following AP 1 and 2, we perform attacks with different l_s and k_{POI} in the same data set. The attack results[9] are given in Table 3.

TABLE 3. THE ATTACK RESULTS OF THE SINGLE-PLATFORM EVALUATION ON CHIPWHISPERER-LITE.

l_s	k_{POI}	n_{POI}	$SR(\%)$	avg_sig
1	1	7000	92.35	0.303
50	1	140	100	0.635
50	5	700	100	0.626
500	1	14	98.7	0.541
500	10	140	100	0.670
1000	1	7	89.9	0.417

We successfully achieve a 100% success rate in three parameter sets, which means all the inputs of every non-XOR gate in a garbled circuit are correctly recovered by us. In these sets, the average significance reaches up to 0.670, which indicates that the correlation values between the matched traces are significantly higher than the unmatched ones, making the distinction quite clear.

The SR and avg_sig drop if we reduce the number of POIs aggressively. However, if we simply use the whole trace for the attack, the SR and avg_sig also worsen. This is because not all the points in a trace are data-dependent, and including the data-independent points introduces noise and leads to poorer results. This suggests the necessity of the profiling stage in our attack.

8. Due to chip shortage, we only have one board, which acts as both the garbler's and the evaluator's device. However, as we will demonstrate later, we achieve a 100% success rate even in the cross-platform attack. We expect the results from single-platform cross-device attacks to be similar to the results we get in single-device attacks.

9. Given the high success rate (100%) of attacks on full gate software implementations, we omit the results on half-gate implementations since it is always easier to attack half gates than full gates.

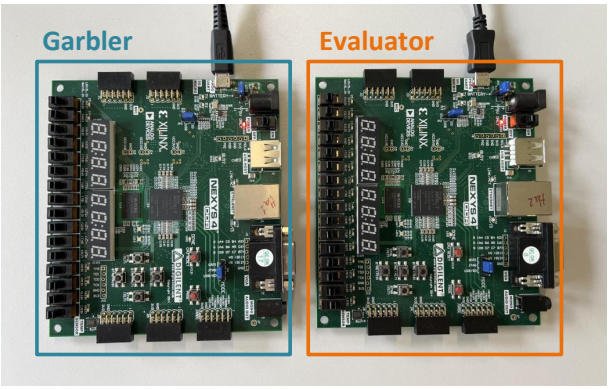

Figure 3. Two FPGA boards corresponding to the garbler and the evaluator.

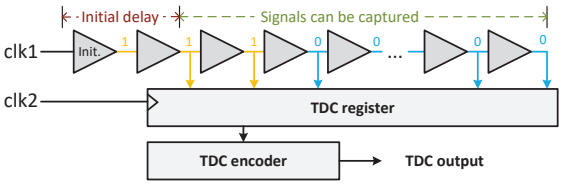

Figure 4. Functional schematic of a delay-line-based TDC.

4.3. Evaluation on FPGA Implementation

Remote power SCA allows the attacker to perform power analysis without physical access to the target devices. Hardware accelerators are typically used to enhance the performance of GC protocols. In the following experiment, we evaluate the proposed attack as a remote power analysis attacking an AES hardware accelerator on FPGAs. In particular, we mimic the cloud-FPGA scenario, where an attacker can implement malicious hardware logic (e.g., a power sensor) on a cloud-FPGA to attack another tenant on the cloud [37].

Experimental Setup. The attack is performed on two Digilent Nexys 4 DDR development boards based on the Xilinx Artix-7 FPGA platform. We deploy the logic of the garbler and evaluator on two separate FPGA boards, as shown in Fig. 3.

An open-source hardware AES core in Project Vault written by Google is implemented at 50 Mhz in the user logic on both FPGAs. The AES core computes each round in one clock cycle, with a total of ten cycles for one encryption.

We implement a delay line (DL)-based Time-to-Digital Converter (TDC) as a power sensor, running at 25 MHz in the attacker's logic on both FPGAs. As illustrated in Fig. 4, the DL-based TDC mainly consists of three parts: (1) **A delay-line** driven by clk1, in which CARRY4s are used as delay units. (2) **A TDC register** that captures the signal from the delay line at the rising edge of clk2. (3) **A TDC encoder** that is implemented through a priority encoder to calculate the index of the most significant activated bit of the delay-line state stored in the TDC register. The phase shift between the synchronized clk1 and clk2 enables the clk1 signal to propagate through the delay line, and the TDC captures the state register at each clock cycle. The signal propagation speed is influenced by the voltage on the board, so the fluctuation of the TDC encoder results represents the voltage variations of the targeted AES accelerator.

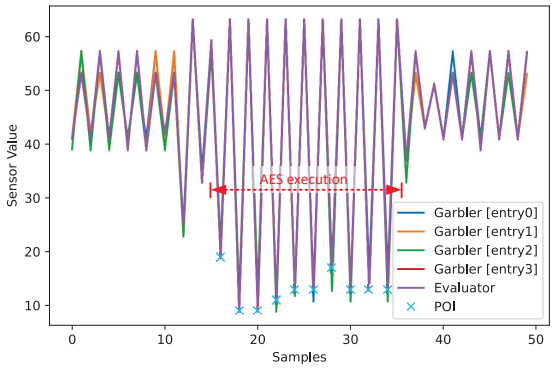

Figure 5. The TDC sensor values of the garbler's and the evaluator's fixed-key AES executions at one gate on an Artix-7 FPGA board.

TABLE 4. THE ATTACK RESULTS OF THE SINGLE-PLATFORM EVALUATION ON THE XILINX ARTIX-7 FPGA.

n_{POI}	$SR^1(\%)$	$SR^2(\%)$	avg_sig	$SR_{half\_gate}(\%)$
10	79.58	94.49	0.263	90.83

Attack Details. We run the experiment and collect the sensor data traces from the FPGAs. Five traces corresponding to the garbler and evaluator are overlaid in Fig. 5. The samples from the ten rounds of AES are within the red interval. Since the frequency of the TDC is half that of the AES core, we can only capture 20 sample points per AES execution. The dynamic range of the TDC we implemented is around 50, which is better than the results reported in [50]. However, the significantly lower resolution and shorter power traces, compared to those captured by a dedicated ADC in previous experiments, still pose a major challenge for the attack.

Similar to the software implementation, the sensor values across the traces are very close, making it hard to directly distinguish which trace of the garbler matches that of the evaluator by SPA. The possible choice for POIs is also limited, so we select POIs manually. During the execution of AES, the decrease in sensor values is due to the slight drop in chip voltage caused by register flipping each time the values are updated. Therefore, we select the 10 points where the voltage drops as POIs, as marked by the blue $\times$ in Fig. 5. Moreover, to mitigate the device variations between the two boards, we calculate an average trace on each board. In the attack, we subtract their respective average power trace from the measured traces.

Experimental Result. We generate random wire labels for 20000 gates and collect the power traces from the FPGAs. Due to the limited choice of POIs, we conduct experiments under only one parameter set. The attack result is given in Table 4.

When attacking the circuit with full gates, the SR^1

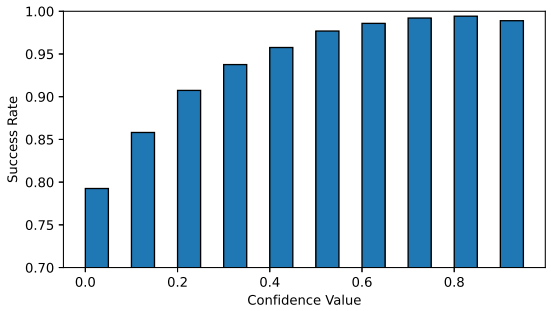

Figure 6. Success rates when confidence exceeds a certain value in the attack experiment on FPGAs.

achieves 79.58%, and the SR^2 reaches 94.49%. The success rate for attacking half gates SR_{half_gate} is 90.83%. Due to the shorter execution time of the hardware AES implementation, fewer feature points can be sampled and used for the attack, resulting in a lower success rate compared with attacks on software implementations.

When the success rate of our attack is not 100%, we can potentially cross-validate and correct results by utilizing the circuit and the confidence values of each guess. The confidence value can inform the attacker of the reliability of a given guess. Fig. 6 shows that a higher confidence value correlates with higher reliability of the results, which can provide additional hints in solving for the correct inputs of the circuit.

4.4. Evaluation on Cross-platform Implementation

In practice, the garbler and the evaluator may use processors with different microarchitectures, which makes it difficult for attackers to align the traces and use the same POIs on both sides. However, we can still select paired POIs between the garbler's and the evaluator's traces, according to AP 1. In the following experiment, we assess the effectiveness of our attack in a cross-platform setting.

Experimental Setup. We use the ChipWhisperer-Lite as the garbler's device and the ChipWhisperer-Nano as the evaluator's device, as shown in Fig. 7. The ChipWhisperer-Nano is equipped with a lighter processor, the STM32F042F6P6, based on ARM Cortex-M0 architecture. The STM32F3 on the ChipWhisperer-Lite is based on ARM Cortex-M4 architecture. Unlike Cortex-M0, which only supports the Thumb instruction set, Cortex-M4 supports the Thumb2 extended instruction set and offers better performance, so we use it as the garbler's device.

We still use the same Tiny-AES C program as before. However, the program is compiled separately using compilers corresponding to the different hardware platforms we use. The ADCs on both platforms sample once every clock cycle.

Attack Details. We run the fixed-key AES on two platforms. Fig. 8a and 8b show the traces corresponding to the

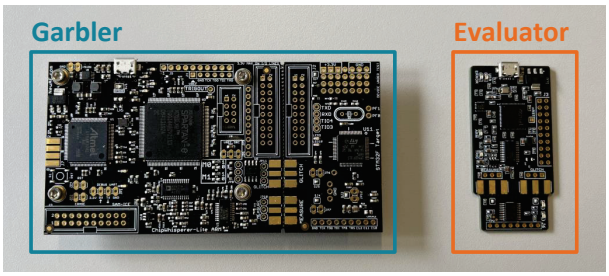

Figure 7. ChipWhisperer-Lite and ChipWhisperer-Nano corresponding to the garbler and the evaluator.

garbler and the evaluator, respectively. Due to the differences in the supported instruction set and microarchitecture, ChipWhisperer-Nano takes more clock cycles per AES execution. In this situation, we cannot use the same POIs on both sides, as we did in the single-platform evaluation described earlier. Therefore, we use steps 2 and 3 in AP 1 to determine the POIs. We collect 1000 traces using the same plaintext on each device for the profiling stage. Note that the profiling traces are independent of the traces for the real attacks. First, we select the POIs on the evaluator's trace by variance. For instance, when $l_s = 500$ and $k_{POI} = 1$, we get 23 POIs. Then, according to step 3, we select a corresponding POI on the garbler's trace for each evaluator's POI and calculate the correlation values associated with each pair of POIs. We filter the POI pairs with $t_{POI} = 0.6$, excluding some points with low correlation. The number of the remaining POIs is 20. The blue crosses in Fig. 8 indicate the positions of the selected POIs on both platforms. After the POIs are selected, the rest of the attack steps are carried out as usual.

Experimental Result. We generate 5000 groups of random wire labels and capture the power traces from the two platforms. We perform the attacks with various parameters. The results are given in Table 5. Four sets of parameters achieve a 100% success rate, indicating that after identifying the corresponding POIs, a successful attack can still be performed across different processor architectures. In the evaluation with $l_s = 500$ and $k_{POI} = 10$, using 1000 traces, the offline profiling takes 215.33s, and the online attack time is 0.032s per gate. Because of the differences in microarchitecture, instructions, and sensors between the two platforms, the average significance of the cross-platform attack is less than that of the single-platform attack with the same parameters.

To evaluate the robustness of our attack in practice, we also performed attacks on the data with additional artificial noise. In the cross-platform evaluation with $l_s = 500$ and $k_{POI} = 10$, we treated the original traces as clean signals and added Gaussian noise with SNRs ranging from 5 to 25. As shown in Fig. 9, the attack's success rate increases as the noise decreases. Note that our original experiments are based on real device measurements, which already contain noise from sources such as measurement errors and cross-device variations.

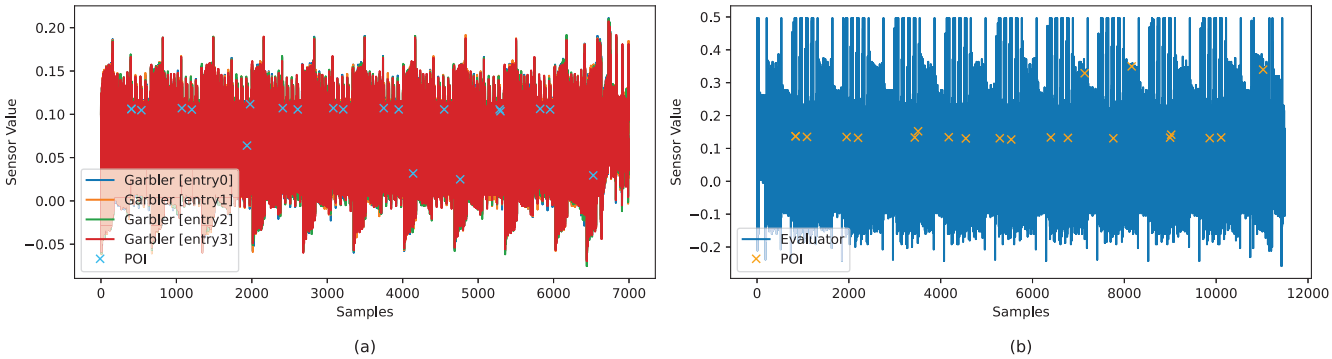

Figure 8. The ADC sensor values on both parties' devices: (a) ChipWhisperer-Lite corresponding to the garbler, (b) ChipWhisperer-Nano corresponding to the evaluator.

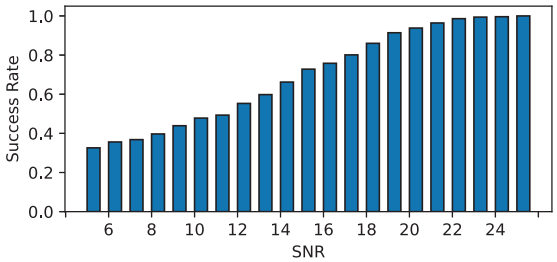

Figure 9. Success rates with $l_s = 500$ and $k_{POI} = 10$ when adding Gaussian noise with SNRs ranging from 5 to 25.

TABLE 5. THE ATTACK RESULTS OF THE CROSS-PLATFORM EVALUATION ON CHIPWHISPERER-LITE AND CHIPWHISPERER-NANO.

l_s	k_{POI}	n_{POI}	$SR(\%)$	avg_sig
1	1	1485	100	0.533
50	1	211	100	0.534
50	5	841	100	0.553
500	1	20	99.28	0.510
500	10	223	100	0.591
1000	1	10	94.88	0.448

5. Possible Countermeasures

The root cause of our attack is that the secret-dependent order of AES execution can be decoded by side-channel information. Based on this, we discuss possible countermeasures against our attack.

Random permutations in the garbling procedure. Within the garbling procedure of every gate, one can execute AES operations in a random order. Actually, we can reuse some of the randomness that already exists in the garbling procedure to randomize the execution order of the AES. For example, the `yaoGenerateGate` function in the Obliv-C framework encrypts four entries in the order of the P&P select bits instead of the actual value of each label. Since the select bit is random and independent of the semantic values, the proposed attack will not work on this particular implementation. This countermeasure incurs almost no overhead compared with the vulnerable implementations.

SCA resilient primitive implementations. One can also implement SCA resilient AES, such as [58], [59], to make the power traces less data-dependent. The randomness introduced by masking techniques can possibly prevent attackers from matching the power traces of the AES executions on the same plaintext performed by the garbler and the evaluator. However, this countermeasure may require additional hardware overhead or execution time. Alternatively, using SCA-resilient cryptographic primitives offers a more efficient solution with less overhead compared to second-order hardware masking [60].

6. Related Work

Existing SCA on GC. Since SCA on GC is still an emerging area, to the best of our knowledge, there are only two prior works on this topic. Levi et al. first proposed SCA on GC in [35]. Their attack functions similarly to a traditional horizontal power SCA, requiring knowledge of the processed data (garbler's labels) and a power trace. They can recover the global offset R in their attack. However, the method is specific to certain encryption schemes and does not work on the most popular encryption scheme (Eq. 1) adopted by all major GC frameworks. In contrast, our attack is effective on the most widely used encryption scheme and one of the most popular optimizations, half-gate.

In [36], Hashemi et al. introduced a machine learning-assisted timing side-channel attack (SCA) that exploits imbalanced if-branches in the garbler's wire label generation process, enabling the recovery of the garbler's input. This timing leakage was identified in TinyGarble, JustGarble, and Obliv-C, but it is not as pervasive as the order of execution issues found across all GC frameworks.

Single-Trace attacks. Classic SCAs like DPA and CPA require many traces from the target. In some cases, attackers can only capture a single trace, so various *single-trace attacks* were proposed [61]–[64].

Blind attacks. When the plaintexts or ciphertexts of a cryptographic algorithm are unknown to the adversary, one can only launch blind SCA. Linge et al. used joint distributions for blind SCA [65]. Clavier and Reynaud improved

the exploitation of joint distributions with the maximum likelihood approach [66].

Horizontal attacks. Horizontal attacks have become an important and popular methodology in power side-channel attacks (SCAs) in recent years [67]–[69]. These attacks select multiple intermediate values as points of interest (POIs) within a single trace and analyze them jointly to reduce the number of traces required [70]–[72]. The selection of POIs is crucial in horizontal attacks, as it significantly impacts the effectiveness and efficiency of the analysis. Among the existing POI selection schemes in [70], [73]–[78], we adopt the CPA method [79] for POI selection in our attack.

Remote side-channel attack. Unlike the classical SCAs that require physical access to the target devices, various remote attacks are proposed to obtain power side-channel leakages from a distance [50], [53]–[56], [80], [81]. For example, in a multi-tenant cloud FPGA scenario, the attacker can remotely obtain the power consumption data by implementing on-chip voltage sensors [50].

7. Conclusion and Future Work

GC is a practical MPC protocol due to its ability to efficiently evaluate Boolean circuits without revealing the private inputs of either party. Many frameworks have been developed to make GC more practical and easier to use. This paper proposes a single-trace blind power SCA against all existing GC frameworks, exploiting the secret-dependent order of operation issue we discovered. We evaluate our attack in multiple scenarios, including standard AES software, interleaved AES, hardware, and cross-platform implementations. According to the experiment results, the proposed attack can reveal the private input of the gates with a 100% success rate in some cases. Broadly speaking, this work challenges the validity of the standard non-collusion assumption in multi-party computation, pointing out the necessity of extending the assumption to the physical layer rather than limiting it to just the protocol participants.

For future research, one could enhance our attack by utilizing side-channel simulators [82], [83] to select points of interest (POIs) instead of profiling with actual hardware. Additionally, one can adjust the power model to attack a GC protocol running between a software implementation and a hardware implementation. Furthermore, if GC is implemented on a modern processor, such as within a trusted execution environment (TEE), additional measurement techniques, like single-step attacks [84], may be required to implement our attack.

References

[1] D. Chaum, C. Crépeau, and I. Damgård, "Multiparty unconditionally secure protocols (abstract) (informal contribution)," in *CRYPTO'87*, ser. LNCS, C. Pomerance, Ed., vol. 293. Springer, Heidelberg, Aug. 1988, p. 462.

[2] R. Cramer, I. Damgård, and U. M. Maurer, "General secure multi-party computation from any linear secret-sharing scheme," in *EUROCRYPT 2000*, ser. LNCS, B. Preneel, Ed., vol. 1807. Springer, Heidelberg, May 2000, pp. 316–334.

[3] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *41st ACM STOC*, M. Mitzenmacher, Ed. ACM Press, May / Jun. 2009, pp. 169–178.

[4] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *EUROCRYPT 2010*, ser. LNCS, H. Gilbert, Ed., vol. 6110. Springer, Heidelberg, May / Jun. 2010, pp. 24–43.

[5] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, "Fairplay - secure two-party computation system," in *USENIX Security 2004*, M. Blaze, Ed. USENIX Association, Aug. 2004, pp. 287–302.

[6] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway, "Efficient garbling from a fixed-key blockcipher," in *2013 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2013, pp. 478–492.

[7] X. Wang, A. J. Malozemoff, and J. Katz, "EMP-toolkit: Efficient MultiParty computation toolkit," https://github.com/emp-toolkit, 2016.

[8] F. K. Dankar, R. Brien, C. Adams, and S. Matwin, "Secure multi-party linear regression." in *EDBT/ICDT Workshops*, 2014, pp. 406–414.

[9] S. Carpov, T. H. Nguyen, R. Sirdey, G. Constantino, and F. Martinelli, "Practical privacy-preserving medical diagnosis using homomorphic encryption," in *2016 ieee 9th international conference on cloud computing (cloud)*. IEEE, 2016, pp. 593–599.

[10] D. Froelicher, J. R. Troncoso-Pastoriza, J. L. Raisaro, M. A. Cuendet, J. S. Sousa, H. Cho, B. Berger, J. Fellay, and J.-P. Hubaux, "Truly privacy-preserving federated analytics for precision medicine with multiparty homomorphic encryption," *Nature communications*, vol. 12, no. 1, p. 5910, 2021.

[11] F. Kerschbaum, "Outsourced private set intersection using homomorphic encryption," in *ASIACCS 12*, H. Y. Youm and Y. Won, Eds. ACM Press, May 2012, pp. 85–86.

[12] G. Xu, H. Li, Y. Zhang, S. Xu, J. Ning, and R. H. Deng, "Privacy-preserving federated deep learning with irregular users," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 2, pp. 1364–1381, 2020.

[13] S. Sav, A. Pyrgelis, J. R. Troncoso-Pastoriza, D. Froelicher, J.-P. Bossuat, J. S. Sousa, and J.-P. Hubaux, "POSEIDON: Privacy-preserving federated neural network learning," in *NDSS 2021*. The Internet Society, Feb. 2021.

[14] A. C.-C. Yao, "How to generate and exchange secrets (extended abstract)," in *27th FOCS*. IEEE Computer Society Press, Oct. 1986, pp. 162–167.

[15] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game or A completeness theorem for protocols with honest majority," in *19th ACM STOC*, A. Aho, Ed. ACM Press, May 1987, pp. 218–229.

[16] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2017, pp. 19–38.

[17] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *2013 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2013, pp. 334–348.

[18] A. Nautsch, A. Jiménez, A. Treiber, J. Kolberg, C. Jasserand, E. Kindt, H. Delgado, M. Todisco, M. A. Hmani, A. Mtibaa *et al.*, "Preserving privacy in speaker and speech characterisation," *Computer Speech & Language*, vol. 58, pp. 441–480, 2019.

[19] B. D. Rouhani, M. S. Riazi, and F. Koushanfar, "Deepsecure: Scalable provably-secure deep learning," in *Proceedings of the 55th annual design automation conference*, 2018, pp. 1–6.

[20] D. Beaver, S. Micali, and P. Rogaway, "The round complexity of secure protocols (extended abstract)," in *22nd ACM STOC*. ACM Press, May 1990, pp. 503–513.

[21] M. Naor, B. Pinkas, and R. Sumner, "Privacy preserving auctions and mechanism design," in *Proceedings of the 1st ACM Conference on Electronic Commerce*, 1999, pp. 129–139.

[22] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free xor gates and applications," in *Automata, Languages and Programming: 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II 35*. Springer, 2008, pp. 486–498.

[23] S. Zahur, M. Rosulek, and D. Evans, "Two halves make a whole - reducing data transfer in garbled circuits using half gates," in *EUROCRYPT 2015, Part II*, ser. LNCS, E. Oswald and M. Fischlin, Eds., vol. 9057. Springer, Heidelberg, Apr. 2015, pp. 220–250.

[24] M. Rosulek and L. Roy, "Three halves make a whole? Beating the half-gates lower bound for garbled circuits," in *CRYPTO 2021, Part I*, ser. LNCS, T. Malkin and C. Peikert, Eds., vol. 12825. Virtual Event: Springer, Heidelberg, Aug. 2021, pp. 94–124.

[25] E. M. Songhori, S. U. Hussain, A.-R. Sadeghi, T. Schneider, and F. Koushanfar, "TinyGarble: Highly compressed and scalable sequential garbled circuits," in *2015 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2015, pp. 411–428.

[26] S. Hussain, B. Li, F. Koushanfar, and R. Cammarota, "Tinygarble2: Smart, efficient, and scalable yao's garble circuit," in *Proceedings of the 2020 Workshop on Privacy-Preserving Machine Learning in Practice*, 2020, pp. 65–67.

[27] D. Demmler, T. Schneider, and M. Zohner, "ABY - A framework for efficient mixed-protocol secure two-party computation," in *NDSS 2015*. The Internet Society, Feb. 2015.

[28] A. Patra, T. Schneider, A. Suresh, and H. Yalame, "ABY2.0: Improved mixed-protocol secure two-party computation," in *USENIX Security 2021*, M. Bailey and R. Greenstadt, Eds. USENIX Association, Aug. 2021, pp. 2165–2182.

[29] P. Mohassel and P. Rindal, "Aby3: A mixed protocol framework for machine learning," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 35–52.

[30] S. Zahur and D. Evans, "Obliv-C: A language for extensible data-oblivious computation," Cryptology ePrint Archive, Report 2015/1153, 2015, https://eprint.iacr.org/2015/1153.

[31] L. Braun, D. Demmler, T. Schneider, and O. Tkachenko, "Motion–a framework for mixed-protocol multi-party computation," *ACM Transactions on Privacy and Security*, vol. 25, no. 2, pp. 1–35, 2022.

[32] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Power analysis attacks of modular exponentiation in smartcards," in *CHES'99*, ser. LNCS, Çetin Kaya. Koç and C. Paar, Eds., vol. 1717. Springer, Heidelberg, Aug. 1999, pp. 144–157.

[33] D. Brumley and D. Boneh, "Remote timing attacks are practical," in *USENIX Security 2003*. USENIX Association, Aug. 2003.

[34] Q. Jean-Jacques, "A new tool for non-intrusive analysis of smart cards based on electromagnetic emissions. the sema and dema methods," *Eurocrypt2000 Rump Session, May*, 2000.

[35] I. Levi and C. Hazay, "Garbled circuits from an SCA perspective free XOR can be quite expensive. ." *IACR TCHES*, vol. 2023, no. 2, pp. 54–79, 2023.

[36] M. Hashemi, D. Forte, and F. Ganji, "Time is money, friend! timing side-channel attack against garbled circuit constructions," in *International Conference on Applied Cryptography and Network Security*. Springer, 2024, pp. 325–354.

[37] C. Jin, V. Gohil, R. Karri, and J. Rajendran, "Security of cloud fpgas: A survey," *arXiv preprint arXiv:2005.04867*, 2020.

[38] M. Bellare, V. T. Hoang, and P. Rogaway, "Foundations of garbled circuits," in *ACM CCS 2012*, T. Yu, G. Danezis, and V. D. Gligor, Eds. ACM Press, Oct. 2012, pp. 784–796.

[39] Y. Lindell and B. Pinkas, "A proof of security of Yao's protocol for two-party computation," *Journal of Cryptology*, vol. 22, no. 2, pp. 161–188, Apr. 2009.

[40] Y. Lindell, B. Pinkas, and N. P. Smart, "Implementing two-party computation efficiently with security against malicious adversaries," in *SCN 08*, ser. LNCS, R. Ostrovsky, R. D. Prisco, and I. Visconti, Eds., vol. 5229. Springer, Heidelberg, Sep. 2008, pp. 2–20.

[41] B. Kreuter, a. shelat, and C.-H. Shen, "Billion-gate secure computation with malicious adversaries," in *USENIX Security 2012*, T. Kohno, Ed. USENIX Association, Aug. 2012, pp. 285–300.

[42] V. Kolesnikov, P. Mohassel, and M. Rosulek, "FleXOR: Flexible garbling for XOR gates that beats free-XOR," in *CRYPTO 2014, Part II*, ser. LNCS, J. A. Garay and R. Gennaro, Eds., vol. 8617. Springer, Heidelberg, Aug. 2014, pp. 440–457.

[43] C. Guo, J. Katz, X. Wang, C. Weng, and Y. Yu, "Better concrete security for half-gates garbling (in the multi-instance setting)," in *CRYPTO 2020, Part II*, ser. LNCS, D. Micciancio and T. Ristenpart, Eds., vol. 12171. Springer, Heidelberg, Aug. 2020, pp. 793–822.

[44] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *CRYPTO'99*, ser. LNCS, M. J. Wiener, Ed., vol. 1666. Springer, Heidelberg, Aug. 1999, pp. 388–397.

[45] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems-CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings 6*. Springer, 2004, pp. 16–29.

[46] S. B. Ors, F. Gurkaynak, E. Oswald, and B. Preneel, "Power-analysis attack on an asic aes implementation," in *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.*, vol. 2. IEEE, 2004, pp. 546–552.

[47] S. Mangard, N. Pramstaller, and E. Oswald, "Successfully attacking masked AES hardware implementations," in *CHES 2005*, ser. LNCS, J. R. Rao and B. Sunar, Eds., vol. 3659. Springer, Heidelberg, Aug. / Sep. 2005, pp. 157–171.

[48] N. Samwel, L. Batina, G. Bertoni, J. Daemen, and R. Susella, "Breaking ed25519 in wolfssl," in *Topics in Cryptology–CT-RSA 2018: The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings*. Springer, 2018, pp. 1–20.

[49] L. Oostrum, J. Romein, B. van Werkhoven, Q. Twisk, G. Schoonderbeek, and S. van der Vlugt, "PowerSensor3." [Online]. Available: https://github.com/nlesc-recruit/PowerSensor3

[50] F. Schellenberg, D. R. Gnad, A. Moradi, and M. B. Tahoori, "An inside job: Remote power analysis attacks on fpgas," *IEEE Design & Test*, vol. 38, no. 3, pp. 58–66, 2021.

[51] J. Choncholas, K. Bhardwaj, and A. Gavrilovska, "Tgh: A tee/gc hybrid enabling confidential faas platforms," *arXiv preprint arXiv:2309.07764*, 2023.

[52] C. Jin, C. Yin, M. van Dijk, S. Duan, F. Massacci, M. K. Reiter, and H. Zhang, "PG: Byzantine fault-tolerant and privacy-preserving sensor fusion with guaranteed output delivery," in *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2024.

[53] M. Lipp, A. Kogler, D. Oswald, M. Schwarz, C. Easdon, C. Canella, and D. Gruss, "Platypus: Software-based power wide-channel attacks on x86," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 355–371.

[54] A. Kogler, J. Juffinger, L. Giner, L. Gerlach, M. Schwarzl, M. Schwarz, D. Gruss, and S. Mangard, "{Collide+ Power}: Leaking inaccessible data with software-based power side channels," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 7285–7302.

[55] C. Liu, A. Chakraborty, N. Chawla, and N. Roggel, "Frequency throttling side-channel attack," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 1977–1991.

[56] D. R. Dipta and B. Gulmezoglu, "Df-sca: Dynamic frequency side channel attacks are practical," in *Proceedings of the 38th Annual Computer Security Applications Conference*, 2022, pp. 841–853.

[57] F.-X. Standaert, T. Malkin, and M. Yung, "A unified framework for the analysis of side-channel key recovery attacks," in *EUROCRYPT 2009*, ser. LNCS, A. Joux, Ed., vol. 5479. Springer, Heidelberg, Apr. 2009, pp. 443–461.

[58] P. Sasdrich, B. Bilgin, M. Hutter, and M. E. Marson, "Low-latency hardware masking with application to AES," *IACR TCHES*, vol. 2020, no. 2, pp. 300–326, 2020, https://tches.iacr.org/index.php/TCHES/article/view/8553.

[59] J.-S. Coron, A. Greuet, and R. Zeitoun, "Side-channel masking with pseudo-random generator," in *EUROCRYPT 2020, Part III*, ser. LNCS, A. Canteaut and Y. Ishai, Eds., vol. 12107. Springer, Heidelberg, May 2020, pp. 342–375.

[60] R. Li, Y. Sun, C. Guo, F.-X. Standaert, W. Wang, and X. Wang, "Leakage-resilience of circuit garbling," in *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2024.

[61] C. Clavier, B. Feix, G. Gagnerot, C. Giraud, M. Roussellet, and V. Verneuil, "ROSETTA for single trace analysis," in *INDOCRYPT 2012*, ser. LNCS, S. D. Galbraith and M. Nandi, Eds., vol. 7668. Springer, Heidelberg, Dec. 2012, pp. 140–155.

[62] A. Bauer, É. Jaulmes, E. Prouff, and J. Wild, "Horizontal collision correlation attack on elliptic curves," in *SAC 2013*, ser. LNCS, T. Lange, K. Lauter, and P. Lisonek, Eds., vol. 8282. Springer, Heidelberg, Aug. 2014, pp. 553–570.

[63] J. W. Bos, S. Friedberger, M. Martinoli, E. Oswald, and M. Stam, "Assessing the feasibility of single trace power analysis of Frodo," in *SAC 2018*, ser. LNCS, C. Cid and M. J. Jacobson Jr:, Eds., vol. 11349. Springer, Heidelberg, Aug. 2019, pp. 216–234.

[64] D. Amiet, A. Curiger, L. Leuenberger, and P. Zbinden, "Defeating NewHope with a single trace," in *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, J. Ding and J.-P. Tillich, Eds. Springer, Heidelberg, 2020, pp. 189–205.

[65] Y. Linge, C. Dumas, and S. Lambert-Lacroix, "Using the joint distributions of a cryptographic function in side channel analysis," in *COSADE 2014*, ser. LNCS, E. Prouff, Ed., vol. 8622. Springer, Heidelberg, Apr. 2014, pp. 199–213.

[66] C. Clavier and L. Reynaud, "Improved blind side-channel analysis by exploitation of joint distributions of leakages," in *CHES 2017*, ser. LNCS, W. Fischer and N. Homma, Eds., vol. 10529. Springer, Heidelberg, Sep. 2017, pp. 24–44.

[67] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil, "Horizontal correlation analysis on exponentiation," in *Information and Communications Security: 12th International Conference, ICICS 2010, Barcelona, Spain, December 15-17, 2010. Proceedings 12*. Springer, 2010, pp. 46–61.

[68] R. Poussier, Y. Zhou, and F.-X. Standaert, "A systematic approach to the side-channel analysis of ECC implementations with worst-case horizontal attacks," in *CHES 2017*, ser. LNCS, W. Fischer and N. Homma, Eds., vol. 10529. Springer, Heidelberg, Sep. 2017, pp. 534–554.

[69] I. Diop, Y. Linge, T. Ordas, P.-Y. Liardet, and P. Maurine, "From theory to practice: horizontal attacks on protected implementations of modular exponentiations," *Journal of Cryptographic Engineering*, vol. 9, no. 1, pp. 37–52, Apr. 2019.

[70] C. Archambeau, E. Peeters, F.-X. Standaert, and J.-J. Quisquater, "Template attacks in principal subspaces," in *CHES 2006*, ser. LNCS, L. Goubin and M. Matsui, Eds., vol. 4249. Springer, Heidelberg, Oct. 2006, pp. 1–14.

[71] M. Bär, H. Drexler, and J. Pulkus, "Improved template attacks," *COSADE2010*, vol. 42, 2010.

[72] O. Choudary and M. G. Kuhn, "Efficient template attacks," in *Smart Card Research and Advanced Applications: 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers 12*. Springer, 2014, pp. 253–270.

[73] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *CHES 2002*, ser. LNCS, B. S. Kaliski Jr., Çetin Kaya. Koç, and C. Paar, Eds., vol. 2523. Springer, Heidelberg, Aug. 2003, pp. 13–28.

[74] B. Gierlichs, K. Lemke-Rust, and C. Paar, "Templates vs. stochastic methods," in *CHES 2006*, ser. LNCS, L. Goubin and M. Matsui, Eds., vol. 4249. Springer, Heidelberg, Oct. 2006, pp. 15–29.

[75] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer Science & Business Media, 2008, vol. 31.

[76] L. Mather, E. Oswald, J. Bandenburg, and M. Wójcik, "Does my device leak information? An a priori statistical power analysis of leakage detection tests," in *ASIACRYPT 2013, Part I*, ser. LNCS, K. Sako and P. Sarkar, Eds., vol. 8269. Springer, Heidelberg, Dec. 2013, pp. 486–505.

[77] N. Veyrat-Charvillon and F.-X. Standaert, "Mutual information analysis: How, when and why?" in *CHES 2009*, ser. LNCS, C. Clavier and K. Gaj, Eds., vol. 5747. Springer, Heidelberg, Sep. 2009, pp. 429–443.

[78] C. Whitnall, E. Oswald, and L. Mather, "An exploration of the kolmogorov-smirnov test as a competitor to mutual information analysis," in *International Conference on Smart Card Research and Advanced Applications*. Springer, 2011, pp. 234–251.

[79] G. Fan, Y. Zhou, H. Zhang, and D. Feng, "How to choose interesting points for template attacks more effectively?" in *Trusted Systems: 6th International Conference, INTRUST 2014, Beijing, China, December 16-17, 2014, Revised Selected Papers 6*. Springer, 2015, pp. 168–183.

[80] Y. Wang, R. Paccagnella, E. T. He, H. Shacham, C. W. Fletcher, and D. Kohlbrenner, "Hertzbleed: Turning power side-channel attacks into remote timing attacks on x86," in *USENIX Security 2022*, K. R. B. Butler and K. Thomas, Eds. USENIX Association, Aug. 2022, pp. 679–697.

[81] D. Spielmann, O. Glamocanin, and M. Stojilovic, "RDS: FPGA routing delay sensors for effective remote power analysis attacks," *IACR TCHES*, vol. 2023, no. 2, pp. 543–567, 2023.

[82] N. Veshchikov and S. Guilley, "Use of simulators for side-channel analysis," in *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2017, pp. 104–112.

[83] N. Sehatbakhsh, B. B. Yilmaz, A. Zajic, and M. Prvulovic, "Emsim: A microarchitecture-level simulation tool for modeling electromagnetic side-channel signals," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 71–85.

[84] M. Lipp, A. Kogler, D. F. Oswald, M. Schwarz, C. Easdon, C. Canella, and D. Gruss, "PLATYPUS: Software-based power side-channel attacks on x86," in *2021 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2021, pp. 355–371.

Appendix

In JustGarble, TinyGarble, and EMP-toolkit frameworks, the fixed-key AES is implemented by specific instructions for higher performance. The following code snippet of TinyGarble shows the AES calls in the garbling scheme of a gate:

```
for (i = 0; i < nblks; ++i) //first round
   blks[i] = _mm_xor_si128(blks[i], sched[0]);
for (j = 1; j < rnds; ++j) //intermediate rounds
   for (i = 0; i < nblks; ++i)
```

TABLE 6. THE ATTACK RESULTS OF THE INTERLEAVED AES IMPLEMENTATION ON CHIPWHISPERER-LITE.

l_s	k_{POI}	n_{POI}	$SR(\%)$	avg_sig
50	1	13	84.95	0.308
50	5	87	100	0.478

```
5       blks[i] = _mm_aesenc_si128(blks[i], sched[j
            ]);
6   for (i = 0; i < nblks; ++i) //last round
7      blks[i] = _mm_aesenclast_si128(blks[i], sched[j
           ]);
```

where the array `blks` stores the plaintexts for the AES calls. The variable `nblks` represents the number of the plaintexts, which equals 2 when using half-gate optimization and 4 in other cases. Though the order of encryption is also secret-dependent, the AES execution is interleaved among a group of executions, necessitating more segmentation and classification of the traces by attackers. However, our proposed attack method for identifying points of interest (POIs) can effectively locate four distinct groups of POIs within the intermingled traces. We evaluate this attack under conditions where AES executions are interleaved.

Experimental Setup. We modified the TinyAES C code to interleave each round of the four AES executions to create an interleaved implementation of AES. The modified program is loaded on ChipWhisperer-Lite to capture the power consumption as in the previous experiments. In each gate, the garbler executes one interleaved AES on four entries, obtaining only a longer trace, while the evaluator still performs a single standard AES operation.

Attack Details. To attack the interleaved AES trace, we separate the feature points corresponding to different entries in the trace by identifying the POIs. Each POI in the evaluator's trace corresponds to four POIs in the garbler's interleaved trace. Therefore, during the profiling phase, we generated four sets of plaintexts and collected the corresponding four sets of standard AES traces and one interleaved AES trace. Using step 2 in AP 1, we select the POIs in the standard AES based on the variance. With the four sets of traces, we locate four corresponding sets of POIs in the longer trace through step 3 and filter them with $t_{POI} = 0.6$. Fig. 10 shows the corresponding relationships of POIs in the evaluator's and the garbler's traces. Since the garbler's AES is executed in an interleaved manner, one POI in the evaluator's trace corresponds to four POIs in the garbler's trace. Finally, we condense the garbler's trace into four feature vectors using the selected four sets of POIs and perform the attack.

Experimental Result. We generate data for 2000 gates, and the power traces are collected from the ChipWhisperer-Lite. We successfully achieved a 100% success rate with parameters $l_s = 50$ and $k_{POI} = 1$. Table 6 gives the detailed attack results.

This result indicates that the way of selecting POIs in AP 1 remains effective in the context of interleaved AES. Due to the addition of the process to separate the interleaved sections, the number of POIs retained after filtering is reduced in the attack, and the average significance of the attack is lower compared to that of a normal AES scenario.

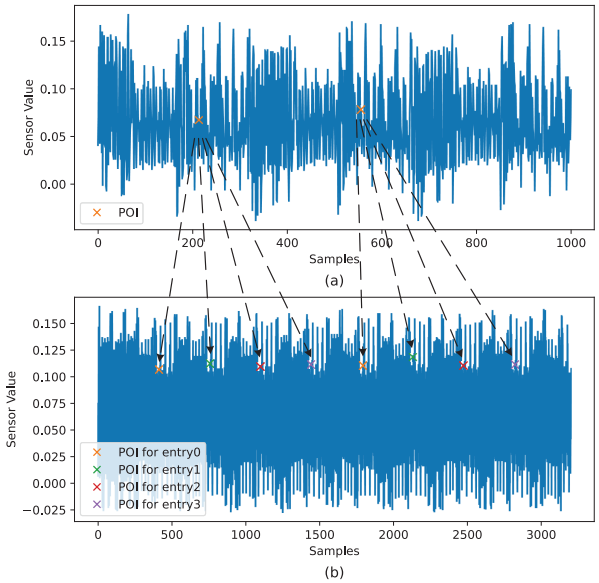

Figure 10. Segments of the power consumption of the AES operation on the evaluator and the garbler, along with the corresponding relationships between two POIs on the traces: (a) a trace of the evaluator's standard AES execution (b) a trace of the garbler's interleaved AES execution.