IACR Communications in Cryptology
ISSN 3006-5496, Vol. 1, No. 3, 25 pages.

https://doi.org/10.62056/angy11zn4

# Side-Channel Linearization Attack on Unrolled Trivium Hardware

Soichiro Kobayashi[1], Rei Ueno[2], Yosuke Todo[3] and Naofumi Homma[1]

[1] Tohoku University, Sendai, Japan
[2] Kyoto University, Kyoto, Japan
[3] NTT Social Informatics Laboratories, Tokyo, Japan

**Abstract.** This paper presents a new side-channel attack (SCA) on unrolled implementations of stream ciphers, with a particular focus on Trivium. Most conventional SCAs predominantly concentrate on leakage of some first rounds prior to the sufficient diffusion of the secret key and initial vector (IV). However, recently, unrolled hardware implementation has become common and practical, which achieves higher throughput and energy efficiency compared to a round-based hardware. The applicability of conventional SCAs to such unrolled hardware is unclear because the leakage of the first rounds from unrolled hardware is hardly observed. In this paper, focusing on Trivium, we propose a novel SCA on unrolled stream cipher hardware, which can exploit leakage of rounds latter than 80, while existing SCAs exploited intermediate values earlier than 80 rounds. We first analyze the algebraic equations representing the intermediate values of these rounds and present the *recursive restricted linear decomposition (RRLD)* strategy. This approach uses correlation power analysis (CPA) to estimate the intermediate values of latter rounds. Furthermore, we present a chosen-IV strategy for a successful key recovery through *linearization*. We experimentally demonstrate that the proposed SCA achieves the key recovery of a 288-round unrolled Trivium hardware implementation using 360,000 traces. Finally, we evaluate the performance of unrolled Trivium hardware implementations to clarify the trade-off between performance and SCA (in)security. The proposed SCA requires 34.5 M traces for a key recovery of 384-round unrolled Trivium implementation and is not applicable to 576-round unrolled hardware.

**Keywords:** Stream cipher · Trivium · Unrolled implementation · Side-channel attack (SCA) · Correlation power analysis (CPA)

## 1 Introduction

### 1.1 Background

**Trivium.** Trivium is a major hardware-oriented stream cipher [DCP08, Can06] included in the ISO/IEC 29192-3 standard [iso] and the portfolio of eSTREAM project [eST]. Given an 80-bit secret key and an 80-bit initial vector (IV), Trivium performs *initialization* phase, following by *key stream generation* phase. The initialization randomizes a 288-bit internal state determined by the secret key and IV over 1152 rounds. Then, the key stream is generated with a throughput of one bit per round.

E-mail: soichiro.kobayashi.q7@dc.tohoku.ac.jp (Soichiro Kobayashi), ueno.rei.2e@kyoto-u.ac.jp (Rei Ueno), yosuke.todo@ntt.com (Yosuke Todo), naofumi.homma.c8@tohoku.ac.jp (Naofumi Homma)

This work is licensed under a "CC BY 4.0" license.

Received: 2024-07-04 Accepted: 2024-09-02

**Hardware implementation of Trivium.** Initially, Trivium hardware was implemented with a *round-based* architecture, which serially performs one round per clock cycle using a round datapath similar to a common hardware implementation of block ciphers. Then, *round unrolling* has been a common technique to implement Trivium with a practical latency and throughput as well as an acceptable area overhead [CMM+23]. Recently, some studies further showed its usefulness in terms of energy efficiency [BMA+18, CBT+21]. In this paper, we refer to the number of unrolled rounds as *unrolling degree*, and denote it by $r$. An $r$-round unrolled implementation, which executes $r$ rounds per clock cycle using an $r$-round datapath, can reduce the initialization latency (*i.e.*, clock cycles) and can improve the key stream generation throughput by a factor of $r$ compared to the round-based implementation. In [BMA+18, CBT+21], it was experimentally showed that Trivium unrolled hardware implementation achieves a higher energy efficiency up to $r = 288$ than the round-based hardware, and $r = 288$ is optimal in terms of energy efficiency. Thus, round unrolling is a common technique to implement practical stream cipher hardware regarding the trade-offs between throughput (latency), area, and energy.

**Side-channel attack on Trivium.** Some SCAs on Trivium have been studied and evaluated, including differential power analysis (DPA), correlation power analysis (CPA), and profiled SCA [FGKV06,SPK09,JHWW12,TSA15a,TSA15b,SJB21,KDB+22]. However, the existing SCAs have not considered the unrolled implementation, and their applicability is unclear, whereas there are some studies on the unrolled implementation of block ciphers [BGSD10, YMHA16, MS16, YMUM+21]. In fact, the above DPAs/CPAs have focused on the leakage from some first rounds of initialization for estimating intermediate bits, but such leakage would *not* be available in the case of unrolled implementation, because they are not stored in register. Furthermore, the state-of-the-art profiled SCA in [KDB+22], which aims to recover the internal state, focused on a round-based implementation and assumed that the leakage of consecutive rounds is available. Thus, there is little study on the evaluation of the side-channel (in)security of unrolled Trivium, while round unrolling is common and practical for implementing Trivium on hardware. In the field of block ciphers, the unrolled implementation was thought to yield a higher side-channel resistance compared to the round-based counterpart because it does not store intermediate values targeted by SCAs in registers [BGSD10]. However, it is unclear whether this thought applies to stream ciphers as well due to the differences in SCA strategies and unrolled hardware architectures. Therefore, it is highly desirable to investigate the SCA (in)security of unrolled stream cipher hardware.

*Remark* 1 (SCA on wire values). Some SCAs have used side-channel information from wire values and glitches [MPO05, YMHA16, YMUM+21]. Specifically, for Trivium, DPAs focused on a wire value of a two-input XOR output were presented in [TSA15a, TSA15b]. If it could be available, an SCA on earlier rounds would be feasible even on unrolled hardware. However, these were only evaluated by simulation; their practical feasibility was unclear. In addition, in [Moo20], Moos discussed static side-channel leakages of unrolled block cipher implementation and its practicality. *No reset attack* in the paper would be infeasible to standard stream cipher implementation even unrolled, because the internal state (and wires) are diffused/randomized and not observable to the attacker. By contrast, if attacker can reset the target module, it may be a threat as well as case of unrolled block ciphers.

## 1.2 Our contributions

**New SCA on unrolled stream cipher implementation.** We present a new chosen-IV SCA methodology on stream ciphers focusing on Trivium, named *side-channel linearization attack*. The existing SCAs estimated intermediate bits depending on a few bits of secret key by CPA [FGKV06, SPK09, JHWW12, TSA15a, TSA15b]. The proposed SCA is their extension/generalization and also uses CPA to estimate the intermediate bits of the $r$-th round internal state, as they would be stored in registers of $r$-round unrolled hardware. There are two major technical challenges addressed by the proposed SCA.

*Estimation of internal state bits of rounds latter than 80.* It is non-trivial how to estimate the intermediate bits at rounds latter than 80 by CPA due to the diffusion, as these bits are represented by high-degree non-linear polynomials with many bits of the secret key and IV. To recover the intermediate bits by CPA with a feasible guess, we analyze the polynomials and present a *recursive restricted linear decomposition (RRLD)* strategy, which linearly separates the polynomials to guess the intermediate bits by CPA with practical complexity. Using the RRLD strategy, we determine the chosen IVs to estimate the intermediate bits by CPA with the help of SAT/SMT solver.

*Feasible key recovery.* For the case of a large unrolling degree, it is non-trivial to reverse the secret key from the intermediate bits by CPA is too high to solve it with a practical complexity. To fully recover a secret key, we propose another chosen-IV strategy based on *linearization.* In restricting the IV in the RRLD strategy, we use two sets of chosen-IV patterns dedicated to the linearization. From the results of two CPAs using each chosen-IV set, we derive a linear equation of key bits. As a result, we can achieve the full key recovery by means of Gaussian elimination with a practical complexity.

**Validation.** We demonstrate the validity of the proposed SCA through experimental attacks on a 288-round unrolled hardware implementation of Trivium on an FPGA. We confirm that the proposed SCA can successfully obtain the linear equation of key bits by RRLD and linearization, and consequently reduce the Trivium key space from $2^{80}$ to $2^{29}$ with a total of 360,000 traces. Furthermore, we evaluate the relation between success rate, the number of guessed bits, and the number of available chosen-IV patterns (*i.e.*, the size of the chosen-IV set). As a result, we reveal that a CPA is not successful if the attacker cannot use a sufficient number of IV patterns. Such an insufficiency occurs in attacking more diffused rounds, namely, attacking unrolled hardware with larger unrolling degrees.

**Quantitative discussion of SCA (in)security of unrolled Trivium.** We evaluate the total number of traces for different unrolling degrees through our experiment. Specifically, evaluations for $r = 384$ and 576, which are divisors of 1152 grater than 288, are included. We confirmed that the total number of traces for full key recovery at $r = 384$ would be significantly greater than that for $r = 288$. Moreover, key recovery at $r = 576$ is deemed unfeasible by the proposed SCA, as we find too few chosen IVs for the proposed SCA at this degree. We finally evaluate the unrolled hardware of Trivium with different unrolling degrees to clarify the trade-off between performance and SCA security (whereas the previous studies did not report the performance in detail for $r > 288$).

The source code for our experiments is available at `https://github.com/ECSIS-lab/CiC_KUTH24`.

## 2 Preliminaries

### 2.1 Notations

Let $k = (k_1, k_2, \ldots, k_{80})$ and $v = (v_1, v_2, \ldots, v_{80})$ denote the 80-bit secret key and IV of Trivium, where $k_j$ and $v_j$ denote the $j$-th bit of the secret key and IV, respectively. Let $s^i = (s^i_1, s^i_2, \ldots, s^i_{288})$ denote the 288-bit internal state at the $i$-th round, where $s^i_j$ denotes its $j$-th bit (we may omit the superscript $i$ in some contexts if it is not needed). A calligraphic letter (*e.g.*, $\mathcal{X}$) denotes a set, and a corresponding lower letter (*i.e.*, $x$) is its element unless defined differently. In particular, $\mathcal{K}$ and $\mathcal{V}$ $(= \{0, 1\}^{80})$ denote the sets

**Algorithm 1** Trivium initialization

```
Input: Secret key $k = (k_1, k_2, \ldots, k_{80})$ and IV $v = (v_1, v_2, \ldots, v_{80})$;
Output: Initialized internal state $s = (s_1, s_2, \ldots, s_{288})$;
 1: $(s_1, s_2, \ldots, s_{93}) \leftarrow (k_1, k_2, \ldots, k_{80}, 0, \ldots, 0)$;    ▷ Internal state setup
 2: $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (v_1, v_2, \ldots, v_{80}, 0, \ldots, 0)$;
 3: $(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (0, 0 \ldots, 0, 1, 1, 1)$;
 4: for $i = 1$ to 1152 do                                          ▷ Round function
 5:     $t_1 \leftarrow s_{66} + s_{93} + s_{91}s_{92} + s_{171}$;
 6:     $t_2 \leftarrow s_{162} + s_{177} + s_{175}s_{176} + s_{264}$;
 7:     $t_3 \leftarrow s_{243} + s_{288} + s_{286}s_{287} + s_{69}$;
 8:     $(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$;
 9:     $(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$;
10:     $(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$;
11: end for
12: return $s$;
```

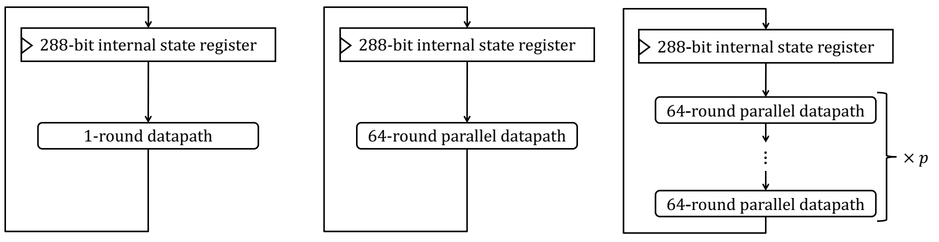
of all secret keys and IVs, respectively. Given a function $f : \mathcal{X} \to \mathcal{Y}$ and an element of its range $y \in \mathcal{Y}$, a subset $\{ x \mid f(x) = y \} \subseteq \mathcal{X}$ may be denoted by $\{f(x) = y\}$. Given a function $f : \mathcal{X} \to \mathcal{Y}$, its restriction to $\mathcal{A} \subseteq \mathcal{X}$ is denoted by $f|_{\mathcal{A}} : \mathcal{A} \to \mathcal{Y}$. Note that, for functions $f$, $g$, and $h$, if $g = f|_{\mathcal{A}}$, $h = g|_{\mathcal{B}}$ and $\mathcal{B} \subseteq \mathcal{A}$ hold, then $h = f|_{\mathcal{B}}$ holds. Moreover, for a function $f : \mathcal{X} \to \mathcal{Y}$ and $y \in \mathcal{Y}$, it always holds that $f|_{\{f(x)=y\}} = y$. In this paper, a restricted function domain corresponds to a chosen-IV set. Finally, we use $+$ as an XOR operator for cryptographic computation over $\mathbb{F}_2$.

## 2.2 Trivium specification

Algorithm 1 describes the Trivium initialization specification, with the key stream generation being similar, except for the output of the key stream bit as $o^i = s^i_{66} + s^i_{93} + s^i_{162} + s^i_{177} + s^i_{243} + s^i_{288}$. Trivium uses a 288-bit internal state, composed of 93-bit, 84-bit, and 111-bit non-linear feedback registers (NLFSRs), interconnected via a round function. During the initialization phase, as described at Lines 1–3 in Algorithm 1, the internal state (*i.e.*, the NLFSRs seed) is set up using an 80-bit secret key $k = (k_1, k_2, \ldots, k_{80})$, an 80-bit IV $v = (v_1, v_2, \ldots, v_{80})$, and constants. This initialization consists of 1152 rounds. In the round function, three intermediate bits denoted by $t_1$, $t_2$, and $t_3$ are generated from 15 state bits and stored in the tails of NLFSRs. These intermediate values $t_1$, $t_2$, and $t_3$ are crucial for existing and proposed SCAs, which are utilized in the key recovery by CPA. After the internal state is updated 1152 times using the round function, the initialized internal state is output. In the key stream generation phase, a one-bit key stream $o^i = s^i_{66} + s^i_{93} + s^i_{162} + s^i_{177} + s^i_{243} + s^i_{288}$ $(i > 1152)$ is generated per round, whereas the internal state is updated similarly to the initialization phase.

## 2.3 Unrolled hardware implementation of Trivium

Historically, hardware implementations of stream ciphers were based on a round-based architecture, serially executing the round function per clock cycle using one round datapath. However, recent studies [BMA+18, CBT+21] have been demonstrated that an unrolled implementation, executing multiple rounds per clock cycle using multiple round datapaths, achieves a higher energy efficiency. Figure 1 shows overviews of (a) the round-based, (b) 64-round unrolled, and (c) 64$p$-round unrolled data flows, where $p$ is an integer such that $p = r/64$. With an unrolling degree of $r$, the initialization latency is reduced to $\lfloor 1152/r \rfloor$, and the key stream generation throughput increases by a factor of $r$, compared to a round-based implementation. Note that the input for the Trivium round function consists of only 15 bits from the 288-bit internal state. A round input is independent of the previous 65 rounds, which means that we can simultaneously execute up to 64 rounds


288-bit internal state register
1-round datapath
288-bit internal state register
64-round parallel datapath
288-bit internal state register
64-round parallel datapath
64-round parallel datapath
× p

(a) Round-based (b) 64-round unrolled (c) 64$p$-round unrolled

Figure 1: Overview of round-based and unrolled data flows of Trivium.

Table 1: Comparison of Trivium hardware architectures

| | Round-based | 64-round unrolled | 64$p$-round unrolled |
|---|---|---|---|
| Area for combinational logic | **1 round datapath** | 64 round datapaths | 64$p$ round datapaths |
| Area for internal state regs. | 288 bits | 288 bits | 288 bits |
| Initialization clock cycles | 1152 | 18 (= 1152/64) | **18/$p$ (= 1152/64$p$)** |
| Throughput per cycle | 1 bit | 64 bits | **64$p$ bits** |
| Critical delay | **1 round** | **1 round** | $p$ rounds |
| Masked implementation | **Known** | **Known** | Unknown and non-trivial |
| Side-channel attack | Known | Known | **This paper** |

using a parallel datapath[1], as illustrated in Figure 1(b). For an unrolling degree of 64$p$, $p$ 64-round parallel datapaths need to be used in a serial manner, as shown in Figure 1(c).

Table 1 displays a comparison of the aforementioned three types of Trivium hardware architectures. The Trivium round function is considerably smaller than those of block ciphers, and it is implemented using only a few logic gates, while the circuit area is mostly occupied with sequential logic for the internal state registers. Therefore, round-based hardware consumes most of the energy for the sequential logic, namely, storing the internal state in every clock cycle. By contrast, unrolled hardware significantly accelerates the completion of initialization and key stream generation compared to the round-based counterpart, whereas it requires a larger area for the unrolled datapath(s). This unrolled architecture saves energy by reducing computational time, notwithstanding modest increase in area and power consumption incurred by unrolled datapath(s). Therefore, unrolled Trivium hardware surpasses its round-based counterpart in energy efficiency. In [BMA+18, CBT+21], it has been indicated that unrolling degrees ranging from 64 to 288 are not only practical in terms of performance but also result in higher energy efficiency.

Masking of round-based implementation, as presented in [SA15, MHBM+18], can be trivially extended to 64-round unrolled implementation because 64-round unrolled datapath has the algebraic degree same as round-based one. However, masked 64$p$-round unrolled implementations, which has a datapath of higher algebraic degree, remain unknown. It remains as the practical challenges in masking the 64$p$-round datapath due to its high algebraic degree, as discussed in [MS16, Moo20].

## 2.4 Existing SCAs on Trivium

### 2.4.1 CPA/DPA on initialization

Major existing SCAs on Trivium focus on the first few rounds of the initialization, before the secret key and IV are sufficiently diffused [FGKV06, SPK09, JHWW12, TSA15a, TSA15b, SJB21]. The attacker presumably can trigger the encryption/decryption with different IVs and an unknown fixed key, given the nature of Trivium as a synchronous stream cipher.

[1]Reference and practical software implementations of Trivium on $r$-bit microcontroller/CPU execute $r$ rounds in parallel using $r$-bit registers in a word-slicing manner.

The existing CPAs target the secret intermediate values $t_1^i$ for $i = 1$ to 80.[2] The intermediate value of $t_1^i$ is represented as $t_1^i = s_{66}^i + s_{93}^i + s_{91}^i s_{92}^i + s_{171}^i$ as in Algorithm 1. Using the secret key $k$ and IV $v$, the intermediate polynomial $t_1^i$ is represented by

$$t_1^1 = k_{66} + 0 + 0 \cdot 0 + v_{78}, \quad t_1^2 = k_{65} + 0 + 0 \cdot 0 + v_{77},$$
$$\cdots$$
$$t_1^{79} = k_{57} + k_{15} + k_{13}k_{14} + k_{66} + v_{78}, \quad t_1^{80} = k_{56} + k_{14} + k_{12}k_{13} + k_{65} + v_{77}.$$

As observed here, $t_1^i$ is given as a sum of terms of $k$ and a term of $v$. Therefore, these intermediate values are linearly decomposed into two subpolynomial functions $f^i(k)$ and $g^i(v)$ consisting of partial bits of $k$ and $v$, respectively. Thus, $t_1^i$ is represented as

$$t_1^i(k, v) = f^i(k) + g^i(v) = \sigma^i + g^i(v). \tag{1}$$

where $\sigma^i = f^i$ is the target key polynomial. We use CPA and the leakage of computation of $t_1^i$ to estimate $\sigma^i$, as $v$ is known. We consider $t_1^i$ as a selection function, $v$ as a known value like a plaintext, and $\sigma^i$ as the secret value to be guessed, similarly to the CPA on AES. Accordingly, the attacker calculates the Pearson's correlation coefficient between the leakage and hypothetical power consumption of $t_1^i$, based on a hypothesis of $\sigma^i$. Note that the estimation of $\sigma^i$ can be performed by a multi-bit CPA in case of unrolled implementation.

This CPA and key recovery method exploit $(\sigma^1, \sigma^2, \ldots, \sigma^{80})$, derived from some first rounds of initialization. In unrolled hardware implementations with a large unrolling degree (exceeding 192), estimating these values from side-channel leakage becomes difficult, as intermediate values of some first rounds are not stored in the state register. Furthermore, reversing the secret key from the latter internal states estimated by CPA, whose values are more diffused and given from higher-degree polynomials, is nontrivial. Nevertheless, no study on the SCA security of unrolled Trivium hardware implementation with large unrolling degree (*i.e.*, 288), which is deemed practical and most energy-efficient, is known.

### 2.4.2 SCA for internal state recovery

In TCHES 2022, Kumar *et al.* presented a profiled SCA on stream ciphers to recover the internal state [KDB+22]. This attack does not necessarily focus on the initialization (although possible), but it requires to trigger only a single encryption/decryption with an IV. The SCA attacker exploits leakage (*i.e.*, HW or HD) of internal states of consecutive rounds from $i$-th to $(i + N)$-th, where $i$ is an arbitrary integer and $N$ is an integer determined for attack success. The attacker first performs a deep-learning training to develop a neural network that predicts the Hamming weight (HW) for software implementations or Hamming distance (HD) for hardware implementations. Subsequently, a MILP modeling and solver are used to correct the errors in estimated HWs/HDs. Finally, the attacker estimates the internal state from the predicted HW/HD using an SMT solver.

However, this attack assumes the availability of leakages of *consecutive* rounds (*i.e.*, $s^i, s^{i+1}, \ldots, s^{i+N}$), which corresponds to a round-based implementation. As aforementioned, practical Trivium software implementation uses word-slicing to exploit the parallelism of Trivium and CPU/microcontroller. For unrolled implementations, the round-by-round leakage is unavailable because $r$ rounds are simultaneously computed and these internal states is not stored in register. This indicates that, for an $r$-round unrolled implementation, HW/HD used in the SMT/MILP solver is only observable once per $r$ rounds, but HW/HD from $r - 1$ rounds between them rounds are unavailable for the attacker. Namely, the attacker can obtains only leakages of $s^i, s^{i+r}, s^{i+2r} \ldots, s^{i+Nr}$ for an $r$-round unrolled implementation. The error-correcting strategy using MILP solver

[2]Note that both conventional and proposed attack can exploit $t_2^i$ and $t_3^i$ similarly to $t_1^i$, while we here focus on $t_1^i$ for the explanation.

makes no sense because the MILP modeling includes no constraint about relationship between HWs/HDs. In addition, as the intermediate state is diffused by $r-1$ rounds, it would be infeasible for an SMT solver to find a unique secret key. In fact, we tried a key recovery using a noise-free leakage of 288-round unrolled implementation by an SMT solver Z3 in Section 3.3.1, but we found it infeasible. Therefore, the profiled SCA is validated only for a round-based implementation, but its applicability to the (practical) unrolled implementation of Trivium is unknown.

## 3 Proposed side-channel linearization attack

### 3.1 Attack scenario and assumptions

There are two major requirements/assumptions for the attacker to mount the proposed SCA:

- The attacker can arbitrarily choose the input (*i.e.*, IV).
- The attacker can trigger the execution of Trivium initialization of the target device with the chosen IV many times.

These requirements have been frequently used in previous studies on SCA on stream ciphers [FGKV06, SPK09, JHWW12, HHN+13, TSA15a, TSA15b, KUH+17, SJB21], as introduced in Section 2.4.1. Such an attack is mainly possible if the attacker targets a decryption device. The IV (nonce) of ciphers is usually given by an upcounter to avoid reuse of IV [BZD+16]. Therefore, an encryption module is frequently stateful to generate the IV securely inside the device, and the attacker cannot control it in this case. However, decryption device basically performs cryptographic operations for a given query. Even if the query is malicious, the decryption device cannot detect the invalidity of queries until completing decryption (or authentication). Thus, such assumptions are sometimes practical for SCA attacker, as many SCA studies have evaluated ciphers in this scenario (even for AES).

Note that decryption device may also be stateful to check the validity of IV for the protection against adaptive attacks (*i.e.*, detect malicious/invalid queries prior to decryption) like [UHIM23, Section 3.2]. The proposed SCA cannot be carried out even on decryption device in such a case. However, having a state for IV on embedded devices incurs a non-negligible cost of non-volatile and secure memory in addition to its operational cost. Thus, evaluation of SCA in this scenario is very important to assess the side-channel vulnerability of practical cryptographic implementation.

### 3.2 Estimation of $\sigma^i$ at latter rounds

#### 3.2.1 Problem statement

An $r$-round unrolled implementation updates the internal state by $r$ rounds within a single clock cycle. The diffusion of the entire internal state in Trivium is achieved over 111 rounds, equivalent to the bit length of NLFSR-3. This implies that the outputs of the first few rounds are not stored in registers if the implementation uses more-than 111-round unrolling. Specifically, for an $r$-round unrolled implementation, the output bits (*i.e.*, $t_1^i$, $t_2^i$, and $t_3^i$) of the first $r-111$ rounds are not stored in registers, rendering their side-channel leakage unavailable. Consequently, an SCA attacker is required to estimate $\sigma^i$ for $t_1^i$ of the latter rounds, specifically for $i > r-111$. However, this poses a significant challenge for existing SCAs, as they exploit the fact that the intermediate values of the first few rounds, which are the primary targets of these attacks, can be represented by a linear composition of a key polynomial and an IV polynomial. By contrast, the intermediate values of the latter rounds are represented by non-linear polynomials. Thus, we developed the RRLD strategy for the estimation of $t_1^i$ of the latter rounds, which is designed to linearly decompose $t_1^i$ by restricting (*i.e.*, choosing) the IV such that the key–IV polynomial transforms into a linear composition. Our proposal includes a methodology for determining and computing such a chosen-IV set.

**Available leakage.** The attacker only uses the leakage of intermediate values stored in register. More precisely, the attacker exploits the register transition and estimate HD of intermediate values as in many conventional CPAs. In contrast, leakage of wire transition is supposed to be unavailable as mentioned in Section 1. This indicates that intermediate values which are not stored in register is unavailable for the attacker through leakage; and therefore, intermediate values at earlier rounds are assumed to be unexploitable by SCA on unrolled implementations. This assumption is because the CPA focusing on wire value/glitches is far more difficult than that focusing on register value.

### 3.2.2 Example of proposed CPA: Determination of $\sigma^{168}$ and its estimation

The proposed algorithm recursively applies the following procedure, named restricted linear decomposition (RLD).

Let us first describe an example with $i = 168$ to briefly explain our idea before describing the proposed algorithm. Note that the proposed algorithm is very general for unrolled Trivium implementation with any unrolling degree, although we take some example values and polynomials for explanation. To develop the RRLD strategy, we first symbolically represent the intermediate value $t_1^i$ in a Product of Sum (PoS) form[3], derived through a repeated substitution of $k$ and $v$ into the Trivium round function $t_1^i = s_{66}^i + s_{93}^i + s_{91}^i s_{92}^i + s_{171}^i$. Subsequently, we linearly decompose the resultant polynomial into three polynomials. The PoS representation of $t_1^{168}$ is given by

$$\begin{aligned} t_1^{168} = k_{37} + k_{64} + k_{46} + k_{55} + k_4 + k_2k_3 \\ + v_{34} + v_{49} + v_{47}v_{48} + v_{61} + v_{76} + v_{74}v_{75} + v_{67} \\ + (k_{62} + v_{59} + v_{74} + v_{72}v_{73})(k_{63} + v_{60} + v_{75} + v_{73}v_{74}). \end{aligned}$$

As well as Equation (1), we aim to linearly decompose $t_1^{168}$ into the polynomial function of key $f^{168}(k)$ and IV $g^{168}(v)$. However, we cannot obtain such a linear decomposition because $t_1^{168}$ includes a non-linear term related to bits of $k$ and $v$. Thus, $t_1^{168}$ can be represented as

$$t_1^{168}(k, v) = f^{168}(k) + g^{168}(v) + h^{168}(k, v), \tag{2}$$

where $f_1^{168} : \mathcal{K} \to \mathbb{F}_2$, $g_1^{168} : \mathcal{V} \to \mathbb{F}_2$, and $h_1^{168} : \mathcal{K} \times \mathcal{V} \to \mathbb{F}_2$ are given by

$$\begin{aligned} f^{168}(k) &= k_{37} + k_{64} + k_{46} + k_{55} + k_4 + k_2k_3, \\ g^{168}(v) &= v_{34} + v_{49} + v_{47}v_{48} + v_{61} + v_{76} + v_{74}v_{75} + v_{67}, \\ h^{168}(k, v) &= (k_{62} + v_{59} + v_{74} + v_{72}v_{73})(k_{63} + v_{60} + v_{75} + v_{73}v_{74}). \end{aligned}$$

Note that such a representation involving three polynomials generally exists in $t_1^i$ for all $i$, and some of them can be 0.

The value of $h^{168}$ varies depending on some bits of $k$ and $v$, and thus it is not constant. Consequently, an attacker cannot guess the value of $\sigma^{168} = f^{168}$ by CPA, as the side-channel leakage depends on the unknown variable $h^{168}$. Unlike CPAs on block ciphers, we

[3]In this paper, the PoS form did not necessarily need to be canonical or reduced; it could include redundancies.

should estimate a key polynomial for Trivium but not secret key itself, as guessing all key bits in $t_1^i$ becomes infeasible for larger $i$. Therefore, the impact of $h^{168}$ is critical, resulting in difficulty of SCA on the latter rounds of Trivium.

We remove the above difficulty using a well-calibrated chosen-IV set, determined by the RRLD strategy. First, we linearly decompose $h^i(k, v)$ into $d$ product polynomials $h_1^i, h_2^i, \ldots, h_l^i, \ldots, h_d^i$. Namely, we represent $h^i$ in a sum of PoS form as

$$h^i(k, v) = \sum_l h_l^i(k, v), \quad h_l^i(k, v) = \prod_u \left(f_{l,u}^i(k) + g_{l,u}^i(v) + h_{l,u}^i(k, v)\right), \tag{3}$$

because of the structural recursiveness of the PoS form (*i.e.*, a product of PoS polynomials should be a PoS polynomial). In case of $i = 168$, we derive $d = 1$ and $h_1^{168} = h^{168}$, and it holds that

$$h_1^{168}(k, v) = \left(f_{1,1}^{168}(k) + g_{1,1}^{168}(v) + h_{1,1}^{168}(k, v)\right)\left(f_{1,2}^{168}(k) + g_{1,2}^{168}(v) + h_{1,2}^{168}(k, v)\right),$$

where the subpolynomials are given by

$$f_{1,1}^{168} = k_{62}, \quad f_{1,2}^{168} = k_{63}, \quad h_{1,1}^{168} = h_{1,2}^{168} = 0,$$
$$g_{1,1}^{168} = v_{59} + v_{74} + v_{72}v_{73}, \quad g_{1,2}^{168} = v_{60} + v_{75} + v_{73}v_{74}.$$

Here, if $g_{1,1}^{168}$ and $g_{1,2}^{168}$ are fixed, then $t_1^{168}$ can be expressed as a sum of key polynomials and IV polynomials. In the proposed SCA, we set the value of $g_{1,1}^{168}$ and $g_{1,2}^{168}$ as $c_{1,1}^{168}$ and $c_{1,2}^{168} \in \mathbb{F}_2$, respectively, indicating that $g_{1,1}^{168}(v) = c_{1,1}^{168}$ and $g_{1,2}^{168}(v) = c_{1,2}^{168}$ ($c_{1,u}^{168}$ can be either 0 or 1). Recall that $h_1^{168}$ is defined as a polynomial function $h_1^{168} : \mathcal{K} \times \mathcal{V} \to \mathbb{F}_2$. In the proposed SCA, we use a restriction of $h_1^{168}$ to

$$\mathcal{V}^{168} = \left\{g_{1,1}^{168}(v) = c_{1,1}^{168}\right\} \cap \left\{g_{1,2}^{168}(v) = c_{1,2}^{168}\right\} \subseteq \mathcal{V},$$

corresponding to a chosen-IV set. Note that, for any function $g : \mathcal{V} \to \mathbb{F}_2$ and element $c \in \mathbb{F}_2$, it always holds that $g|_{\{g(v)=c\}} = c$, as mentioned in Section 2.1. Let $c_{1,1}^{168} = c_{1,2}^{168} = 0$ for example. The restricted $h_1^{168}$ is given by

$$h_1^{168}\big|_{\mathcal{K}\times\mathcal{V}^{168}} = \left(f_{1,1}^{168}(k) + 0 + 0\right)\left(f_{1,2}^{168}(k) + 0 + 0\right) = f_{1,1}^{168} f_{1,2}^{168}.$$

For simplicity, we hereafter write $h|_{\mathcal{K}\times\mathcal{V}'}$ for a given function $h : \mathcal{K} \times \mathcal{V} \to \mathbb{F}_2$ and set $\mathcal{V}' \subseteq \mathcal{V}$ as $h|_{\mathcal{V}'}$, because we did not consider any subset of $\mathcal{K}$ at all in this paper. Note that the computation to derive $\mathcal{V}^i$ can be readily realized using a common SAT or SMT solver such as Z3 [dMB08, Mic23].

Finally, we identify the target key polynomial $\sigma^i$ to be estimated by CPA. As we use a chosen-IV set of $\mathcal{V}^{168}$, we should consider $t_1^{168}$ restricted to $\mathcal{V}^{168}$, denoted by

$$\begin{aligned} t_1^{168}(k, v)\big|_{\mathcal{V}^{168}} &= f^{168}(k) + g^{168}(v)\big|_{\mathcal{V}^{168}} + h^{168}(k, v)\big|_{\mathcal{V}^{168}} \\ &= f^{168} + f_{1,1}^{168} f_{1,2}^{168} + g^{168}\big|_{\mathcal{V}^{168}}, \end{aligned} \tag{4}$$

if $c_{1,1}^{168} = c_{1,2}^{168} = 0$. In this case, the target key polynomial is given by

$$\sigma^{168} = f^{168} + f_{1,1}^{168} f_{1,2}^{168} = k_{37} + k_{64} + k_{46} + k_{55} + k_4 + k_2k_3 + k_{62}k_{63}.$$

Meanwhile, Equation (4) is represented by

$$t_1^{168}(k, v)\big|_{\mathcal{V}^{168}} = \sigma^{168} + g^{168}\big|_{\mathcal{V}^{168}}.$$

Thus, $t_1^{168}$ is linearly decomposed into a key polynomial and restricted IV polynomial; the attacker can estimate $\sigma^{168}$ by CPA with a chosen-IV set of $\mathcal{V}^{168}$. This strategy is termed as *restricted linear decomposition (RLD)*. Here, it suffices *for $\sigma^{168}$ estimation* to set $c_{1,1}^{168}$ and $c_{1,2}^{168}$ to an arbitrary value of 0 or 1. Then, we present another strategy about how to determine the above constants *for feasible full key recovery* in Section 3.3.

*Remark* 2 (Infeasibility of naïve CPA). An attacker may mount a CPA that directly guesses all key bits in $h^i$. However, such a naïve CPA is feasible only for earlier rounds, but guess of all key bits in $h^i$ is infeasible to estimate $\sigma^i$ for latter rounds due to the diffusion. For example, one-bit CPA focusing on $\sigma^{264}$ and $\sigma^{288}$ needs to guess 16 and 26 bits, respectively. It takes non-negligible cost for full key recovery to repeat such CPA of high complexity for all target rounds. For larger $i$, we have to guess more bits, resulting in an infeasible attack. In contrast, the proposed CPA requires only one-bit guess per one-bit CPA (and eight-bit guess per eight-bit CPA) independently of $i$. Thus, our RRLD strategy realizes the estimation of $\sigma^i$ for large $i$ with a feasible complexity. Moreover, an increase in the number of key candidates would degrade the success rate of SCA.

### 3.2.3 Algorithmic description

In the example of Section 3.2.2, the key–IV polynomial $h^{168}$ does not contain key–IV subpolynomials (*i.e.*, $h^{168}_{1,1} = h^{168}_{1,2} = 0$). Generally, for larger $i$, the key–IV polynomial $h^i$ is likely to contain non-trivial key–IV subpolynomials, with $h^i_{l,u} \neq 0$, due to diffusion, as expressed in Equation (3). This means that one application of RLD does not always result in $h = 0$ if we consider a general polynomials. Consequently, the proposed RRLD strategy applies RLD recursively to the key–IV subpolynomials. The degree of $h$ obtained by an RLD is always smaller than the input polynomial; therefore, by recursively applying the RLD to $h$, the degree of RLD output eventually becomes 0, indicating a complete linear decomposition. Thus, RRLD can always achieve a complete linear decomposition for general polynomials thanks to the recursion of RLD. Finally, we identify $\sigma^i$ (to be estimated by CPA) using all the key subpolynomials, and use a restriction (*i.e.*, chosen-IV set) as an intersection of all $\{g^i_{l,u}(v) = c^i_{l,u}\}$ for each RLD. For example, let us consider $t^i_1 = f^i + g^i + h^i$, where $h^i$ is given by

$$h^i = \left(f^i_{1,1} + g^i_{1,1} + h^i_{1,1}\right)\left(f^i_{1,2} + g^i_{1,2} + h^i_{1,2}\right) + \left(f^i_{2,1} + g^i_{2,1} + h^i_{2,1}\right)\left(f^i_{2,2} + g^i_{2,2} + h^i_{2,2}\right),$$

with non-zero $h^i_{l,u}$. Using a restriction to $\mathcal{V}^i = \{g^i_{1,1}(v) = c^i_{1,1}\} \cap \{g^i_{1,2}(v) = c^i_{1,2}\} \cap \{g^i_{2,1}(v) = c^i_{2,1}\} \cap \{g^i_{2,2}(v) = c^i_{2,2}\} \subseteq \mathcal{V}$, where $c^i_{l,u}$ denotes a constant value of 0 or 1, it can be represented as

$$\begin{aligned} h^i\big|_{\mathcal{V}^i} = &\left(f^i_{1,1} + c^i_{1,1} + h^i_{1,1}\big|_{\mathcal{V}^i}\right)\left(f^i_{1,2} + c^i_{1,2} + h^i_{1,2}\big|_{\mathcal{V}^i}\right) \\ &+ \left(f^i_{2,1} + c^i_{2,1} + h^i_{2,1}\big|_{\mathcal{V}^i}\right)\left(f^i_{2,2} + c^i_{2,2} + h^i_{2,2}\big|_{\mathcal{V}^i}\right), \end{aligned}$$

because $g^i_{l,u}|_{\{g^i_{l,u}(v)=c^i_{l,u}\}} = c^i_{l,u}$ holds. For all $(l,u)$, the degree of $h^i_{l,u}|_{\mathcal{V}^i}$ should be smaller than that of $h^i$. In addition, each $h^i_{l,u}$ can be represented in the form of Equation (2) (with the restriction). Thus, by applying RLD to its subpolynomials in a recursive manner, we can finally obtain the restricted linear decomposition of $t^i_1$ as

$$t^i_1\big|_{\mathcal{V}^i} = f^i + g^i\big|_{\mathcal{V}^i} + h^i\big|_{\mathcal{V}^i} = \sigma^i + g^i\big|_{\mathcal{V}^i},$$

where $\sigma^i = f^i + h^i\big|_{\mathcal{V}^i}$ represents the target key polynomial and $\mathcal{V}^i$ indicates the restriction (*i.e.*, chosen-IV set) in estimating $\sigma^i$ derived as the intersection of all $\{g(v) = c\}$ during RLDs.

Algorithm 2 describes the determination of $\sigma^i$ and the corresponding chosen-IV set for a general $i$ based on RRLD, given a target $t^i_1$ as input. Note that the determination of constant values of $c_{l,u}$ is described in Section 3.3 since Algorithm 2 works with arbitrary values of $c_{l,u}$. Algorithm 2 consists of recursive calls of the main function RLD$^i$. At Line 2, we first represent the input polynomial using $f$, $g$, and $h$ like Equation (2), which implies that we do not have any restriction to the function domain initially. At Line 4, if $h = 0$,

**Algorithm 2** Identification of $\sigma^i$ and chosen-IV set based on RRLD

```
Input: Polynomial function $t : \mathcal{K} \times \mathcal{V}' \to \mathbb{F}_2$ (where $\mathcal{V}' \subseteq \mathcal{V}$)
Output: Target key polynomial $\sigma^i$ and chosen-IV set $\mathcal{V}^i$
 1: Function RLD$^i(t)$
 2:     $f, g, h \leftarrow$ LinearlyDecompose$(t)$;
 3:     $\mathcal{V}^i = \{0,1\}^{80}$;
 4:     if $h = 0$ then
 5:         return $f + c, \{g(v) = c\}$;
 6:     else
 7:         for each $l$ do
 8:             for each $u$ do
 9:                 $f_{l,u}, g_{l,u}, h_{l,u} \leftarrow$ Extract$(h; l, u)$;      ▷ Extract $(l,u)$-th tuple of subpolynomials of $h$
10:                 $\mathcal{V}_{l,u} \leftarrow \{g_{l,u}(v) = c_{l,u}\}$;      ▷ $c_{l,u} \in \{0,1\}$ is constant
11:                 $\tilde{f}_{l,u}, \tilde{\mathcal{V}}_{l,u} \leftarrow$ RLD$^i\left(h_{l,u}\right)$;      ▷ Apply RLD to $h_{l,u}$ recursively
12:                 $\sigma_{l,u} \leftarrow f_{l,u} + c_{l,u} + \tilde{f}_{l,u}$;
13:                 $\mathcal{V}^i \leftarrow \mathcal{V}^i \cap \mathcal{V}_{l,u} \cap \tilde{\mathcal{V}}_{l,u}$;
14:             end for
15:         end for
16:         $\sigma^i \leftarrow f + \sum_l \prod_u \sigma_{l,u}$;
17:         return $\sigma^i, \mathcal{V}^i$;
18:     end if
19: end Function
```

we return $f + c$ and $\{g(v) = c\}$ for a fixed $c \in \mathbb{F}_2$ as the input polynomial no longer needs decomposing. Otherwise, at Line 9, for each $(l, u)$, we first extract the subpolynomials $f_{l,u}$, $g_{l,u}$, and $h_{l,u}$ from $h$ as in Equation (3). At Line 10, we derive the restriction for the subpolynomial $g_{l,u}$ from fixed $c_{l,u} \in \mathbb{F}_2$, as $\{g_{l,u}(v) = c_{l,u}\}$. Line 11 is the core part of this algorithm, where we recursively call the function RLD$^i$ for $h_{l,u}$ to reduce it until $h_{l,u}$ contains no non-linear composition of $k$ and $v$. Then, at Lines 12 and 13, we derive the target key polynomial and the restriction for the $(l, u)$-th subpolynomial. Here, we take the intersection of $\mathcal{V}_{l,u}$ for all $(l, u)$ because the IV should be restricted simultaneously with regard to all $g_{l,u}$ to estimate the key polynomial corresponding to the input polynomial. At Line 16, we compute the resulting key polynomial corresponding to the input polynomial, as in the form of Equation (3). Finally, the RLD function returns the key polynomial and chosen-IV set for the input polynomial. Using Algorithm 2, we can exploit the leakage of $t_1^i$ relating to the corresponding key polynomial, which indicates an SCA on the $i$-th round intermediate value in the unrolled implementation, without using the leakage from prior rounds. Note that, for practical computation, sets $\mathcal{V}^i$, $\mathcal{V}_{l,u}$, and $\mathcal{V}_{l,u}$ should be represented by their corresponding function $g_{l,u}$. The actual elements of the input set (*i.e.*, chosen IVs) should be finally computed from the functions after the execution of Algorithm 2.

Algorithm 2 always terminates within a finite number of steps, because one application of RLD eliminates at least one non-linear term of $k$ and $v$, and the recurrence ends if there are no remaining non-linear terms. However, for large $i$ (and multi-bit CPA), the size of chosen-IV set $\mathcal{V}^i$ can be small because $\mathcal{V}^i$ is given by the intersection of $\{g(v) = c\}$ and the number of RLD calls becomes greater due to the diffusion. Ultimately, $\mathcal{V}^i$ can be an empty set if the internal state is sufficiently diffused, which indicates that the attack is inapplicable to such $i$. As well, if $g^i|_{\mathcal{V}^i} = 0$ or $g^i|_{\mathcal{V}^i} = 1$ (*i.e.*, if we cannot control the value of $g^i$ with $\mathcal{V}^i$), then the attack is deemed infeasible, which indicates that the size of the chosen-IV set determines the feasibility of the proposed SCA with regard to larger $i$. Additionally, the success rate of CPA in estimating $\sigma^i$ depends on the number of available chosen-IV patterns (*i.e.*, the size of the chosen-IV set), similar to the SCA on AES [FPS12]. In other words, the CPA success rate gets worse if the variety of IVs is insufficient.

In Section 4, we experimentally evaluate the relation between the success rate and the size of the chosen-IV set, in addition to the bit length of CPA, using an actual Trivium unrolled hardware implementation. Moreover, in Section 5, we discuss the relation between the unrolling degree and the size of the chosen-IV set to analyze the leakage resilience of unrolled Trivium hardware.

**Extension to multi-bit CPA.** Algorithm 2 corresponds to one-bit CPA; however, it can be readily extended to an $n$-bit CPA, which estimates $\sigma^i$ for the $i$-th to $(i+n-1)$-th rounds simultaneously, by executing Algorithm 2 for $t_1^i, t_1^{i+1}, \ldots, t_1^{i+n-1}$ and taking the intersection of the respective chosen-IV sets (*i.e.*, $\mathcal{V}^i \cap \mathcal{V}^{i+1} \cap \cdots \cap \mathcal{V}^{i+n-1}$).

## 3.3 Feasible key recovery based on linearization

### 3.3.1 Problem statement and basic concept

Although the SCA attacker can successfully estimate $\sigma^i$'s by CPA, it is non-trivial to recover the secret key from them, especially for larger value of $i$, corresponding to the unrolling degree. This is because, if $i$ is large, $\sigma^i$ is given by a high degree polynomial due to diffusion. Although we have many equations of $\sigma^i = f^i(k) + h^i(k,v)\big|_{\mathcal{V}^i}$ by CPA, it would be infeasible to solve the system of equations due to their high degree. In fact, for a fixed key $k$, then we derived $\sigma^{209}, \sigma^{210}, \ldots, \sigma^{288}$, created the system of equations, and then attempted to solve it using Z3Py with the aforementioned CPU. However, we found that we could not recover the secret key within a week. Therefore, we needed a new strategy to recover the secret key from CPA results with a practical complexity.

Our basic idea is to directly derive the value of a linear relation between two key bits $k_j + k_{j'}$ by removing its non-linear term $k_j k_{j'}$ using two CPA results for different constant $c_{l,u}$. In Section 3.2, we did not mention how the constant $c_{l,u}$ should be determined, but always let $c_{l,u} = 0$ for a simple explanation. We show that we can directly obtain the value of $k_j + k_{j'}$ for some $j$ and $j'$ by carefully determining the value of $c_{l,u}$. After obtaining these values for many $k_j$ and $k_{j'}$ pairs, we solve the system of equations using Gaussian elimination, which has a complexity of $O(m^3)$, where $m$ denotes the number of variables in the equations. We first explain our idea using a linearization example of $\sigma^{168}$ and then generalize the proposed linearization method as well as Section 3.2.

### 3.3.2 Example of $\sigma^{168}$

Recall that $t_1^{168} = f^{168} + g^{168} + h^{168}$, where $h^{168} = h_1^{168} = (f_{1,1}^{168} + g_{1,1}^{168} + h_{1,1}^{168})(f_{1,2}^{168} + g_{1,2}^{168} + h_{1,2}^{168})$ with $h_{1,1}^{168} = h_{1,2}^{168} = 0$. In the RLD described in Section 3.2.2, we fixed $g_{1,1}^{168} = g_{1,2}^{168} = 0$ using a restriction of $\{g_{1,1}^{168}(v) = 0\} \cap \{g_{1,2}^{168}(v) = 0\}$. In this case, $\sigma^{168}$ is given by $f^{168} + f_{1,1}^{168} f_{1,2}^{168} = k_{37} + k_{64} + k_{46} + k_{55} + k_4 + k_2 k_3 + k_{62} k_{63}$. Thus, by estimating $\sigma^{168}$ using one CPA, we can derive an equation of $\sigma^{168} = k_{37} + k_{64} + k_{46} + k_{55} + k_4 + k_2 k_3 + k_{62} k_{63}$, but it contains non-linear terms.

We then explain how to derive a linear relation between key bits by two CPAs. For $c \in \mathbb{F}_2$, let $\mathcal{V}^{168}(c) = \{g_{1,1}^{168}(v) = c\} \cap \{g_{1,2}^{168}(v) = c\}$ and let $\sigma^{168}(c) = f^{168} + (f_{1,1}^{168} + c)(f_{1,2}^{168} + c)$. When $c = 0$ and 1, they are given by

$$\sigma^{168}(0) = f^{168} + f_{1,1}^{168} f_{1,2}^{168}, \tag{5}$$

$$\begin{aligned} \sigma^{168}(1) &= f^{168} + (f_{1,1}^{168} + 1)(f_{1,2}^{168} + 1) \\ &= f^{168} + f_{1,1}^{168} f_{1,2}^{168} + f_{1,1}^{168} + f_{1,2}^{168} + 1, \end{aligned} \tag{6}$$

respectively. Both of them can be estimated by CPA if $\mathcal{V}^{168}(c)$ is not empty (which is actually true). Then, by taking the difference between $\sigma^{168}(0)$ and $\sigma^{168}(1)$, Equations (5) and (6) are followed by

$$\sigma^{168}(0) + \sigma^{168}(1) = f_{1,1}^{168} + f_{1,2}^{168} + 1 = k_{62} + k_{63} + 1,$$

which indicates that we can eliminate the non-linear term $f_{1,1}^{168}f_{1,2}^{168}$ $(= k_{62}k_{63})$ if both $\sigma^{168}(0)$ and $\sigma^{168}(1)$ are available. Thus, in the proposed side-channel linearization attack on $i = 168$, we perform CPA twice on

$$t_1^{168}|_{\mathcal{V}^{168}(0)} = \sigma^{168}(0) + g^{168}|_{\mathcal{V}^{168}(0)}, \quad t_1^{168}|_{\mathcal{V}^{168}(1)} = \sigma^{168}(1) + g^{168}|_{\mathcal{V}^{168}(1)},$$

to estimate $\sigma^{168}(0)$ and $\sigma^{168}(1)$, respectively. Finally, we derive the linear relation $k_{62}+k_{63}$ as $\sigma^{168}(0) + \sigma^{168}(1) + 1$.

### 3.3.3 General description

Recall that, in general, the key–IV subpolynomial of $t_1^i$, namely $h^i$, is given in the form of

$$h^i = \sum_l h_l^i, \quad h_l^i = \prod_u \left(f_{l,u}^i + g_{l,u}^i + h_{l,u}^i\right).$$

For Trivium, we empirically confirmed that, for most $i$ up to 576, there exists $\lambda$ such that $h_\lambda^i = (k_j + g_{\lambda,1}^i)(k_{j'} + g_{\lambda,2}^i)$ for some $j$ and $j'$ (*i.e.*, $f_{\lambda,1}^i = k_j$, $f_{\lambda,2}^i = k_{j'}$, and $h_{l,1}^i = h_{l,2}^i = 0$ hold like $h_1^{168}$)[4]. Thus, the subpolynomial $h^i$ is generally represented as

$$h^i = \sum_{l \neq \lambda} h_l^i + (k_j + g_{\lambda,1}^i)(k_{j'} + g_{\lambda,2}^i).$$

Therefore, we define $\sigma^i(0)$ and $\sigma^i(1)$ to obtain $\sigma^i(0) + \sigma^i(1) = k_j + k_{j'} + 1$ such that we apply the linearization strategy to $h_\lambda^i$, while $\sum_{l \neq \lambda} h_l^i$ is fixed for both $\sigma^i(0)$ and $\sigma^i(1)$. Let $\mathcal{V}_{l,u}^i = \{g_{l,u}^i(v) = c_{l,u}^i\}$ and $\tilde{\mathcal{V}}_{l,u}^i$ denote the output set of $\mathrm{RLD}^i(h_{l,u}^i)$ (*i.e.*, the restriction for the key–IV subpolynomial $h_{l,u}^i$), as defined in Algorithm 2. Formally, for $c \in \mathbb{F}_2$, the restriction for CPA estimation with linearization is defined as

$$\mathcal{V}^i(c) = \bigcap_{l \neq \lambda} \bigcap_u \left(\mathcal{V}_{l,u}^i \cap \tilde{\mathcal{V}}_{l,u}^i\right) \cap \left(\{g_{\lambda,1}^i(v) = c\} \cap \{g_{\lambda,2}^i(v) = c\}\right).$$

Here, it suffices to use arbitrary constant values for $c_{l,u}^i \in \mathbb{F}_2$ for each $l \neq \lambda$ and $u$; therefore, we omitted them from the input. For example, we can set $c_{l,u}^i = 0$ for all $l \neq \lambda$ and $u$ (this is the same for its key–IV subpolynomial during the recurrent calls of RLD). Accordingly, the restricted target polynomial $t_1^i|_{\mathcal{V}^i(c)}$ is given by

$$\begin{aligned} t_1^i|_{\mathcal{V}^i(c)} &= f^i + g^i|_{\mathcal{V}^i(c)} + \sum_{l \neq \lambda} h_l^i|_{\mathcal{V}_l^i} + (f_{\lambda,1}^i + g_{\lambda,1}^i|_{\{g_{\lambda,1}^i(v)=c\}})(f_{\lambda,2}^i + g_{\lambda,2}^i|_{\{g_{\lambda,2}^i(v)=c\}}) \\ &= f^i + g^i|_{\mathcal{V}^i(c)} + \sum_{l \neq \lambda} h_l^i|_{\mathcal{V}_l^i} + (k_j + c)(k_{j'} + c), \end{aligned}$$

where $\mathcal{V}_l^i = \bigcap_u (\mathcal{V}_{l,u}^i \cap \tilde{\mathcal{V}}_{l,u}^i)$. The target key polynomial $\sigma^i(c)$ is defined as

$$\sigma^i(c) = f^i + \sum_{l \neq \lambda} h_l^i|_{\mathcal{V}_l^i} + (k_j + c)(k_{j'} + c).$$

Note that $\sum_{l \neq \lambda} h_l^i|_{\mathcal{V}_l^i}$ is a key polynomial because of the restriction by RRLD. Therefore, for both $c = 0$ and 1, we can derive $\sigma^i(c)$ as a concrete polynomial and its corresponding

[4]A few $i$ do not have such $\lambda$ utilized in the proposed SCA, as $g_{\lambda,1}^i$ and $g_{\lambda,2}^i$ are not controllable by the chosen-IV set derived by RRLD for such $i$.

chosen-IV set $\mathcal{V}^i(c)$ using Algorithm 2 by letting $c^i_{\lambda,1} = c^i_{\lambda,2} = c$, while $c^i_{l,u} = 0$ for all $l \neq \lambda$ and $u$. We can easily confirm that

$$\sigma^i(0) = f^i + \sum_{l\neq\lambda} h^i_l\big|_{\mathcal{V}^i_l} + k_j k_{j'},$$
$$\sigma^i(1) = f^i + \sum_{l\neq\lambda} h^i_l\big|_{\mathcal{V}^i_l} + k_j k_{j'} + k_j + k_{j'} + 1,$$

which are followed by

$$\sigma^i(0) + \sigma^i(1) = k_j + k_{j'} + 1,$$

because $f^i + \sum_{l\neq\lambda} h^i_l\big|_{\mathcal{V}^i_l}$ is a fixed value for the restriction.

## 3.4 Wrap up: Flow of side-channel linearization attack

The proposed SCA consists of five steps, among which the first, second, fourth, and fifth are offline computations, whereas the third step requires online side-channel measurements. We also mention estimated time for each step in attacking a 288-round unrolled hardware, as mainly targeted in Section 4.

1. We identify a polynomial of $(k_j + g^i_{\lambda,1})(k_{j'} + g^i_{\lambda,2})$ in $t^i_1$ for targeted $i$ values[5]. The target $i$ is determined directly from the unrolling degree of the implementation, and it took little time to identify the polynomials because it is a symbolic computation.

2. We derive $\sigma^i(c)$ and its chosen-IV set $\mathcal{V}^i(c)$ using Algorithm 2 for each $c \in \mathbb{F}_2$. The execution of Algorithm 2 in our environment took only several minutes.

3. We acquire the side-channel traces with controlling inputs as an online phase. This step was bottleneck of our attack in terms of attack duration for our environment; it took about eight hours to acquire 360,000 traces, which is required for full-key recovery.

4. For each target $i$, we perform CPA twice to estimate $\sigma^i(0)$ and $\sigma^i(1)$ (which can be multi-bit CPA to reduce the total number of traces) and obtain the value of $k_j + k_{j'} + 1$ for the corresponding $(j, j')$. In case of attack on 288-round unrolled hardware, it only requires $2 \times 9$ byte-wise CPAs, which were completed in several minutes.

5. After collecting a sufficient number of linear relations between key bits, we recover the secret key by solving the system of linear equations using Gaussian elimination. In case of attack on 288-round unrolled implementation, the execution of this step was completed by a symbolic computation followed by a brute-force of $2^{29}$ key guesses (see Section 4). The brute-force would be computational intensive but sufficiently feasible.

## 4 Experimental validation

### 4.1 Experimental setup

We demonstrate the validity of the proposed SCA through experimental attacks using an unrolled Trivium hardware implementation on an FPGA. Table 2 presents a summary

[5]Here, $\lambda$ should be selected such that $(k_j + g^i_{\lambda,1})(k_{j'} + g^i_{\lambda,2})$ should not be included in other subpolynomials of $h^i_{l,u}$. For multi-bit CPAs, this exclusion should be considered for all target $i$.

Table 2: Experimental setup

| Device | Xilinx Kintex-7 |
|---|---|
| Board | SAKURA-X |
| Oscilloscope | KEYSIGHT InfiniiVision DSOX6004A |
| Clock frequency | 24 MHz |
| Number of traces | 700,000 |

Table 3: Target polynomials and corresponding linear relation between key bits in the proposed SCA on the 288-round unrolled hardware implementation

| Target key polynomial | Corresponding linear relation between key bits |
|---|---|
| $z_1^{219} = (\sigma_1^{219}, \sigma_1^{220}, \ldots, \sigma_1^{226})$ | $k_{11} + k_{12}, k_{10} + k_{11}, \ldots, k_4 + k_5$ |
| $z_1^{236} = (\sigma_1^{236}, \sigma_1^{237}, \ldots, \sigma_1^{243})$ | $k_4, k_2 + k_3, k_1 + k_2, k_{69} + k_1, k_{68} + k_{69}, \ldots, k_{65} + k_{66}$ |
| $z_1^{244} = (\sigma_1^{244}, \sigma_1^{245}, \ldots, \sigma_1^{251})$ | $k_{64} + k_{65}, k_{63} + k_{64}, \ldots, k_{57} + k_{58}$ |
| $z_1^{252} = (\sigma_1^{252}, \sigma_1^{253}, \ldots, \sigma_1^{259})$ | $k_{56} + k_{57}, k_{55} + k_{56}, \ldots, k_{49} + k_{50}$ |
| $z_1^{257} = (\sigma_1^{257}, \sigma_1^{258}, \ldots, \sigma_1^{264})$ | $k_{51} + k_{52}, k_{50} + k_{51}, \ldots, k_{44} + k_{45}$ |
| $z_2^{256} = (\sigma_2^{256}, \sigma_2^{257}, \ldots, \sigma_2^{263})$ | $k_{43} + k_{44}, k_{42} + k_{43}, \ldots, k_{36} + k_{37}$ |
| $z_2^{264} = (\sigma_2^{264}, \sigma_2^{265}, \ldots, \sigma_2^{271})$ | $k_{35} + k_{36}, k_{34} + k_{35}, \ldots, k_{28} + k_{29}$ |
| $z_2^{272} = (\sigma_2^{272}, \sigma_2^{273}, \ldots, \sigma_2^{279})$ | $k_{27} + k_{28}, k_{26} + k_{27}, \ldots, k_{20} + k_{21}$ |
| $z_2^{280} = (\sigma_2^{280}, \sigma_2^{281}, \ldots, \sigma_2^{287})$ | $k_{19} + k_{20}, k_{18} + k_{19}, \ldots, k_{12} + k_{13}$ |

of the experimental setup. In this experiment, we used the 288-round unrolled hardware, which was identified as optimal in energy efficiency [CBT+21], as discussed in Section 2.3. We specifically designed the target implementation for this experiment[6]. The target hardware completes its initialization with 4 (= 1152/288) clock cycles. We obtained power traces during the first clock cycle, which includes the execution of the first 288 rounds and register update. At this time, $(t_1^{205}, t_1^{206}, \ldots, t_1^{288})$, $(t_2^{178}, t_2^{179}, \ldots, t_2^{288})$, and $(t_3^{196}, t_3^{197}, \ldots, t_3^{288})$ are computed and stored in NLFSRs, which indicates that leakage of these intermediate values are available. More precisely, in attacking an $r$-round unrolled implementation, leakages of $t_1^{i_1}$, $t_2^{i_2}$, and $t_3^{i_3}$ for $r - 93 \leq i_1 \leq r$, $r - 84 \leq i_2 \leq r$, and $r - 111 \leq i_3 \leq r$, are available from the first round computation. For clarity, we denote the target key polynomial and its chosen-IV set corresponding to $t_\tau^i$ ($\tau \in \{1, 2, 3\}$) by $\sigma_\tau^i$ and $\mathcal{V}_\tau^i$, respectively (this notation is consistent for other polynomials).

Thus, to perform the proposed SCA on the 288-round unrolled hardware, we focus on $t_1^{219}, t_1^{220}, \ldots, t_1^{226}$ and $t_1^{236}, t_1^{237}, \ldots, t_1^{264}$ as well as $t_2^{256}, t_2^{257}, \ldots, t_2^{287}$ and execute Algorithm 2 for them. Note that some other intermediate values are also available (*e.g.*, $t_3^{288}$), but they are sufficient for full key recovery. This process derived the corresponding target polynomials and chosen-IV sets, which are sufficient for full key recovery. Table 3 illustrates the linear relations between key bits obtained from the target $\sigma_1^i$ or $\sigma_2^i$. Notably, $k_4$ was directly obtained instead of linear relation when using $t_1^{236}(c)$ for CPA. Table 3 lists the target polynomials in an eight-bit-wise manner, as we here performed eight-bit CPAs. Let $z_\tau^{i,n}(c) = (\sigma_\tau^i(c), \sigma_\tau^{i+1}(c), \ldots, \sigma_\tau^{i+n-1}(c))$ represent the tuple of eight target key polynomials corresponding to $t_\tau^i, t_\tau^{i+1}, \ldots, t_\tau^{i+n-1}$ estimated simultaneously using an $n$-bit CPA. The notation $\mathcal{V}_\tau^{i,n}(c) = \bigcap_{\nu=0}^{n-1} \mathcal{V}_\tau^{i+\nu}(c)$ signifies the number of available chosen-IV patterns for an $n$-bit CPA to estimate $\sigma_\tau^i, \sigma_\tau^{i+1}, \ldots, \sigma_\tau^{i+n-1}$. Utilizing Z3, the same SMT solver used in Section 3.2, we successfully found over 300,000 IVs for the CPA on $z_\tau^{i,8}(c)$ for both $c = 0$ and 1 listed in Table 3. This finding confirms that the proposed SCA is applicable to the 288-round unrolled hardware.

In the CPA, we used the eight-bit HD between the first round internal state $s_e^1(v), s_{e+1}^1(v), \ldots, s_{e+7}^1(v)$ and the target polynomials $t_\tau^i, t_\tau^{i+1}, \ldots, t_\tau^{i+7}$, where $e$ denotes the bit position of the register storing $t_\tau^i$ on the hardware. We calculated the hypothetical power consumption of $t_\tau^i, t_\tau^{i+1}, \ldots, t_\tau^{i+7}$ with an eight-bit guess of $z_\tau^{i,8}$ (considering $t_\tau^i = \sigma_\tau^i + g_\tau^i(v)$) for each IV.

[6]The source code will be publicly available upon acceptance.

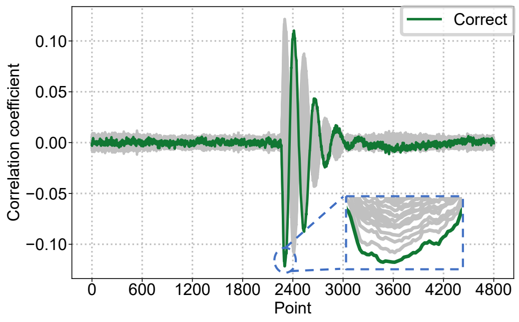


Figure 2: Sample-point-wise eight-bit CPA result on $z_1^{257}(0)$ with 250,000 traces.

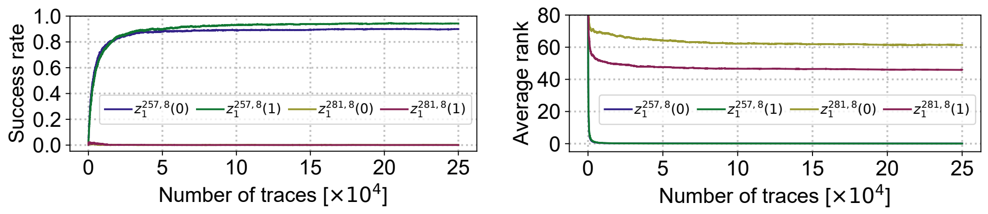
Subsequently, we computed the empirical Pearson's correlation coefficient between this hypothetical power consumption and observed side-channel traces, similarly to standard CPAs. Here, we guessed only eight bits out of the 288-bit internal state. Therefore, the CPA result would be influenced by the key values, as those not included in the estimate would have a non-trivial effect on the resulting side-channel trace. Therefore, we evaluated the CPAs using 30 different keys, to marginalize (or average) the influence of the key value.

## 4.2 Evaluation of eight-bit CPA

Figure 2 reports the sample-point-wise CPA result, exemplified through the estimation of $z_1^{257,8}(0)$ using 250,000 IVs in $\mathcal{V}_1^{257,8}(0)$. The horizontal axis represents the sample point index, and the vertical axis indicates the correlation coefficient value. The green and gray curves correspond to the correlation coefficient values for the correct and incorrect guesses, respectively. Notably, the correct guess yields the highest absolute correlation coefficient value around the 2200–2400 point range among all candidates and points, as highlighted. Therefore, we confirm that the eight-bit value of $z_1^{257,8}(0)$ can be correctly estimated by CPA. Note that we have the same results on $z_\tau^{i,8}(c)$ for different $\tau$, $i$, and $c$, given that a sufficient number of chosen-IV patterns are available.

Here, some incorrect guesses yielded in high correlation coefficient values due to the target polynomial $t_\tau^i$ (*i.e.*, selection function) being an *XOR* of the guessed value and input IV, differing from SCAs on block ciphers like AES. More precisely, the hypothetical power consumption guessed with wrong keys actually have correlation (although they are expected to be as high as that of correct key). The magnitude of correlation for wrong keys mainly depends on the selection function (rather than what the leakage includes information others than target intermediate value). Here, if the selection function includes non-linear function (*e.g.*, Sbox), the correlation would be weak. However, in the proposed SCA, the selection function is linear (as it is derived by RRLD), which results in a high correlation for wrong keys. See [Pro05, FDLZ15] for analyses on correlation for wrong keys. Furthermore, for given candidates $\hat{z}_\tau^{i,8}$, the correlation coefficient of another guess $\bar{z}_\tau^{i,8}$, with all bits inverted, equaled the sign-reversed correlation coefficient of $\hat{z}_\tau^{i,8}$ with an identical absolute value, due to the same reason. Consequently, the attacker should ascertain the polarity of the largest peak for the correct guess, based on the leakage characteristics of the device. Otherwise, the attacker cannot distinguish which candidate with a positive or negative peak is correct. In the following, we assume that the attacker is aware of such leakage characteristics[7].

[7]However, this assumption is not essential. If the attacker is unaware of the leakage characteristics, such an attacker can guess two patterns of leakage characteristics (*i.e.*, guess whether the polarity for the correct guess is negative or positive). Additionally, the success of the linearization attack does *not* rely on



(a) Success rate

(b) Average rank

Figure 3: Result of eight-bit CPAs on $z_1^{257,8}(c)$ and $z_1^{288,1}(c)$.

We then evaluated the CPA success rate and complexity. Figure 3 reports the (a) success rate and (b) average rank[8] of CPA for estimating $z_1^{257,8}$. For each key value, we acquired 700,000 traces, computed its CPA success rate using 100 trials using the *bootstrap* method for varying number of traces, and finally averaged the success rate across 30 key patterns. To determine the key rank in the CPA, we used the average of correlation coefficients for 10 sample points exhibiting high negative values, as shown in Figure 2. From Figure 3(a), we confirm that the averaged success rate eventually approximates to 0.9. Specifically, the proposed SCA achieved significant success for 28 out of the 30 keys, whereas little success was observed for the remaining 2 keys. Moreover, as observed in Figure 3(b), the average ranks of both $z_1^{257,8}(0)$ and $z_1^{257,8}(1)$ were very close to zero. Remarkably, the aforementioned two keys had a rank of one in this experiment. In other words, we can reduce the number of key candidates to one or two. Consequently, the CPA results suggest that the value of eight linear relations $k_{51} + k_{52}, k_{50} + k_{51}, \ldots, k_{44} + k_{45}$ corresponding to $z_1^{257,8}(0)$ and $z_1^{257,8}(1)$ can take at most $2^2$ patterns. The result on $z_1^{257,8}(c)$ is almost the same for other bytes listed in Table 3, as a sufficient number of chosen-IV patterns are available for them. Therefore, the total number of candidate values of the linear relations can be reduced to $(2^2)^9 = 2^{18}$ patterns, as the attacker should perform the CPA $2 \times 9$ times. As 11 bits of the secret key are not included in the linear relations in Table 3, the proposed SCA can reduce the key space from $2^{80}$ to $2^{18} \times 2^{11} = 2^{29}$, which is sufficiently small to perform an exhaustive search. Additionally, we achieve a successful estimation of $z_\tau^{i,8}(c)$ with approximately 20,000 traces for each CPA and need 18 $(= 2 \times 9)$ CPAs. This result indicates that the total number of traces for key recovery is at most 360,000, which is also sufficiently practical. Thus, we confirm that the proposed SCA can recover the secret key of the 288-round unrolled Trivium hardware implementation.

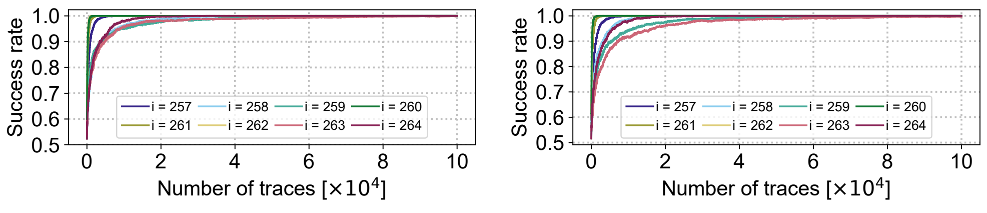
## 4.3 Effect of the number of available chosen-IV patterns on CPA

We then evaluated the impact of the number of available chosen-IV patterns on the success rate. For the 288-round unrolled Trivium, the number of chosen-IV patterns was limited to 84 and 72 for $z_1^{281,8}(0)$ and $z_1^{281,8}(1)$, respectively (*i.e.*, $|\mathcal{V}_1^{281,8}(0)| = 84$ and $|\mathcal{V}_1^{281,8}(1)| = 72$), because the internal state is more diffused for larger $i$. Note that $z_1^{281,8}(c)$ is not included in Table 3, while $z_2^{280,8}(c)$ is. Typically, $t_1$ would be more diffused than $t_2$ and $t_3$ even at the same round. We evaluated the CPA on $z_1^{281,8}(0)$ and $z_1^{281,8}(1)$ using the aforementioned 84 and 72 chosen-IV patterns, respectively.

Figure 3 also presents the success rate and average rank of CPA on $z_1^{281,8}(c)$, averaged across 15 different keys. Here, IVs were randomly sampled from $\mathcal{V}_1^{281,8}(c)$ repeatedly, which means that each chosen IV in $\mathcal{V}_1^{281,8}(c)$ is used multiple times. We found no successful attack and confirmed that the average ranks were significantly worse than those on $z_1^{257,8}(c)$, even when using 250,000 traces, maybe because of the algorithmic noise. Thus, we confirm

the success of this guess, as $\sigma^i(0) + \sigma^i(1) = (\sigma^i(0) + 1) + (\sigma^i(1) + 1)$.

[8]Commonly referred to the *guessing entropy*.



(a) $c = 0$ (b) $c = 1$

Figure 4: Success rate of one-bit CPA to estimate $z_1^{i,1}(c)$ for $i = 257, 258, \ldots, 264$.

Table 4: Number of traces to achieve the success rate of 1.0 for one-bit CPA on $z_1^{i,1}(c)$

| $i$ | 257 | 258 | 259 | 260 | 261 | 262 | 263 | 264 |
|---|---|---|---|---|---|---|---|---|
| $c = 0$ | 9,201 | 92,240 | 220,845 | 2,089 | 3,315 | 1,977 | 157,371 | 51,976 |
| $c = 1$ | 18,518 | 43,833 | 204,575 | 2,438 | 5,429 | 1,714 | 225,725 | 53,185 |

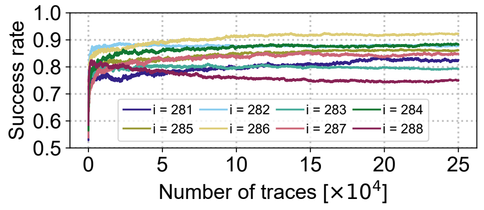
the importance of the number of chosen-IV patterns, which gets smaller for larger $i$, because we should have more restriction to the IV due to the diffusion.

## 4.4 Evaluation of one-bit CPA

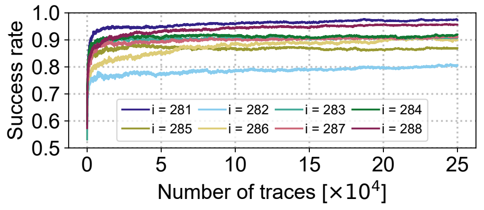
Section 4.3 demonstrated that an eight-bit CPA on $z_1^{281,8}(c)$ was unsuccessful owing to an insufficient number of chosen-IV patterns. For an eight-bit CPA, the restriction to the IV set is defined by $\mathcal{V}_1^{281,8}$, as we should simultaneously restrict $g_1^{281}, g_1^{282}, \ldots, g_1^{288}$. However, for a one-bit CPA, the IV restriction is determined by $\mathcal{V}_1^{i,1} = \mathcal{V}_1^{i}$, the size of which is always greater than (or equal to) $|\mathcal{V}_1^{281,8}|$ (*i.e.*, $|\mathcal{V}_1^{i,1}| \geq |\mathcal{V}_1^{i,8}|$ for any $i$). Indeed, we found a sufficient number of chosen-IV patterns for a one-bit CPA on $z_1^{i,1}$ for each $i = 281, 282, \ldots, 288$ (where $z_1^{i,1} = \sigma_1^i$); specifically, we confirmed that $|\mathcal{V}_1^{i,1}|$ is greater than 10,000. Refer to Section 5 for detailed information. Thus, a one-bit CPA on $z_1^{i,1}$ for each $i = 281, 282, \ldots, 288$ is likely to be successful owing to the larger size of $\mathcal{V}_1^{i,1}$.

Then, we experimentally evaluated the one-bit CPA on $z_1^{i,1}$. First, to assess the case with a sufficient number of chosen-IV patterns, we performed one-bit CPA on $z_1^{i,1}$ for $i = 257, 258, \ldots, 264$. Figure 4 displays the success rates of these one-bit CPAs, averaged across 30 different keys. We can confirm that the empirical success rates for each bit eventually reached one with the use of 250,000 traces. However, the stability of the CPA result was worse than that observed in the eight-bit CPA. Table 4 lists the number of traces required to achieve an empirical success rate of one for each $i$ and $c$. We required more than 100,000 traces for recovering $z_1^{259,1}$ and $z_1^{263,1}$, whereas less than 10,000 traces sufficed for CPAs on $z_1^{260,1}$, $z_1^{261,1}$, and $z_1^{262,1}$. This variation would be likely because of the increased sensitivity of one-bit CPA to noise, as the signal component is confined to a single bit. Nonetheless, we confirm the feasibility of a one-bit CPA, given that a sufficient number of IV patterns are available.

We finally evaluate one-bit CPA with a restricted number of chosen-IV patterns. We executed one-bit CPA on $z_1^{i,1}(c)$ for $c = 0$ and $c = 1$ using 84 and 72 chosen-IV patterns (which are the same ones as those in Section 4.3), respectively. Figure 5 presents the empirical success rates of these CPAs, averaged across 15 different keys. Although the success rates were non-trivial, they did not reach one in our experiment and were insufficient for reliable key recovery. Thus, we confirm that the number of available chosen-IV patterns has a significant impact on the success rate for both one-bit and multi-bit CPAs.

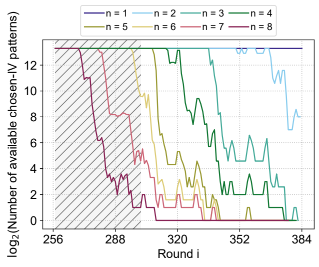


(a) $c = 0$

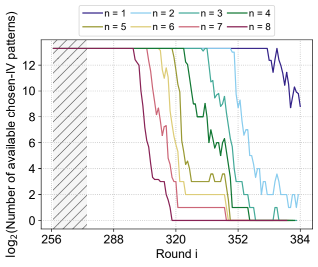


(b) $c = 1$

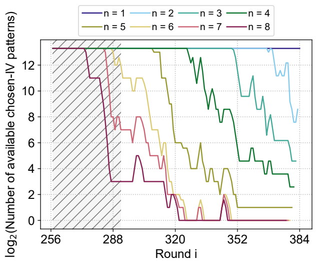
Figure 5: Success rate of one-bit CPA to estimate $z_1^{i,1}(c)$ with limited chosen-IV patterns.



(a) $\tau = 1$



(b) $\tau = 2$



(c) $\tau = 3$

Figure 6: Size of $\mathcal{V}_\tau^{i,n}(0)$ for $n$-bit CPAs on $r$-round unrolled implementation with $r = 384$.

## 5 Discussion

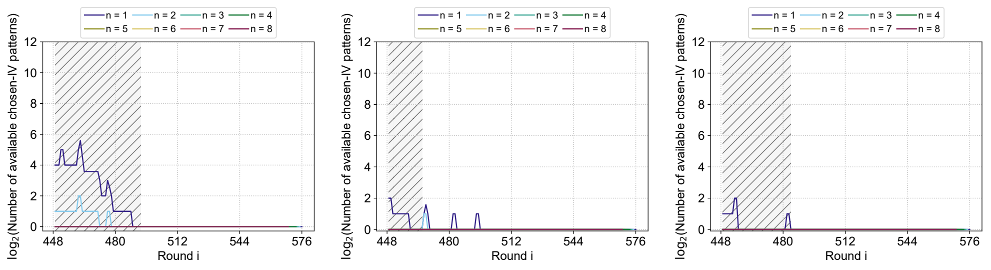
### 5.1 SCA (in)feasibility evaluation about larger unrolling degrees

We here consider unrolling degrees of 384 and 576, divisors of 1152 exceeding 288.

In an $n$-bit CPA for the $i$-th round, the subsequent $n$ rounds beginning with the $i$-th round are the target. For example, for $i = 320$ and $n = 6$, the target polynomials are $t_\tau^{320}(0), t_\tau^{321}(0), \dots, t_\tau^{325}(0)$, with the key polynomial being $z_\tau^{320,6}$, and the chosen-IV set is $\mathcal{V}_\tau^{320,6}$. Figure 6 and Figure 7 illustrate the size of $\mathcal{V}_\tau^{i,n}(0)$ for one- to eight-bit CPAs on $r$-round unrolled Trivium hardware (*i.e.*, $i = 274$ to 384 and $i = 466$ to 576, respectively). These values were derived using Algorithm 2 and the SMT solver Z3. We terminated the IV search upon finding 10,000 chosen IVs; therefore, the maximum value is 10,000. Note that the results for $\mathcal{V}_\tau^{i,n}(1)$ are nearly identical. The hatched rounds in these figures cannot be targeted by SCA for $r = 384$ and 576, as they are not stored in registers (these values provided merely for reference).

We confirm that the number of available chosen-IV patterns decreases with an increase in $n$ for an $n$-bit CPA. Specifically, for $r = 384$ depicted in Figure 6, no variety of available chosen-IV patterns was found for $n = 8$ when $i$ exceeded 309, and the number of available chosen-IV patterns was deemed insufficient for $n \geq 2$. Eight-bit CPA is feasible only for $274 \leq i \leq 309$ and $\tau$, which is insufficient for key recovery. Conversely, we found a sufficient number of chosen IVs for $n = 1$, suggesting the feasibility of one-bit CPA on 384-round unrolled hardware for full key recovery. However, a one-bit CPA attacker should obtain traces to estimate $z_\tau^{i,1}$ with IVs from $\mathcal{V}_\tau^{i,1}$ for each $i$, which results in $n$ times more traces than an $n$-bit CPA. Moreover, as observed in Section 4.4, one-bit CPA requires a greater number of traces for reliable estimation of $z_\tau^{i+\nu,1}$ for each $\nu$ compared to an eight-bit CPA. Regarding these facts, we can quantitatively evaluate its SCA resistance in terms of the total number of traces for full key recovery.

For $r = 576$, a considerably limited number of chosen-IV patterns were available. Even for one-bit CPA, we found at most $2^2 = 4$ IV patterns, which is too few for CPA to estimate



(a) $\tau = 1$ (b) $\tau = 2$ (c) $\tau = 3$

Figure 7: Size of $\mathcal{V}_{\tau}^{i,n}(0)$ for $n$-bit CPAs on $r$-round unrolled implementation with $r = 576$.

Table 5: Performance evaluation result of unrolled Trivium hardware on Nangate 45 nm Open Cell Library (Power is evaluated at 100 MHz)

| $r$ | Critical delay | Initialization latency | | Throughput | Area | Power | Energy | SCA security |
|---|---|---|---|---|---|---|---|---|
| | [ns] | [Clock cycles] | [ns] | [Gbits/s] | [GE] | [mW] | [fJ/bit] | [# traces] |
| 288 | 1.04 | 4 | 4.16 | 277 | 11,358 | 2.58 | 89.58 | 360,000 |
| 384 | 1.26 | 3 | 3.78 | 305 | 14,428 | 3.91 | 101.82 | 34,500,000 |
| 576 | 1.86 | 2 | 3.72 | 310 | 19,391 | 7.49 | 130.03 | N/A |
| 1152 | 3.16 | 1 | 3.16 | 364 | 38,608 | 24.0 | 208.33 | N/A |

$\sigma_{\tau}^{i}$. Therefore, we conclude that 576-round unrolled Trivium hardware is resistant to the proposed SCA, and the round unrolling would be useful for achieving leakage resilience.

## 5.2 Tradeoff between energy consumption and SCA security

We evaluate the performance and SCA security of unrolled Trivium hardware using Nangate 45 nm Open Cell Library and the Synopsys Design Compiler. Table 5 provides the logic synthesis results, power/energy consumption estimations, and SCA security assessments, where Power is estimated at 100 MHz, and SCA security is defined as the number of traces required for full key recovery by the proposed SCA. "N/A" indicates that the proposed SCA is not applicable, which implies an unbounded security against the proposed SCA. To determine the number of traces for $r = 384$, we assumed that one-bit CPA requires 250,000 traces to reliably estimate $z_{\tau}^{i,1}(c)$ for each $i$ and $c$, as per the experimental results in Table 4. In total, $69 \times 2$ CPAs are required to collect key linear relations for key recovery.

We confirm that the larger round unrolling contributes to the high throughput and low latency at a cost of area and power consumption. The energy consumption escalates with an increase in the unrolling degree for $r \geq 288$. Therefore, as demonstrated in [CBT+21], the 288-round unrolled hardware is optimal in terms of energy. Concurrently, the number of traces for key recovery (*i.e.*, SCA security) increases significantly with $r$. The key recovery from a 384-round unrolled implementation would require 34.5 M traces, which is almost 100 times greater than that of the 288-round unrolled variant. Therefore, key lifetime against SCA substantially improves by the round unrolling, contributing to the reduction of rekeying cost [UHIM23]. Moreover, the proposed SCA is inapplicable to 576- and 1152-round unrolled implementations, which indicates their unbounded security against the proposed SCA. Thus, we confirm that round unrolling in stream ciphers thus would be promising to achieve SCA security in addition to high throughput, low energy with practical implementation costs.

## 5.3 Relation to reduced-round cryptanalyses

SCA on stream cipher focusing on the initialization would be closely related to the reduced-round cryptanalaysis. SCAs focus on some intermediate bits to estimate the secret key from the leakage of these bits, while reduced-round cryptanalyses focus on keystream bits. Therefore, they focus on different target bits. Besides, there is a fundamental difference.

**Cube attacks.** Most reduced-round cryptanalyses on Trivium are based on the cube attack [DS09]. The first cube attack [DS09], which requires data and time complexities of $2^{18.6}$ and $2^{17}$ for the key recovery of 672-round Trivium, respectively, is most efficient in terms of these complexities. A state-of-the-art cube attack in [HHLW24] achieved the key recovery of 851-round Trivium with data and time complexities of $2^{44}$ and $2^{79}$, respectively. The cube attack (and its variant) first takes all possible values for specific bits in IV and gets these keystreams (without any noise). Then, it exploits the sum of these keystreams can be represented by a polynomial with a low-degree key monomial. Since the attacker can get each keystream without noise, he/she can compute the sum over massive data.

In contrast, the proposed SCA utilizes the key polynomials $\sigma^i$ to be estimated by CPA from practical leakages *with noise*. In the context of the cube attack, this indicates that the attacker observes keystreams with noise. Without eliminating the noise, the attacker cannot recover the keystream and compute the sum. Our proposal in Section 3.2 focuses on how to estimate $\sigma^i$ from practical leakages, different from the cube attack. Contrarily, once a sufficient number of $\sigma^i$'s are estimated, the linearization for feasible key recovery shares the same observation of the cube attack. Namely, when $\sigma^i$ contains high-degree key monomials, we decrease the degree of key monomials by exploiting the sum of $\sigma^i$.

Although the sum of $\sigma^i$ is a low-degree polynomial, note that we cannot estimate the sum directly by the SCA because the sum never appears on the encryption device. Therefore, the feasibility of the proposed SCA depends on whether $\sigma^i$ can be estimated or not. Section 5.1 actually discussed the infeasibility evaluation about large unrolling. Consequently, the proposed SCA strategy addressed a problem different from reduced-round cryptanalyses to feasibily estimate $\sigma^i$ via power/EM side-channel, while the reduced-round cryptanalyses assume that the output bits are directly available for the attacker.

**Linear cryptanalysis.** Our attack can be regarded as linear cryptanalysis, where the outputs of any nonlinear AND gates are conditionally computed, instead of approximation. Importantly, our attack approximates no AND gates. If we approximate $x$ AND gates like standard linear cryptanalysis, the correlation is $2^{-x}$ and $2^{2x}$ samples are roughly required even if we observe noise-free states, which implies the number of traces for SCA is $2^{2x}$-times. For example, an AND gate approximation leads to 4-times more traces. If several gates are approximated, SCA immediately gets infeasible. Hence, our attack is, now, regarded as chosen-IV linear cryptanalysis with correlation 1. IVs are chosen to control output of AND gates to be constant (rather than approximation). Note that our attack would be trivial in the blackbox analysis context, where the main focuses are to show contradiction to the 80-bit security and to detect better approximation whose correlation is higher than $2^{-40}$. So far, our analysis has not been deeply done in some non-trivial contexts.

**Conditional differential.** Our linearization technique would be roughly similar conditional differential. We focus on an intermediate bit and exploit difference between two chosen-IVs to linearize the key polynomial. However, noise of side-channel measurement makes non-trivial difference. If noise-free, we can compute exact difference. Otherwise, the noise of difference is amplified. In the blackbox analysis context, when noise is $2^{-x}$ and $2^{-y}$ for each bit, a well-known piling-up lemma states the noise of difference is $2^{-xy}$. Then, we need $2^{2xy}$-times traces. In blackbox analysis, we can explore the conditional differential probability, while the amplified noise in SCA renders it difficult.

Table 6: Applicability of proposed attack to major stream ciphers

| Type | w/o arithmetic operations | w/ arithmetic operations |
|---|---|---|
| Examples | Trivium, Enocoro, Trivium-LE, Kreyvium, Lizard, Grain | ChaCha20, Salsa, RC4, KCipher-2 |
| Applicability | Maybe | Unclear |

## 6 Conclusion

We presented a side-channel linearization attack to evaluate the SCA security of unrolled implementations of stream ciphers, with a focus on Trivium. Given the unavailability of side-channel leakage of the first rounds from unrolled hardware, we proposed RRLD with linearization, estimating the internal state bits of latter rounds, and recovering the full secret key with feasible complexity. Based on our experimental results obtained from an actual unrolled hardware implementation, we reported that the key recovery on the 288-round unrolled hardware implementation, which was recognized as the most energy-efficient unrolling degree, is sufficiently feasible with the proposed SCA. Furthermore, we evaluated the SCA (in)feasibility for 384- and 576-round unrolled Trivium hardware to analyze its leakage resilience. The 576-round unrolled hardware implementation is found to be unboundedly secure against the proposed SCA. Additionally, the 384-round unrolled variant is deemed more secure than its 288-round unrolled counterpart in terms of the number of traces for key recovery.

The proposed SCA and our discussions would be potentially applicable to other stream ciphers based on bit-wise (N)LFSR(s), such as Trivium-LE [CBT+21], while feasibility/applicability to other types of stream ciphers (*e.g.*, ChaCha20 and KCipher-2) is unclear, as listed in Table 6. This is because we can apply the RRLD strategy if we derive the logic representation of intermediate bits, by expanding the round function, and it would be too complicated to express intermediate bits of unrolled arithmetic-oriented streams ciphers. However, note that unrolled hardware implementation of such arithmetic-oriented stream cipher is not common due to the large logic depth and area of arithmetic adders, and our strategy would offer applicability to most stream ciphers suitable to unrolled implementation. The application and evaluation of the proposed SCA on these ciphers remain as important future work.

Additionally, SCAs using other distinguishers (*e.g.*, deep-learning based SCA) would be available to estimate the intermediate values more efficiently than CPA, which would contribute to a reduction of the number of traces in key recovery and may be able to exploit the leakage from wires. However, the usefulness and applicability of such SCAs to stream ciphers and/or exploitation of wire values are yet to be studied, because there have been no previous studies on these aspects (as far as we know). For example, the condition that a profiling DL-based SCA successfully works/fails on block cipher has been widely studied (*e.g.*, [PHJ+19, ISUH21, IUH21]), the condition is unknown for stream ciphers (and actually some conditions may not be satisfied in SCA on stream cipher such as KIC in [IUH21]). These SCAs are to be studied for more severe evaluation of stream cipher implementations in future.

## References

[BGSD10] Shivam Bhasin, Sylvain Guilley, Laurent Sauvage, and Jean-Luc Danger. Unrolling cryptographic circuits: A simple countermeasure against side-channel attacks. In *Topics in Cryptology—CT-RSA 2010*, pages 195–207, 2010. doi:10.1007/978-3-642-11925-5_14.

[BMA+18] Subhadeep Banik, Vasily Mikhalev, Frederik Armknecht, Takanori Isobe, Willi Meier, Andrey Bogdanov, Yuhei Watanabe, and Francesco Regazzoni. Towards low energy stream ciphers. *IACR Transactions on Symmetric Cryptology*, (2):1–19, 2018. doi:10.13154/tosc.v2018.i2.1-19.

[BZD+16] Hanno Böck, Aaron Zauner, Sean Devlin, Juraj Somorovsky, and Philipp Jovanovic. Nonce-disrespecting adversaries: practical forgery attacks on GCM in TLS. In *Proceedings of the 10th USENIX Conference on Offensive Technologies*, WOOT'16, pages 15–25, 2016. URL: https://www.usenix.org/conference/woot16/workshop-program/presentation/bock.

[Can06] Christophe De Canniere. Trivium specifications, 2006. http://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf.

[CBT+21] Andrea Caforio, Subhadeep Banik, Yosuke Todo, Willi Meier, Takanori Isobe, Fukang Liu, and Bin Zhang. Perfect trees: Designing energy-optimal symmetric encryption primitives. *IACR Transactions on Symmetric Cryptology*, (4):36–73, 2021. doi:10.46586/tosc.v2021.i4.36-73.

[CMM+23] Gaëtan Cassiers, Loïc Masure, Charles Momin, Thorben Moos, Amir Moradi, and François-Xavier Standaert. Randomness generation for secure hardware masking - unrolled Trivium to the rescue. Cryptology ePrint Archive, Paper 2023/1134, 2023. URL: https://eprint.iacr.org/2023/1134.

[DCP08] Christophe De Canniere and Bart Preneel. Trivium. *New Stream Cipher Designs: The eSTREAM Finalists*, pages 244–266, 2008.

[dMB08] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, 2008.

[DS09] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In *Advances in Cryptology—EUROCRYPT 2009*, pages 278–299, 2009. doi:10.1007/978-3-642-01001-9_16.

[eST] The eSTREAM portfolio—eSTREAM: the ECRYPT stream cipher project.

[FDLZ15] Yunsi Fei, A. Adam Ding, Jian Lao, and Liwei Zhang. A statistics-based success rate model for DPA and CPA. *Journal of Cryptographic Engineering*, 5, 2015. doi:10.1007/s13389-015-0107-0.

[FGKV06] Wieland Fischer, Berndt M Gammel, Oliver Kniffler, and Joachim Velten. Differential power analysis of stream ciphers. In *Topics in Cryptology–CT-RSA 2007: The Cryptographers' Track at the RSA Conference 2007, San Francisco, CA, USA, February 5-9, 2007. Proceedings*, pages 257–270. Springer, 2006. doi:10.1007/11967668_17.

[FPS12] Sebastian Faust, Krzysztof Pietrzak, and Joachim Schipper. Practical leakage-resilient symmetric cryptography. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2012)*, volume 7428 of *Lecture Notes in Computer Science*, pages 213–232. Springer, 2012. doi:10.1007/978-3-642-33027-8_13.

[HHLW24] Jiahui He, Kai Hu, Hao Lei, and Meiqin Wang. Massive superpoly recovery with a meet-in-the-middle framework: Improved cube attacks on trivium and kreyvium. In *Advances in Cryptology—EUROCRYPT 2024*, 2024. doi:10.1007/978-3-031-58716-0_13.

[HHN+13] Takafumi Hibiki, Naofumi Homma, Yuto Nakano, Kazuhide Fukushima, Shinsaku Kiyomoto, Yuta Miyake, and Takafumi Aoki. Chosen-IV correlation power analysis on KCipher-2 and a countermeasure. In *International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE 2013)*, volume 7864 of *Lecture Notes in Computer Science*, pages 169–183, 2013. doi:10.1007/978-3-642-40026-1_11.

[iso] ISO/IEC 29192-3:2012 Information technology—Security techniques—Lightweight cryptography— Part 3: Stream ciphers. https://www.iso.org/standard/56426.html.

[ISUH21] Akira Ito, Kotaro Saito, Rei Ueno, and Naofumi Homma. Imbalanced data problems in deep learning-based side-channel attacks: Analysis and solution. *IEEE Transactions on Information Forensics and Security*, 16:3790–3802, 2021. doi:10.1109/TIFS.2021.3092050.

[IUH21] Akira Ito, Rei Ueno, and Naofumi Homma. Toward optimal deep-learning based side-channel attacks: Probability concentration inequality loss and its usage. Cryptology ePrint Archive, Report 2021/1216, 2021. https://ia.cr/2021/1216.

[JHWW12] Yanyan Jia, Yupu Hu, Fenghe Wang, and Hongxian Wang. Correlation power analysis of Trivium. *Security and Communication Networks*, 5(5):479–484, 2012. doi:10.1002/sec.329.

[KDB+22] Satyam Kumar, Vishnu Asutosh Dasu, Anubhab Baksi, Santanu Sarkar, Dirmanto Jap, Jakub Breier, and Shivam Bhasin. Side channel attack on stream ciphers: A three-step approach to state/key recovery. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 166–191, 2022. doi:10.46586/tches.v2022.i2.166-191.

[KUH+17] Wataru Kawai, Rei Ueno, Naofumi Homma, Takafumi Aoki, Kazuhide Fukushima, and Shinsaku Kiyomoto. Practical power analysis on KCipher-2 software on low-end microcontrollers. In *Workshop on Security for Embedded and Mobile System, IEEE European Symposium on Security and Privacy Workshops (SEMS, EuroSPW 2017)*, pages 113–121, 2017. doi:http://dx.doi.org/10.1109/EuroSPW.2017.60.

[MHBM+18] Maxime Montoya, Thomas Hiscock, Simone Bacles-Min, Anca Molnos, and Jacques J.A. Fournier. Energy-efficient masking of the Trivium stream cipher. In *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 393–396, 2018. doi:10.1109/ICECS.2018.8617892.

[Mic23] Microsoft. Z3 API in Python. https://www.microsoft.com/en-us/research/project/z3-3/, 2023.

[Moo20] Thorben Moos. Unrolled cryptography on silicon: A physical security analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(4):416–442, 2020. doi:10.13154/tches.v2020.i4.416-442.

[MPO05] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked AES hardware implementations. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, pages 157–171, 2005. doi:10.1007/11545262_12.

[MS16] Amir Moradi and Tobias Schneider. Side-channel analysis protection and low-latency in action. In *Advances in Cryptology—ASIACRYPT 2016*, pages 517–547, 2016. doi:10.1007/978-3-662-53887-6_19.

[PHJ+19] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, (1):209–237, 2019. doi:10.13154/tches.v2019.i1.209-237.

[Pro05] Emmanuel Prouff. DPA attacks and S-boxes. In *Fast Software Encryption*, pages 424–441, 2005. doi:10.1007/11502760_29.

[SA15] Dillibabu Shanmugam and Suganya Annadurai. Secure implementation of stream cipher: Trivium. In *Innovative Security Solutions for Information Technology and Communications*, pages 253–266, 2015. doi:10.1007/978-3-319-27179-8_18.

[SJB21] Siang Meng Sim, Dirmanto Jap, and Shivam Bhasin. DAPA: Differential analysis aided power attack on (non-)linear feedback shift registers. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 169–191, 2021. doi:10.46586/tches.v2021.i1.169-191.

[SPK09] Daehyun Strobel, Ing Christof Paar, and M Kasper. Side channel analysis attacks on stream ciphers. *Masterarbeit Ruhr-Universität Bochum, Lehrstuhl Embedded Security*, 2009.

[TSA15a] Erica Tena-Sánchez and Antonio J Acosta. DPA vulnerability analysis on trivium stream cipher using an optimized power model. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1846–1849. IEEE, 2015. doi:10.1109/ISCAS.2015.7169016.

[TSA15b] Erica Tena-Sánchez and Antonio J Acosta. Optimized DPA attack on trivium stream cipher using correlation shape distinguishers. In *2015 Conference on Design of Circuits and Integrated Systems (DCIS)*, pages 1–6. IEEE, 2015. doi:http://dx.doi.org/10.1109/DCIS.2015.7388578.

[UHIM23] Rei Ueno, Naofumi Homma, Akiko Inoue, and Kazuhiko Minematsu. Fallen sanctuary: A higher-order and leakage-resilient rekeying scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 1(2024):264–308, 2023. doi:10.46586/tches.v2024.i1.264-308.

[YMHA16] Ville Yli-Mäyry, Naofumi Homma, and Takafumi Aoki. Improved power analysis on unrolled architecture and its application to PRINCE block cipher. In *Lightweight Cryptography for Security and Privacy*, pages 148–163, 2016. doi:10.1007/978-3-319-29078-2_9.

[YMUM+21] Ville Yli-Mäyry, Rei Ueno, Noriyuki Miura, Makoto Nagata, Shivam Bhasin, Yves Mathieu, Tarik Graba, Jean-Luc Danger, and Naofumi Homma. Diffusional side-channel leakage from unrolled lightweight block ciphers: A case study of power analysis on prince. *IEEE Transactions on Information Forensics and Security*, 16:1351–1364, 2021. doi:10.1109/TIFS.2020.3033441.