Fallen Sanctuary: A Higher-Order and Leakage-Resilient Rekeying Scheme

Rei Ueno[1], Naofumi Homma[1], Akiko Inoue[2] and Kazuhiko Minematsu[2]

[1] Tohoku University, 2–1–1 Katahira, Aoba-ku, Sendai-shi, Miyagi, 980-8577, Japan
rei.ueno.a8@tohoku.ac.jp, naofumi.homma.c8@tohoku.ac.jp
[2] NEC Secure System Platform Laboratories, 1753 Shimonumabe, Nakahara, Kawasaki, Kanagawa 211–8666, Japan
a_inoue@nec.com, k-minematsu@nec.com

Abstract. This paper presents a provably secure, higher-order, and leakage-resilient (LR) rekeying scheme named LR Rekeying with Random oracle Repetition (LR4), along with a quantitative security evaluation methodology. Many existing LR primitives are based on a concept of leveled implementation, which still essentially require a *leak-free sanctuary* (*i.e.*, differential power analysis (DPA)-resistant component(s)) for some parts. In addition, although several LR pseudorandom functions (PRFs) based on only bounded DPA-resistant components have been developed, their validity and effectiveness for rekeying usage still need to be determined. In contrast, LR4 is formally proven under a leakage model that captures the practical goal of side-channel attack (SCA) protection (*e.g.*, masking with a practical order) and assumes no unbounded DPA-resistant sanctuary. This proof suggests that LR4 resists exponential invocations (up to the birthday bound of key size) without using any unbounded leak-free component, which is the first of its kind. Moreover, we present a quantitative SCA success rate evaluation methodology for LR4 that combines the bounded leakage models for LR cryptography and a state-of-the-art information-theoretical SCA evaluation method. We validate its soundness and effectiveness as a DPA countermeasure through a numerical evaluation; that is, the number of secure calls of a symmetric primitive increases exponentially by increasing a security parameter under practical conditions.

Keywords: Fresh rekeying · Leakage resilience · Side-channel attack

1 Introduction

1.1 Background

Side-channel attacks (SCAs) are physical attacks on cryptographic implementations [KJJ99]. SCA countermeasures are roughly categorized into three types: *masking*, *hiding*, and *leakage-resilient (LR) cryptography*. Both masking and hiding are basically designed to suppress/eliminate the leakage for a given algorithm/device. However, it has been shown experimentally [BS21] and theoretically [DFS15, IUH22a, MRS22] that it might be difficult for masking to achieve secure implementation on some low-end devices with trivial noise. Hiding (*e.g.*, a secure logic style like WDDL [TV04]) is also sometimes unsuitable because its effectiveness depends heavily on the given device/technology. In contrast, LR cryptography features cryptographic algorithms capable of secure computation up to a specific, predetermined level of leakage. For developing practically secure cryptographic modules, it is essential to investigate the possibility and limitations of LR cryptography as well as masking and hiding.

This paper focuses on LR symmetric key cryptography[1]. Rekeying (or fresh rekeying) is the pioneering and primordial concept of leakage resilience [KJJ99, Koc98, MOP07, MSGR10, GFM13, BDSH+14] that updates the secret key using a deterministic function with a master key before the attacker obtains a sufficient amount of leakages for the key recovery. Then, LR pseudorandom generators (PRGs) and stream ciphers have been studied. They take a secret key K and possibly an initializing vector IV and outputs a long output S that are computationally hard to be distinguished from random as in the case of conventional PRG/stream cipher. Moreover, LR PRGs/stream ciphers are designed in such a way as to make distinguishing attack S from random infeasible even if there exists some leakage of each internal state [DP08, Pie09, YSPY10, DP10]. That is, $S = \text{LR-PRG}(K, \text{IV})$ is pseudorandom even with a partial information on K that is leaked. In 2012, two LR pseudorandom functions (PRFs) [FPS12, MSJ12] based on the classical Goldreich–Goldwasser–Micali (GGM) PRF [GGM86] were proposed. GGM PRF is the classical method to build a PRF based on a PRG. Using a PRG $f : \{0,1\}^{n_f} \to \{0,1\}^{2n_f}$, GGM PRF forms a binary tree where each node corresponds to the input bit and each path has a binary label. If label is 0 (1), it means the child node uses the first (last) n_f bits of the PRG output. Let $f(K) = (f_0(K), f_1(K))$ where $|f_i(K)| = n_f$. Then, for example, 2-bit-input GGM PRF based on f is a depth-2 tree and it outputs $f_1(f_0(K))$ for input (01), using the key K (See also Appendix A). As PRF is more functional than PRG, LR PRFs expand the application area of LR cryptography, say it can be used to build a LR message authentication code. More recently, LR authenticated encryption (AE) has become an active research topic [PSV15, BGP+19, DJS19, KS20, DEM+20, BBB+20, BGP+19, SPS+22]. Here, a secure AE scheme simultaneously implements encryption and message authentication, meaning that the plaintext is computationally hard to guess from the ciphertext (*i.e.*, confidentiality) and a forgery against the ciphertext is also computationally hard (*i.e.*, authenticity) [BN00]. AE has been extensively studied, and regarding the black-box security (no SCAs), there are number of efficient schemes with provable security guarantee, namely the whole AE security is reducible to the primitive's security. However, most of such provable-secure schemes, such as GCM or OCB, totally lose the security if SCAs are mounted and the underlying primitive is vulnerable to SCAs. Applying full protection, say by using fully-protected AES for every call of AES–GCM, is usually quite costly. The goal of LR-AE is to realize a secure AE even with SCAs, in a more efficient way than the aforementioned naive combination of a black-box-secure AE and a fully protected primitive. Typical LR-AEs consist of *leak-free* (*i.e.*, DPA-resistant) component(s) only for initialization/finalization and SPA-resistant component(s) for associated data (AD) and plaintext/ciphertext processing, which enables them to achieve more efficient SCA-resistant AE implementation than non-LR-AEs with DPA-resistant components for whole computation (*e.g.*, OCB) [BGP+19].

On the one hand, most rekeying schemes, LR-PRGs/stream ciphers, and LR-AEs are based on the concept of *leveled implementation* [PSV15], which effectively combines DPA- and SPA-resistant components. Such LR cryptographic schemes (implicitly) assume the availability of leak-free (*i.e.*, DPA-resistant) component(s). Rekeying schemes and LR-AEs explicitly require such a leak-free component for their rekeying function and initialization/finalization, respectively. LR-PRG/stream cipher is designed to be secure even if there exists some leakage of the internal state during random bits/key stream generation; however, the initial state updates must be protected against DPAs, as some practical DPAs defeating PRG/stream ciphers have been reported, such as in [SPY+10, HHN+13, JB17, KUH+17] (although these attacks might be outside the scope of LR-PRG/stream ciphers). In practice, it is difficult to achieve a *completely* leak-free component (*i.e.*, resistance against DPA with an *infinite* number of traces, or *unbounded* DPA-resistance) even with higher-order masking. In other words, masking schemes cannot achieve unbounded security

[1]LR public key cryptography is also an active research topic (*e.g.*, [NS09, ADW09, KV09, DHLAW10, BSW12, ASB14, CMY+16]).

against higher-order attackers, while the number of traces for key recovery increases exponentially by the masking order [DFS15, IUH22a, MRS22].

On the other hand, the security of LR-PRFs has been discussed using a bounded data or trace complexity model, which means that the number of plaintexts/traces available in attacking the PRF with a secret key is bounded. Here, secure LR-PRF implementation requires a component resistant to DPA with m plaintexts or traces (*i.e.*, m-bounded DPA-resistance). This is a more relaxed and practical condition than unbounded DPA resistance. Nonetheless, the validity of the bounded complexity model needs to be clarified in practice. At least, the model is valid only if it works with a dedicated protocol, but there has been little discussion about its practical use. In addition, it is not trivial how to determine m for a given device about SCA success rate on the LR-PRF. See Appendix A for the details of existing LR-PRFs.

In summary, the existing LR cryptography schemes have some non-trivial limitations in their practical use, and the relation and combination of the aforementioned LR cryptography schemes have not been comprehensively discussed. It remains an open problem to determine how far away LR cryptography is from a *leak-free sanctuary*.

1.2 Our contributions

We present a cryptographic scheme and its security evaluation to address the abovementioned challenges.

New LR rekeying scheme. We propose a provably secure, higher-order, and LR rekeying scheme, named LR Rekeying with Random oracle Repetition (LR4). For this purpose, we introduce a new leakage model for rekeying and provide a formal security proof of LR4. We then show the validity of the leakage model and how to utilize LR4 in practice. We also discuss its practical aspects and analyze its implementation cost, efficiency, and low latency. We compare LR4 to state-of-the-art LR encryption schemes in Section 3.3, and confirm the advantage and effectiveness of LR4.

From a technical viewpoint, our major contributions include the new definition of leakage function, rather than the development of security proof technique or security notion. The definition of the leakage function for security notion/proof has been extensively studied, but its link to bounded trace complexity is largely unexplored. Currently, Accumulated Interference (AI) in [DMP22] (see Section 2.2) is the state-of-the-art for it. In this paper, we define another leakage function regarding trace complexity bound, which captures practical features of side-channel leakage and overcomes some drawbacks of existing leakage functions. We then prove the LR4 security using a promising security notion. Thus, LR4 achieves preferable features for practical rekeying (see Section 3), compared to existing LR schemes.

Evaluation methodology for practical usage. We propose an information-theoretical methodology for evaluating the attack cost and success rate on LR4 given a device/condition by utilizing and extending state-of-the-art SCA evaluation methods [dCGRP19, IUH22a, MRS22]. So far, the success rate has been commonly used for evaluating the SCA capability/resistance [SMY09], and many studies have been devoted for its practical and feasible estimation on symmetric primitive [MOS11, DFS15, dCGRP19, IUH22a, MRS22, BCG$^+$23], while there are few studies on the success rate evaluation on mode of operations (rather than primitive). Thus, we formally define the attack success rate on LR4, and then formally analyze the relationship between the attacks on LR4 and the underlying symmetric primitive, which enables a *quantitative* evaluation of the attack cost as success rate and the number of attack traces. Our methodology is able to determine the rekeying order d and the trace bound m as the rekeying interval from a quantity of target device (*i.e.*, mutual information or signal-to-noise ratio (SNR)).

Validation. Using the proposed methodology, we show a numerical evaluation of the attack cost on LR4 instantiated with AES. The results confirm an exponential increase of the number of secure calls of a symmetric primitive by increasing the rekeying order d under practical conditions. We also discuss properties required for a secure rekeying.

Technical challenges. The main technical challenge in our work is the simultaneous pursuit of high practicality and strong provable security on leakage resilience in rekeying mechanism. The former excludes several conventional techniques in LR cryptographic schemes, most notably a leak-free component and generation and transmission of true random values (where generation must also be secure against leakage). Our LR4 could be viewed as an alternative interpretation of classical GGM PRF with a counter, however, we need to show its leakage resilience both from theory and practice. This requires us to develop a dedicated security model capturing practical protection methods applicable to each module in LR4, such as higher-order masking of a *practical* order. Meanwhile, we should consider how to determine the parameter in the security proof (*i.e.*, trace bound m), given a device, for the practical usage of LR4 with a quantitative security guarantee. For evaluating LR4's practical security, we develop a formal definition of SCA success rate on LR4 and extend a state-of-the-art information theoretical evaluation method [dCGRP19] to its evaluation. Consequently, we are able to show that LR4 has exponential security with respect to the parameters (number of modules and leakage resistance of each module) both from practical and theoretical viewpoints.

1.3 Conventional studies on rekeying

Rekeying is one of the primordial countermeasures against DPA suggested by Kocher *et al.* [KJJ99]. The basic form of rekeying is illustrated in Figure 1. The rekeying schemes exploit the fact that most SCAs on symmetric primitives require a number of traces (*i.e.*, calls of the target primitive) for the key recovery. The basic idea behind the rekeying is to use a temporal key (*i.e.*, session key) k_{tmp} generated from a master key k_{mst}, and then update the session key using a deterministic rekeying function (g in Figure 1) and a (random) IV r at a frequency that does not allow any attackers to succeed in the temporal key recovery. Here, the target primitive is assumed to have a minimal resistance against SCAs with a certain number of traces (m-bounded DPA-resistance, or SPA-resistance if the number of traces is one) because the temporal key is discarded after the number of calls (or one). In contrast, the rekeying function should be DPA-resistant or *leak-free* because it is called many times with the master or internal temporal key.

The above idea was formalized by Medwed *et al.* in 2010 (MSGR) [MSGR10] as *Fresh Rekeying*. Fresh rekeying and its variants have been extensively studied from various viewpoints such as efficient instantiations, formal or practical security, and both with and without leakage [GFM13, MPR$^+$11, BDSH$^+$14, DEMM14, MS14, PM16, DFH$^+$16, DMMS21]. In particular, realization of key derivation function (g in Figure 1) is one of the central research topics in Fresh rekeying. MSGR suggested to use non-cryptographic operation (ring/field multiplication) for the key derivation based on the observation that key derivation should have no black-box security as the derived temporal key is never given in clear. Dobraunig *et al.* [DEMM14] pointed out a problem of this instantiation by showing a chosen-plaintext (master) key recovery attack. Their attack is a simple time-memory tradeoff using a set of precomputed ciphertexts with guessed temporal keys and a fixed plaintext. Dziembowski *et al.* [DFH$^+$16] proposed rekeying components based on lattice cryptography backed by a certain theoretical guarantee. This direction has been further

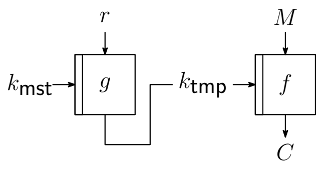

Figure 1: Basic form of rekeying scheme.

explored by Duval *et al.* [DMMS21]. Despite reports on some attacks [DEMM14, PM16], in reality, the root of security—an unbounded DPA-resistant/leak-free module—is barely implemented with sophisticated SCA countermeasures (*e.g.*, masking) to the best of authors' knowledge.

1.4 Paper organization

Section 2 introduces notations and existing attack/leakage models for LR cryptography. Section 3 proposes our higher-order and LR rekeying scheme, named LR4. Section 4 formally proves the security of LR4, with a formalization of leakage model for rekeying. Section 5 presents a quantitative and information-theoretical evaluation methodology based on formal analysis on attack cost and success rate on LR4 and how to use LR4 in practice. This is followed by Section 6, which demonstrates the validity of LR4 through numerical evaluations and discussion. Section 7 discusses the relation, comparison, and compatibility of LR4 with existing LR cryptographic schemes. Finally, Section 8 concludes this paper.

2 Preliminaries

2.1 Basic notations

Let $[i]$ denote $\{1, 2, \ldots, i\}$ for any positive integer i. We define $\{0,1\}^i$ and $\{0,1\}^*$ as the set of i-bit strings and the set of all arbitrary-length strings, respectively. Let log denote the binary logarithm.

A tweakable block cipher (TBC) is a keyed function $\widetilde{E} : \mathcal{K} \times \mathcal{T}_w \times \mathcal{M} \to \mathcal{M}$, where $\mathcal{K}$ is the key space, $\mathcal{T}_w$ is the tweak space, and $\mathcal{M} = \{0,1\}^n$ is the message space, such that for any $(K, T_w) \in \mathcal{K} \times \mathcal{T}_w$, $\widetilde{E}(K, T_w, \cdot)$ is a permutation over $\mathcal{M}$. We interchangeably write $\widetilde{E}(K, T_w, M)$, $\widetilde{E}_K(T_w, M)$, or $\widetilde{E}_K^{T_w}(M)$. The decryption routine is written as $(\widetilde{E}_K^{T_w})^{-1}(\cdot)$, where, if $C = \widetilde{E}_K^{T_w}(M)$ holds for some (K, T_w, M), we have $M = (\widetilde{E}_K^{T_w})^{-1}(C)$. When $\mathcal{T}_w$ is a singleton, it is essentially a block cipher and is written as $E : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$. For sets $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{T}_w$, $\mathrm{Func}(\mathcal{X}, \mathcal{Y})$ denotes the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$, $\mathrm{Perm}(\mathcal{X})$ denotes the set of all permutations over $\mathcal{X}$, and $\mathsf{TPerm}(\mathcal{T}_w, \mathcal{X})$ denotes the set of all functions $f : \mathcal{T}_w \times \mathcal{X} \to \mathcal{X}$ such that for any $T_w \in \mathcal{T}_w$, $f(T_w, \cdot)$ is a permutation over $\mathcal{X}$. A tweakable uniform random permutation (TURP) with a tweak space $\mathcal{T}_w$ and a message space $\mathcal{X}$, $\widetilde{\mathsf{P}} : \mathcal{T}_w \times \mathcal{X} \to \mathcal{X}$, is a random tweakable permutation with uniform distribution over $\mathsf{TPerm}(\mathcal{T}_w, \mathcal{X})$. The decryption is written as $(\widetilde{\mathsf{P}}^{-1})^{T_w}(\cdot)$ for TURP given tweak T_w. A random oracle (RO) is a random function that is uniformly distributed over $\mathsf{Func}(\{0,1\}^*, n)$ for some fixed n (which can be implemented with a lazy sampling). An ideal cipher (IC) is a random block cipher that is uniformly distributed over $\mathsf{TPerm}(\mathcal{K}, \mathcal{X})$ for some fixed finite sets $\mathcal{K}$ and $\mathcal{X}$ (that is, the set of all the block ciphers with key space $\mathcal{K}$ and message space $\mathcal{X}$). These are assumed to be publicly accessible when involved in the game. An IC accepts both encryption and decryption queries with a chosen key.

2.2 Attack/leakage models for LR cryptography

Data and trace bounded attacker. In an m-bounded data complexity model, the attacker can call the underlying symmetric primitive with m different plaintexts using an identical secret key [MR04, MSJ12]. If the data complexity bound m is sufficiently small, we expect that the attacker cannot obtain sufficient information about the secret intermediate variable (*e.g.*, Sbox output) and secret key. Thus, a key-recovery SCA is (believed to be) difficult if the number of available plaintexts is sufficiently bounded. However, Medwed *et al.* reported in [MSJ12] that, even if $m = 2$, the attacker can recover the secret key from low-noise devices (*e.g.*, a low-end microcontroller) with a non-trivial success probability. Accordingly, they suggested considering the number of available *traces* for an attack success. In this paper, we suppose that an m-trace bounded attacker can utilize not more than m traces. In the rekeying context, m corresponds to the rekeying interval. Note that a malicious attacker may call the primitive with a fixed input many times, especially if attacking the decryption module of a nonce/IV-based (authenticated) encryption scheme. This observation means that we should consider how to implement a symmetric primitive under m-bounded trace complexity in practice (which may require considering a high-level protocol). LR4 handles this consideration, as discussed in Section 3.2.

Bounded leakage function. In LR cryptography, secret information (*e.g.*, temporal key and internal value/state) leaks through each call of the underlying symmetric primitive. Let $s^{(i)}$ be the i-th state (which may include the i-th temporal key). At the i-th call, information about $s^{(i)}$ is leaked as $\mathscr{L}_i(s^{(i)})$ for each i, where $\mathscr{L}_i$ is the i-th leakage function. From practical constraints, we consider a non-adaptive leakage model: a fixed $\mathscr{L}$ exists such that $\mathscr{L} = \mathscr{L}_i$ for any i [SPY$^+$10]. The leakage function is given by, for example, some bits of $s^{(i)}$ and its Hamming weight with noise; or, it is defined using the number of leaked bits so that a state leaks at most λ bits. Note that the leakage function is usually *bounded* somehow. The attacker trivially wins if he/she gets all the information about the secret key or the intermediate value/state through leakage.

Accumulated interference (AI). Related to the bounded leakage function, Dobraunig *et al.* presented the concept of Accumulated Interference (AI) [DMP22] in 2022, which models leakages of permutation-based LR-AEs through both SCA and fault attacks. This model allows evaluating the attacker's advantage by the accumulated gain (AG). AG is defined using an input dataset for a given SCA, and AI is related to the trace complexity bound. However, AG has been evaluated only experimentally and empirically for specific SCAs; therefore, the evaluated advantage value cannot be an upper bound regarding all possible (even theoretically optimal) SCAs, which is essential for leakage resilience. Although an asymptotic approximation of the attacker's advantage for a given SCA is important to evaluate the SCA resistance of a device in some applications, a theoretical upper-bound evaluation is essential for both the theory and practice of LR cryptography. Dobraunig *et al.* then presented an LR encryption scheme asakey and its variant *strengthened* asakey based on AI/AG. In particular, the variant utilizes a caching strategy similar to LR4 for improved leakage resilience. In Section 3.3, we compare LR4 and asakey to demonstrate the significance of LR4.

Adaptive vs. non-adaptive leakages. Major existing LR cryptography schemes adopted a non-adaptive leakage model (*e.g.*, [DP10, YSPY10, FPS12, DMP22]). Actually, the practical validity of the non-adaptive leakage model/assumptions were discussed in [SPY$^+$10, YSPY10, FPS12]; power/EM attackers must fix leakage function before obtaining any leakage/output because of physical constraints of power/EM measurement (*e.g.*, on-board pin/connector and EM probe). Although extreme attackers might move

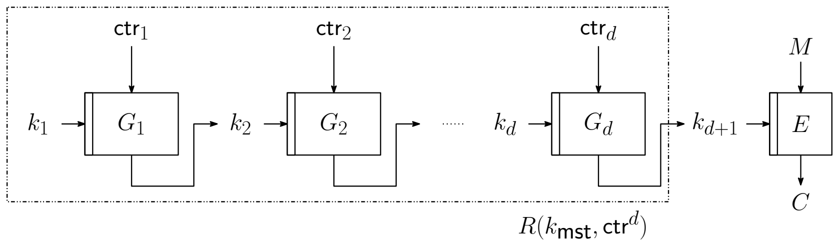

Figure 2: Encryption of LR4, where $k_1 = k_{\mathsf{mst}}$.

probe(s), its impracticality and meaninglessness were discussed in [SPY+10, FPS12]. Meanwhile, some impossibility results (difficulty in LR cryptography with practical construction under adaptive leakage) were shown in [SPY+10, FPS12]. Known practical remote power SCAs (*e.g.*, [ZS18, LKO+21]) also utilize non-adaptive leakage. Thus, non-adaptive leakages are more common, practical, and significant than adaptive ones, and we focus on non-adaptive leakage in this paper.

3 Proposed scheme

3.1 Basic concept

As discussed in Section 1, while (fresh) rekeying is a promising approach for LR cryptography, its full practicality is questionable due to the need for a leak-free function. We present a solution to this problem, dubbed LR4, and detail its construction below.

Let a positive integer d be the rekeying order. For each $i \in [d]$, let $G_i : \{0,1\}^{n_k} \times \{0,1\}^{n_{\mathsf{ctr}}} \to \{0,1\}^{n_k}$ be a function called a *rekeying component*, which takes an n_k-bit key and an n_{ctr}-bit counter and outputs an n_k-bit key. We require each G_i should behave like an independent RO; that is, for any input, the output looks *random*, and the outputs from the same input to G_i and G_j for $i \neq j$ are uncorrelated. Let $E, D : \{0,1\}^{n_k} \times \{0,1\}^{n_{\mathsf{bc}}} \to \{0,1\}^{n_{\mathsf{bc}}}$ denote the encryption and decryption routines of an n_{bc}-bit block cipher with an n_k-bit key. The LR4 rekeying scheme with order d consists of an encryption function LR4.$\mathcal{E}$ and a decryption function LR4.$\mathcal{D}$ such that LR4.$\mathcal{E}$, LR4.$\mathcal{D}$: $\{0,1\}^{n_k} \times (\{0,1\}^{n_{\mathsf{ctr}}})^d \times \{0,1\}^{n_{\mathsf{bc}}} \to \{0,1\}^{n_{\mathsf{bc}}}$. LR4.$\mathcal{E}$ (resp. LR4.$\mathcal{D}$) takes an n_k-bit master key k_{mst}, a d-tuple of n_{ctr}-bit counters $\mathsf{ctr}^d = (\mathsf{ctr}_1, \mathsf{ctr}_2, \ldots, \mathsf{ctr}_d)$, and an n_{bc}-bit plaintext (resp. ciphertext) as inputs, and outputs an n_{bc}-bit ciphertext (resp. plaintext). Figure 3 show LR4.$\mathcal{E}$ and LR4.$\mathcal{D}$ using a temporal key derivation function R, such that $R : \{0,1\}^{n_k} \times (\{0,1\}^{n_{\mathsf{ctr}}})^d \to \{0,1\}^{n_k}$. R takes an n_k-bit master key k_{mst} and a d-tuple of n_{ctr}-bit counters ctr^d as inputs, and generates a temporal (or session) key k_{tmp} (as defined in Figure 3). See also Figure 2 for illustration.

For LR security, we assume that each G_i does not leak anything up to $m \leq 2^{n_{\mathsf{ctr}}}$ encryption calls with the same key under SCAs. Similarly, we assume that E does not leak up to m' encryption and decryption calls with the same temporal key. See Section 4 for the formal treatment and Section 5 for how to determine m (and m').

A sound SCA countermeasure should increase the number of traces for an attack success by increasing the security parameter(s). The LR4 can generate m^d temporal keys securely under the m-bounded trace complexity model. Thus, the number of secure E calls increases exponentially by d, from m to m^d, under the bounded trace complexity model (although m should be determined dependently on d, as discussed in Section 5). Formal and rigorous security with a leakage model is defined and proven in Section 4. Note that the proposed scheme does not contribute to the decrease of an SCA success rate, although we confirm the exponential increase of the number of secure E calls under some practical

```
Algorithm LR4.$\mathcal{E}(k_{\mathsf{mst}}, \mathsf{ctr}^d, M)$
1 $k_{\mathsf{tmp}} \leftarrow R(k_{\mathsf{mst}}, \mathsf{ctr}^d)$
2 $C \leftarrow E_{k_{\mathsf{tmp}}}(M)$
3 return $C$
```

```
Algorithm LR4.$\mathcal{D}(k_{\mathsf{mst}}, \mathsf{ctr}^d, C)$
1 $k_{\mathsf{tmp}} \leftarrow R(k_{\mathsf{mst}}, \mathsf{ctr}^d)$
2 $M \leftarrow D_{k_{\mathsf{tmp}}}(C)$
3 return $M$
```

```
Algorithm $R(k_{\mathsf{mst}}, \mathsf{ctr}^d)$
1 $k_1 \leftarrow k_{\mathsf{mst}}$
2 for $i = 1$ to $d$
3     $k_{i+1} \leftarrow G_i(k_i, \mathsf{ctr}_i)$
4 $k_{\mathsf{tmp}} \leftarrow k_{d+1}$
5 return $k_{\mathsf{tmp}}$
```

Figure 3: Abstract of LR4 algorithm.

$$\begin{tikzcd}
 & & & & k_{1,1} \arrow[dlll] \arrow[d, dashed] \arrow[drrr, dash dot] & & & & \\
 & k_{2,1} \arrow[dl] \arrow[d, dashed] \arrow[dr, dash dot] & & & k_{2,2} \arrow[dl] \arrow[d, dashed] \arrow[dr, dash dot] & & & k_{2,3} \arrow[dl] \arrow[d, dashed] \arrow[dr, dash dot] & \\
k_{3,1} & k_{3,2} & k_{3,3} & k_{3,4} & k_{3,5} & k_{3,6} & k_{3,7} & k_{3,8} & k_{3,9}
\end{tikzcd}$$

Figure 4: Key diagram of LR4 when $d = 2$ and $m = 3$. Solid, dashed, and dash-dotted arrows mean that key is derived from key at parent's node when $\mathsf{ctr}_i = 0$, $\mathsf{ctr}_i = 1$, and $\mathsf{ctr}_i = 2$, for each $i \in [d] = [2]$, respectively. Temporal keys are generated and used from left to right.

conditions as demonstrated in Section 6.1. Thus, the proposed scheme can be another direction for provably secure SCA countermeasures apart from masking.

LR4 has a structural similarity to the classical GGM PRF, as the temporal key generation is represented by an m-ary tree with nodes of counter value, as shown in Figure 4. As mentioned in Section 1, GGM PRF has also been adopted by several LR-PRFs [FPS12, MSJ12, MS14], although our objective—an LR rekeying scheme without an (unbounded) leak-free component—makes LR4 different from these LR-PRFs in terms of the interface and the security notion under leakage. See Appendix A for GGM and existing LR-PRFs.

On implementation efficiency. Compared to the existing related LR cryptography (*e.g.*, GGM, the LR-PRFs described in Appendix A, and asakey), LR4 achieves an advantage of high-rate construction with provable security. For example, asakey, which is a state-of-the-art nonce-based and sponge-based LR encryption scheme in 2022 [DMP22], has a nonce-processing part with bit-by-bit absorption (represented by a binary tree like GGM) for its leakage resilience. Due to the leakage function definition, asakey is enforced to use a nonce processing part with a very low rate (*i.e.*, 1-bit absorption per permutation) for provable security with leakage resilience. In other words, it is difficult to achieve a high-rate construction with provable security under its leakage function, as well as the existing LR-PRFs. In contrast, LR4 is the first GGM-like LR scheme that achieves both the provable security and a high rate. For example, if we instantiate ROs using SHA-3 as $G_i(k_i, \mathsf{ctr}_i) = \mathsf{SHA\text{-}3}(k_i \parallel \mathsf{ctr}_i \parallel i)$, LR4 readily absorbs more than 128 bits with temporal key using only one SHA-3 computation, whereas asakey requires n permutations to absorb an n-bit nonce. See Section 3.3 for quantitative comparisons.

3.2 Caching intermediate keys for improved SCA security and computational efficiency

We propose that the computation of LR4 utilizes caching of all intermediate keys as long as they can be used later in order to improve the computational efficiency and SCA security. For example, if we increment a counter from 0 to 1 in Figure 2, $k_{2,1}$ ($k_{i,j}$ denotes the j-th intermediate/temporal key of k_i), which has been computed by processing counter 0, is cached to derive $k_{3,2}$ for counter 1, and the computing device releases it when the counter becomes 3. This is essential because re-computation of an intermediate key would lead to unexpected side-channel leakage violating the trace bound (see also Remark 1).

Figure 5 displays the *cache-based* version of the temporal key derivation function R, denoted by R_C. It caches the intermediate keys and counters for given m and m', where $\mathtt{kh}_i$ and $\mathtt{ch}_i$ denote the caches for i-th intermediate key $\mathtt{kh}^{d+1} = (\mathtt{kh}_1, \mathtt{kh}_2, \ldots, \mathtt{kh}_d, \mathtt{kh}_{d+1})$ and counter values $\mathtt{ch}^{d+1} = (\mathtt{ch}_1, \mathtt{ch}_2, \ldots, \mathtt{ch}_d, \mathtt{ch}_{d+1})$, respectively. Unlike Figure 2 and Figure 3, we require an additional counter ctr_{d+1} to guarantee that E is called not more than m' times with an identical k_{tmp} (note that the value of ctr_{d+1} has no influence on the output, except for $\perp$). Here, we consider a total counter value as $\sum_{i=1}^{d+1} 2^{n_{\mathsf{ctr}}(i-1)} \mathsf{ctr}_{d+2-i}$. This indicates that the counter is incremented from ctr_{d+1}. In R_C, we first check the validity of input counters to detect counter replay attacks at Line 1. If the counter value is smaller than the cached one (*i.e.*, replayed value) or out of range (*i.e.*, $\mathsf{ctr}_i \geq m$ for $1 \leq i \leq d$ or $\mathsf{ctr}_{d+1} \geq m'$), we abort the encryption/decryption. Otherwise, we compute the temporal key from the counters and master key with minimal computations. If $\mathsf{ctr}_i = \mathtt{ch}_i$ and $\mathtt{flag} = 0$, the computation is omitted and the cached key is used to avoid the leakage (where $\mathtt{flag}$ represents the necessity of computation). Note that $\mathtt{ch}^{d+1}$ and ctr^{d+1} are known to the attacker, which indicates that the side-channels (*e.g.*, power/EM and timing) of the **if** branches in R_C leak no secret information.

Latency and computational efficiency. The caching strategy also improves the latency and computational efficiency[2], as stated in Proposition 1. This is a substantial improvement over the straightforward computation requiring d times G.

Proposition 1. *For a rekeying interval m, we need to run G only $1 + 1/(m-1) < 2$ times on average.*

Proof. Let d and m denote the rekeying order and interval, respectively. By definition, LR4 takes at most m^d distinct counter values. The average number of G calls for processing these m^d values is

$$\frac{1}{m^d}\left(\sum_{i}^{d} i m^{d-i}(m-1) + d\right) = \frac{d}{m^d} + (m-1)\sum_{i=1}^{d} i m^{-i}. \quad (1)$$

Let $S_d := \sum_{i=1}^{d} i m^{-i}$. Taking the sum of geometric progression, we obtain

$$S_d = \frac{m}{m-1}\left(\frac{m^{-1}(1-m^{-d})}{1-m^{-1}} - \frac{d}{m^{d+1}}\right) = \frac{1}{m-1}\left(\frac{1-m^{-d}}{1-m^{-1}} - \frac{d}{m^d}\right).$$

[2]This is also (briefly) mentioned in [MSJ12, Section 6] when their LR-PRF/encryption are used in the CTR mode. In [DMP22, Section 3.2], a caching strategy is also (briefly) discussed for their LR encryption. However, neither analyses on computational cost nor some practical aspects (*e.g.*, replay detection) were explicitly discussed.

```
Algorithm $R_C(\mathsf{ctr}^{d+1}; \mathtt{kh}^{d+1}, \mathtt{ch}^{d+1})$
1  if $\sum_{i=1}^{d+1} 2^{n_{\mathsf{ctr}}(i-1)}\mathsf{ctr}_{d+2-i} \le \sum_{i=1}^{d+1} 2^{n_{\mathsf{ctr}}(i-1)}\mathtt{ch}_{d+2-i}$ or $\exists i, \mathsf{ctr}_i \ge m$ or $\mathsf{ctr}_{d+1} \ge m'$
2    return $\perp$ (Abort encryption/decryption)
3  else if
4    $\mathtt{flag} \leftarrow 0$
5    for $i = 1$ to $d$
6      if $\mathsf{ctr}_i > \mathtt{ch}_i$ and $\mathtt{flag} = 0$
7        $\mathtt{flag} \leftarrow 1$
8      if $\mathtt{flag} = 1$
9        $\mathtt{kh}_{i+1} \leftarrow G_i(\mathtt{kh}_i, \mathsf{ctr}_i)$
10   $k_{\mathsf{tmp}} \leftarrow \mathtt{kh}_{d+1}$
11   $\mathtt{ch}^{d+1} \leftarrow \mathsf{ctr}^{d+1}$
12   return $k_{\mathsf{tmp}}; \mathtt{kh}^{d+1}, \mathtt{ch}^{d+1}$
```

Figure 5: The cache-based version of the temporal key derivation function R. LR4 with cache invokes R_C instead of R. The caches $\mathtt{ch}^{d+1}$ and $\mathtt{kh}^{d+1}$ are initially given as $\mathtt{ch}^{d+1} = (0, 0, \ldots, 0)$, $\mathtt{kh}_{i+1} = G_i(\mathtt{kh}_i, 0)$ for each $1 \le i \le d$, and $\mathtt{kh}_1 = k_{\mathsf{mst}}$. Here, $\mathtt{kh}_i$ should be called at Line 9 only when it is actually used.

Hence, the right-hand side of Equation (1) is

$$\begin{aligned}\frac{d}{m^d} + (m-1)S_d = \frac{1-m^{-d}}{1-m^{-1}} &= \frac{m^d - 1}{m^d} \cdot \frac{m}{m-1} \\ &< \frac{m}{m-1} = 1 + \frac{1}{m-1} \\ &< 2.\end{aligned}$$

This completes the proof. □

Memory overhead. The cache-based LR4 requires a non-volatile memory (NVM) to cache the intermediate keys. The memory overhead is given by $(n_k + n_{\mathsf{ctr}})(d+1)$ bits. For a practical parameter of $n_k = 128$ and $n_{\mathsf{ctr}} = 20$ (see Section 6.1), the memory overhead is $148(d+1)$ bits (*e.g.*, 888 bits (= 111 bytes) when $d = 5$), which is efficient and sufficiently practical, compared to existing ones (see Section 3.3). For example, even low-end microcontrollers such as Atmel AVR Xmega128D series, which may be a major target of SCA countermeasures, have a 16K–128K byte flash memory and 1K–2K byte EEPROM. The memory overhead of LR4 is less than 10% of the very low-end ones. Thus, we confirm the practicality of cache-based LR4.

Explicit synchronization and replay detection. As the nonce (*i.e.*, counters) forms a total order and LR4 caches its internal counters, LR4 detects replayed queries by comparing the orders of query and internal counters at the validity check at Line 1 in $R_C(\mathsf{ctr}^{d+1}; \mathtt{kh}^{d+1}, \mathtt{ch}^{d+1})$, before performing encryption/decryption. So, the attacker cannot perform any replay (for trace averaging and trace complexity bound violation). In contrast, if receiving a forwarded nonce (maliciously or accidentally), the LR4 module updates the internal counter to the nonce at Line 11 (as well as a valid counter) and performs encryption/decryption using the temporal key corresponding to the values. This is because such counter-forwarding never yields a violation of trace bound. Thus, LR4 offers secure explicit synchronization for the cases of malicious/accidental synchronization

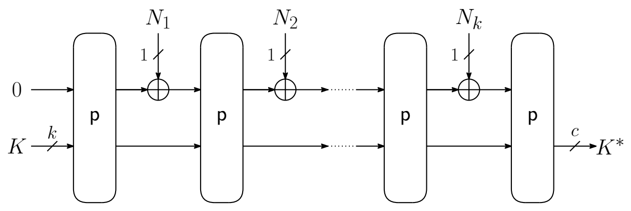

Figure 6: The rekeying function of asakey. K is a master key, and p is an underlying permutation. $N_1, \ldots, N_k$ are one-bit split nonces where an input nonce N is written as $N_1 \parallel \cdots \parallel N_k$. K^* is a derived temporal key input to the sponge-based encryption part. ISAP has a similar rekeying function as this construction.

failures. Note that denial-of-service attacks by counter forwarding to exhaust key lifetime remains an open problem to be prevented, while LR4 can prevent key-recovery SCAs.

Remark 1 (SCA security and potential threat). This paper focuses on a countermeasure against *non-invasive* power/EM analysis, in which any cached key does not leak unless it is called [MOP07]. We consider the trace complexity bound as how many times the cached key is called; thus, the caching strategy essentially improves the SCA security of LR4. In contrast, *(semi-)invasive* attacker may utilize a static leakage directly from memory components [MOP07, SSAQ02]. Such a (semi-)invasive attacker attempts to directly read secret/cached keys to bypass SCA countermeasures. For such cases, the number of calls and trace complexity bound are no longer meaningful. Such (semi-)invasive attacks are outside the scope, as in many existing studies on LR cryptography and SCA countermeasures. (Semi-)Invasive attacks should be prevented by, for example, tamper-resistant memory and memory encryption, rather than SCA countermeasures.

3.3 Comparison with state-of-the-art

The high rate and leakage function of LR4 yield an efficient implementation with provable security, in comparison with existing LR cryptography such as asakey [DMP22] and ISAP [DEM+20]. asakey and ISAP are sponge-based encryption and AE, respectively, and thus they are functionally different from LR4. However, LR4's temporal key derivation function R and asakey/ISAP's rekeying function are functionally compatible; therefore, we compare them. As depicted in Figure 6, asakey uses a nonce-processing part as a rekeying function, which consists of a bit-by-bit absorption of the nonce, followed by a sponge-based encryption. The intermediate state derivation in the nonce absorption of asakey is representable as a binary GGM. The *strengthened* asakey, which is a cache-based variant with an up-counter nonce, caches the intermediate states during the nonce processing part. To validate the effectiveness of LR4, we show its comparison to the strengthened asakey. We instantiate the strengthend asakey using Keccak-$p[1600, 12]$ with 128-bit key according to [DMP22], while we instantiate RO in LR4 using Keccak-$p[1600, 24]$ (*i.e.*, SHA-3). We mainly discuss (strengthened) asakey in the following paragraphs because ISAP has a similar rekeying function as that of asakey. Note that ISAP has more parameters than asakey, such as the length of nonce absorption per one permutation call and the number of rounds of the underlying permutations. However, all instances of ISAP determine the length of nonce absorption to be one, which is the same as asakey. The number of rounds of the underlying permutations varies in each instance (mainly 1 round or 12 rounds), but we leave out the schemes using 1-round permutations from our comparison since we would like to focus on the schemes having provable security (also see the last paragraph in comparison of computational cost/latency).

Memory overhead. The strengthened asakey requires an NVM to cache all its internal states, whose bit length is the product of the permutation length (1,600 here) and key/nonce length. In the above instantiation, the memory size of asakey is $1600 \times 128 = 204800$ bits. Although it can be reduced by limiting the number of calls by m, the required NVM are given by $1600 \log m$ bits, which is still very high cost for practical m. In contrast, cache-based LR4 requires an NVM of only $(n_k + n_{\mathsf{ctr}})(d+1)$ bits, which is far fewer for realistic d (*e.g.*, 888 bits for $d = 5$ and a practical parameter of $n_k = 128$ and $n_{\mathsf{ctr}} = 20$ for example). Thus, LR4 is adoptable of even low-end microcontrollers with a 16 K–128 K byte flash memory and 1 K–2 K byte EEPROM as mentioned before.

Computational cost/latency. The strengthened asakey must call at least one Keccak-$p[1600, 12]$ for each encryption call, while LR4 calls at least one Keccak-$p[1600, 24]$ (*i.e.*, SHA-3) per m' E_K calls, where m' is trace bound for E_K. Note that E_K can be instantiated with an (LR-)AE. For example, as sponge-based encryption like ISAP or Ascon [DEM+20, DEMS21], which can encrypt a message with (practically-)arbitrary bit length by only one E_K call (see also Section 6.2.1). On average per E_K call, LR4 requires less than $2/m'$ Keccak-$p[1600, 24]$ calls (as proven in Proposition 1), while asakey always requires at least one Keccak-$p[1600, 12]$ call for nonce-processing. As m' is usually greater than 10 (see Section 6.1), cache-based LR4 has a far lower latency than strengthened asakey on average. Also, for non-cached version, LR4 requires only d Keccak-$p[1600, 24]$ calls while strengthened asakey requires 128 Keccak-$p[1600, 12]$ calls.

Moreover, to improve computational cost and latency, asakey and ISAP specify instantiations that use reduced-round permutations. For example, two instances of ISAP employ a 1-round Keccak/Ascon permutation in the nonce processing part except for the first and the last permutations. It significantly reduces computation cost and latency. To the best of our knowledge, any practical weaknesses have not been reported in these instances. However, in the asakey/ISAP's security proofs, the underlying primitive should be a public random permutation, which is intepreted as non-existence of structural distinguisher in practice. The use of a 1-round permutation implies a deviation from this assumption.

Provable security. The LR notions of asakey and LR4 share a (common) principle; consider a distinguishing game involving a leaky oracle, non-leaky oracle, and an idealized primitive oracle, while asakey's leakage model and ours are different and incomparable. Our model captures features of practical SCA countermeasures (*e.g.*, higher-order masking) and dedicated to the tree-based re-keying schemes. As Section 4.3 shows, LR4 can be used as a replacement for leak-free/fresh-rekeying components in some of the existing LR-AEs (*e.g.*, [Men20]), which helps make these schemes real.

Deleated and moved into above partially: We hereafter compare LR4 to ISAP's rekeying function (it is similar to asakey). ISAP uses lightweight primitives to overcome its low rate. In fact, two schemes of the ISAP family use the 1-round permutation of ascon-p/keccak-$p[400]$ as a primitive in their rekeying functions. However, in the asakey/ISAP's security proofs, the underlying primitive should be a random permutation, which is far from a 1-round permutation. Thus, its rekeying function cannot avoid a huge gap between provable security and practical construction. In contrast, LR4 has both provable security and efficiency.

Leakage evaluation. In [DMP22], asakey's leakage resilience was evaluated for specific SCAs (*e.g.*, CPA) to derive the AG value. In other words, asakey's leakage resilience can be evaluated for SCAs *feasible by the evaluator/designer*. However, an advanced attacker may mount stronger SCAs, which makes the asakey's practical security unclear. In contrast, LR4's security proof and leakage evaluation method (in Section 5.2) consider theoretically-optimal SCAs (*i.e.*, most advanced SCA attacker) and capture the practical aspects of (higher-order) masking. Thus, LR4's leakage resilience includes practical security and covers all possible SCA attackers including one evaluated in [DMP22].

4 Provable security analysis on LR4

4.1 Security definition

We introduce a formal security notion under leakage for rekeying schemes, including LR4. The core idea of our security model is the same as [BMOS17]. We define the security of an LR rekeying scheme as the probability that an adversary querying some leakage oracles successfully distinguishes the two worlds: real and ideal. Following Mennink [Men20], we model a rekeying scheme as a TBC, where a tweak corresponds to the IV (r in Figure 1) of a rekeying scheme. Our model allows the adversary to choose tweaks arbitrarily in the game and hence is more general than assuming it is a random value or a counter (although a practical choice would be a counter). In the real world, the adversary accesses LR4.$\mathcal{E}$ and LR4.$\mathcal{D}$, while, in the ideal world, it accesses a TURP $\widetilde{\mathsf{P}}$ of tweak space $\{\{0,1\}^{n_{\mathsf{ctr}}}\}^d$ and message space $\{0,1\}^{n_{\mathsf{bc}}}$. Regarding leakage oracles, we define LR4-L.$\mathcal{E}$ and LR4-L.$\mathcal{D}$ as those of LR4.$\mathcal{E}$ and LR4.$\mathcal{D}$, respectively. We will detail them later. We also assume that $G_1, G_2, \ldots, G_d$ are independent ROs, and E and D are IC. We define the *leakage-resilience of Fresh Rekeying* (LFR) advantage for the security of LR4 as

$$\mathrm{Adv}^{\mathrm{LFR}}_{\mathrm{LR4}}(\mathcal{A}) \coloneqq \left|\Pr\left[\mathcal{A}^{\mathrm{LR4}^{\pm}, G, E^{\pm}, \mathrm{LR4\text{-}L}^{\pm}} \to 1\right] - \Pr\left[\mathcal{A}^{\widetilde{\mathsf{P}}^{\pm}, G, E^{\pm}, \mathrm{LR4\text{-}L}^{\pm}} \to 1\right]\right|,$$

where LR4$^{\pm}$ denotes a pair of oracles LR4.$\mathcal{E}$ and LR4.$\mathcal{D}$, and LR4-L$^{\pm}$ denotes LR4-L.$\mathcal{E}$ and LR4-L.$\mathcal{D}$. Also, $\widetilde{\mathsf{P}}^{\pm}$ (resp. $E^{\pm}$) denotes $\widetilde{\mathsf{P}}$ (resp. E) and its inverse $\widetilde{\mathsf{P}}^{-1}$ (resp. D), and $G \coloneqq \{G_1, G_2, \ldots, G_d\}$. We call LR4$^{\pm}$ and $\widetilde{\mathsf{P}}^{\pm}$ construction oracles and call queries to them construction queries. Similarly, we call LR4-L$^{\pm}$ leakage oracle and call queries to it leakage queries. The use of idealized primitives, such as RO, can be found in the theoretical analysis of LR schemes, particularly for obtaining efficient schemes [YSPY10, BGP+19, DJS19, DM19, GSWY20, FPS12]. See also [BBC+20] for an overview and discussion.

Leakage oracle. We here define LR4-L$^{\pm}$ to capture the m-bounded trace complexity model introduced in Section 3.1, including caching keys shown in Section 3.2. We assume that LR4-L.$\mathcal{E}$ and LR4-L.$\mathcal{D}$ have the same input/output as LR4.$\mathcal{E}$ and LR4.$\mathcal{D}$, except that they additionally output leakage $\mathsf{Leak} \in (\{0,1\}^{n_k} \cup \{\perp\})^{d+1}$. For the definition of Leak, recall that we assume each rekeying component and a block cipher can securely perform m and m' calls with the same key, respectively. Also, we assume $m = 2^{n_{\mathsf{ctr}}}$ to simplify the proof. To capture this leakage assumption, we define that LR4-L.$\mathcal{E}$ and LR4-L.$\mathcal{D}$ leak an *overused* key, which we detail below. We first assume that the leakage oracle records all the invoked intermediate and the temporal key values of the underlying G and E that appeared in the queries to the leakage oracle. For a query to the leakage oracle, when the first $i \in [d]$ counters $(\mathsf{ctr}_1, \ldots, \mathsf{ctr}_i)$ are the same as some previous queries, the leakage oracle merely refers to the memorized key value of $(i+1)$-th intermediate key k_{i+1}; thus, they do not invoke $G_1, G_2, \ldots, G_i$. Then, for $i \in [d]$, when G_i is invoked with the same key (*e.g.*, $k_i \in \{0,1\}^{n_k}$) more than m times, we define k_i as an overused key and set the i-th element of Leak as its value k_i. Otherwise, we set it as $\perp$. Similarly, when E is invoked with the same temporal key k_{d+1} m' times, we define k_{d+1} as an overused key and set the $(d+1)$-th element of Leak as k_{d+1}; otherwise, we set it as $\perp$. We also write $\mathsf{Leak} = \perp$ to mean that no keys are overused, *i.e.*, $\mathsf{Leak} = \perp = (\perp, \perp, \ldots, \perp)$. Here, our model regards the side-channel leakage with m traces during the computation of all intermediate values and outputs. Thus, our leakage model and proof consider the best possible SCAs (including an optimal SCA in Section 5.1).

Query rules. We assume the adversary can query the same counters up to m' times in the leakage queries to prevent a trivial leak of a temporal key. Note that there is no restriction on repeating the same counters in the construction queries. Also, we suppose the adversary does not perform *repeating/replaying* and *forwarding* queries; it does not repeat a query across different oracles or the same oracle. In construction queries, we assume that the adversary does not query (ctr, C) to LR4.$\mathcal{D}$ after querying (ctr, M) to LR4.$\mathcal{E}$ and obtaining C, and *vice versa.* The same assumption applies to the leakage queries. Note that the adversary can query any oracle in any order and can query counters in any order in construction and leakage queries.

Relation to existing leakage models. Our security notion under leakage is defined with a distinguishing game consisting of real and ideal worlds involving leaky and non-leaky (classical) oracles, where the former is in the both worlds. The latter in the real world performs real encryption whereas in the ideal world, it is idealized and returns always random. We also allow the adversary to query primitive oracles (G and E) in the both worlds. This framework itself is identical to those in the literature [BMOS17, KS20, DEM$^+$20, DMP22]. The main difference is the definition of leakage function (the response of leaky oracle). A leakage function in existing models is stateless, namely does not reflect the query/response history, while ours depends on the previous queries as we care about how many times the same key has been used in each rekeying module. In addition, leakage functions in the existing LR-AEs, such as [BMOS17, KS20], are a direct composition of those defined for internal components (*e.g.*, key/tag derivation functions, encryption function and message hashing function). The leakage functions for leak-free components (typically key/tag derivation functions) is vacuous and those for leaky components leak everything about its computation. In case of sponge-based LR-AEs, the internal primitive is typically a single cryptographic permutation and the leakage function determines the input and output leakage per every permutation call occurred in an encryption/decryption query to the leaky oracle. LR4 consists of two components, G and E, and defines different leakage functions that are dependent on the query histories. So our model shares some basic principles with existing works, however, the leakage function is dedicated to capture what is aimed by practical protection methods, *e.g.*, high-order masking.

4.2 Security bound for LR4

Theorem 1. *Let $\mathcal{A}$ be the adversary following* LFR *game. Let q be the total number of construction queries, and q_L be the total number of leakage queries. For $i \in [d]$, we assume $\mathcal{A}$ queries p_i times to G_i and p_I times to E. Then, we have*

$$\mathrm{Adv}^{\mathrm{LFR}}_{\mathrm{LR4}}(\mathcal{A}) \leq \frac{d(q+q_L)^2}{2^{n_k+1}} + \frac{(q+q_L)(p+p_I)}{2^{n_k}} + \frac{4m'q}{2^{n_{\mathrm{bc}}}},$$

where $p = \sum_{i=1}^{d} p_i$ and $q \leq 2^{n_{\mathrm{bc}}-1}$. We also assume $q_L \leq m' 2^{n_{\mathrm{ctr}} d}$.

This theorem shows that LR4 has birthday-bound security regarding the internal key length and almost optimal security regarding the block cipher length since m' is small. Also, the term $(q+q_L)(p+p_I)/2^{n_k}$ indicates the relationship between the upper bounds of online and offline complexities that the adversary requires to attack LR4.

Proof. We use the H-Coefficient technique [Pat08, CS14] for the proof. See Section B.1 for the technical background. We first define *transcripts*, a set of input/output values of oracles in the LFR game the adversary obtains. Let $\mathcal{Q}_C$, $\mathcal{Q}_{G_1}$, …, $\mathcal{Q}_{G_d}$, $\mathcal{Q}_E$, and $\mathcal{Q}_L$ be the transcripts consisting of input/output of the construction oracle, G_1, …, G_d, E and D, and the leakage oracle, respectively. In detail, we define $\mathcal{Q}_C = \{(\mathsf{ctr}^d_1, M_1, C_1), \ldots, (\mathsf{ctr}^d_q, M_q, C_q)\}$, $\mathcal{Q}_{G_i} = \{(\mathsf{IK}_{i,1}, \mathsf{IV}_{i,1}, \mathsf{OK}_{i,1}), \ldots, (\mathsf{IK}_{i,p_i}, \mathsf{IV}_{i,p_i}, \mathsf{OK}_{i,p_i})\}$ for each $i \in [d]$, where $\mathsf{IK}_{i,\cdot}$ is the input key; $\mathsf{IV}_{i,\cdot}$ is the input counter; and $\mathsf{OK}_{i,\cdot}$ is the output key. We also define $\mathcal{Q}_E = \{(K_1, X_1, Y_1), \ldots, (K_{p_I}, X_{p_I}, Y_{p_I})\}$, where $K_\cdot$ is the input key; $X_\cdot$ is the plaintext; and $Y_\cdot$ is the ciphertext. In addition, we define $\mathcal{Q}_L = \{(\mathsf{Lctr}^d_1, LM_1, LC_1, \mathsf{Leak}_1), \ldots, (\mathsf{Lctr}^d_{q_L}, LM_{q_L}, LC_{q_L}, \mathsf{Leak}_{q_L})\}$.

To simplify the security proof, we assume that, after $\mathcal{A}$ finishes its interactions with all oracles, the leakage oracle (LR4-L.$\mathcal{E}$ and LR4-L.$\mathcal{D}$) reveals all the involved keys (the master key, intermediate keys, and temporal keys) in computing the outputs. Then the construction oracle also reveals keys involved in its output computations. Note that constructions oracle in the ideal world uses $\widetilde{\mathsf{P}}$ and $\widetilde{\mathsf{P}}^{-1}$ hence have no keys to reveal; instead, the oracle outputs dummy values sampled uniformly at random from $\{0,1\}^{n_k}$. To prevent a trivial win of $\mathcal{A}$, the construction oracle (of both worlds) does not reveal the keys already revealed by the leakage oracle. For example, in Figure 4, assume that the adversary queries three counters $(\mathsf{Lctr}^d_1, \mathsf{Lctr}^d_2, \mathsf{Lctr}^d_3) = ((0,0),(1,0),(1,1))$ to the leakage oracle and queries three counters $(\mathsf{ctr}^d_1, \mathsf{ctr}^d_2, \mathsf{ctr}^d_3) = ((0,1),(1,1),(2,0))$ to the construction oracle. In this case, as shown in Figure 7, the leakage oracle reveals ($k_{1,1}$, $k_{2,1}$, $k_{3,1}$, $k_{2,2}$, $k_{3,4}$, $k_{3,5}$) and then the construction oracle reveals only ($k_{3,2}$, $k_{2,3}$, $k_{3,7}$) since $k_{1,1}$, $k_{2,1}$, $k_{2,2}$, and $k_{3,5}$ are already revealed by the leakage oracle. Let $\mathcal{Q}_K \coloneqq \{k^{(a)}_{i,j} : a \in \{0,1\}, i \in [d+1], j \in [2^{n_{\mathsf{ctr}}(i-1)}]\}$ be the transcript of keys revealed by the leakage and the construction oracles, where a indicates which oracle reveals the key: $a = 0$ means that the leakage oracle reveals the key, and $a = 1$ means that the construction oracle does. The index i indicates the depth of the key, and j indicates the index of the key in i-th depth keys, as shown in Figure 4 and Figure 7. In the case of the above example (*i.e.*, Figure 7), the adversary obtains $\mathcal{Q}_K = \{k^{(0)}_{1,1}, k^{(0)}_{2,1}, k^{(0)}_{2,2}, k^{(1)}_{2,3}, k^{(0)}_{3,1}, k^{(1)}_{3,2}, k^{(0)}_{3,4}, k^{(0)}_{3,5}, k^{(1)}_{3,7}\}$. Thus, all transcripts $\mathcal{A}$ obtains are $\{\mathcal{Q}_C, \mathcal{Q}_{G_1}, \ldots, \mathcal{Q}_{G_d}, \mathcal{Q}_E, \mathcal{Q}_L, \mathcal{Q}_K\}$.

We introduce four *bad events*. Roughly, if the transcripts defined above fulfill any bad event, the adversary successfully distinguishes two worlds with high probability.

Bad1: A collision between the elements of $\mathcal{Q}_K$ in the same depth. That is, the event that there exists $i \in [d+1], j_1, j_2 \in [2^{n_{\mathsf{ctr}}(i-1)}]$, $j_1 \neq j_2$, and $a_1, a_2 \in \{0,1\}$ s.t. $k^{(a_1)}_{i,j_1} = k^{(a_2)}_{i,j_2}$. This event also includes the event that $\mathcal{A}$ obtains Leak other than $\perp$.

Bad2: A collision between the revealed key in i-th depth and the input key of G_i, where $i \in \{1,\ldots,d\}$. That is, the event that there exists $i \in \{1,\ldots,d\}$, $j \in [2^{n_{\mathsf{ctr}}(i-1)}]$, $a \in \{0,1\}$, $j' \in [p_i]$, s.t. $k^{(a)}_{i,j} = \mathsf{IK}_{i,j'}$.

Bad3: A collision between the revealed temporal key and the input key of E. That is, the event that there exists $j \in [2^{n_{\mathsf{ctr}}d}]$, $j' \in [p_I]$, $a \in \{0,1\}$ s.t. $k^{(a)}_{d+1,j} = K_{j'}$.

Bad4: A ciphertext (resp. plaintext) collision between construction queries and leakage queries when counters are the same and plaintexts (resp. ciphertexts) are distinct. That is, the event that there exists $i \in [q]$, $j \in [q_L]$, and $\mathsf{ctr}^d_i = \mathsf{Lctr}^d_j$ s.t. $M_i = LM_j$ or $C_i = LC_j$.

An upper bound of $\mathrm{Adv}^{\mathrm{LFR}}_{\mathrm{LR4}}(\mathcal{A})$ would correspond to an upper bound of $p_{\mathrm{bad}} \coloneqq \Pr[\mathrm{Bad1} \cup \mathrm{Bad2} \cup \mathrm{Bad3} \cup \mathrm{Bad4}]$ in the ideal world. This argument holds because the second part of the H-Coefficient technique, the so-called *good transcript probability ratio*, is lower bounded by 1 (see *e.g.*, [CS14] for details). We here move out the derivation of this part to Appendix B because it is a typical one for birthday-secure constructions and is tedious but rather straightforward. For readers unfamiliar with H-Coefficient, we refer

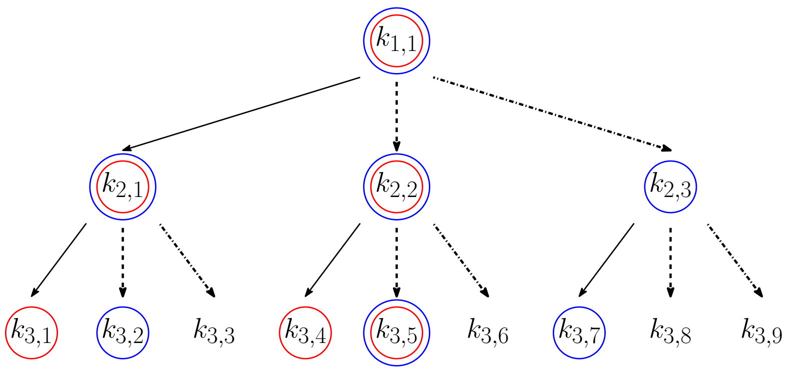

Figure 7: Example of key reveal procedure in the proof. The leakage queries are for the leaves $(3,1)$, $(3,4)$ and $(3,5)$ and the relevant (intermediate) keys are circled by red. The construction oracle queries are for the leaves $(3,2)$, $(3,5)$ and $(3,7)$ and the relevant (intermediate) keys are circled by blue. First, the keys circled by red will be revealed. Second, the keys circled by blue will be revealed—following the tree in the real world and randomly sampled in the ideal world—except those not already revealed (*i.e.*, circled by red). Thus, the latter step only reveals $k_{3,2}$, $k_{2,3}$, and $k_{3,7}$.

to [CLS15, Theorem 1] to get an idea of how typical H-coefficient proofs with the case that the good transcript probability ratio is larger than one are conducted.

We evaluate p_{bad} in the ideal world. We start by evaluating $\Pr[\text{Bad1}]$. For each $i \in [d+1]$, let nk_i be the number of elements $k_{i,\cdot}^{(\cdot)}$ in $\mathcal{Q}_K$ (*i.e.*, the number of revealed keys in i-th depth). Now, we have $\mathsf{nk}_1 = 1$ assuming $q+q_L \neq 0$, and $\mathsf{nk}_1 \leq \mathsf{nk}_2 \leq \cdots \leq \mathsf{nk}_{d+1} \leq q+q_L$. In the ideal world, the elements $k_{\cdot,\cdot}^{(1)}$ are chosen uniformly at random and independently, and $k_{\cdot,\cdot}^{(0)}$ are derived from ROs. Thus, we obtain $\Pr[\text{Bad1}] \leq \sum_{i=2}^{d+1} \binom{\mathsf{nk}_i}{2} \cdot 1/2^{n_k} \leq d(q+q_L)^2/2^{n_k+1}$. We then evaluate $\Pr[\text{Bad2} \cap \overline{\text{Bad1}}] \leq \Pr[\text{Bad2} \mid \overline{\text{Bad1}}]$. We obtain $\Pr[\text{Bad2} \mid \overline{\text{Bad1}}] \leq \sum_{i=1}^{d} \mathsf{nk}_i p_i/2^{n_k} \leq \sum_{i=1}^{d} (q+q_L)p_i/2^{n_k} = (q+q_L)p/2^{n_k}$. For Bad3, $\Pr[\text{Bad3} \cap \overline{\text{Bad1}} \cap \overline{\text{Bad2}}] \leq \Pr[\text{Bad3} \mid \overline{\text{Bad1}} \cap \overline{\text{Bad2}}] \leq (q+q_L)p_I/2^{n_k}$ holds. For Bad4, we define Cnc as the number of distinct counters in construction queries, and $\widetilde{\mathsf{ctr}_1^d}, \ldots, \widetilde{\mathsf{ctr}_{\mathsf{Cnc}}^d}$ as the distinct counters. Let $q_1, \ldots, q_{\mathsf{Cnc}}$ be the number of construction queries whose counter is $\widetilde{\mathsf{ctr}_1^d}, \ldots, \widetilde{\mathsf{ctr}_{\mathsf{Cnc}}^d}$, respectively; thus, $\sum_{i=1}^{\mathsf{Cnc}} q_i = q$. We evaluate $\Pr[\text{Bad4} \cap \overline{\text{Bad1}} \cap \overline{\text{Bad2}} \cap \overline{\text{Bad3}}] \leq \Pr[\text{Bad4} \mid \overline{\text{Bad1}} \cap \overline{\text{Bad2}} \cap \overline{\text{Bad3}}]$. For each $i \in [\mathsf{Cnc}]$, recall that the adversary queries to LR4-L with the counter $\widetilde{\mathsf{ctr}_i^d}$, at most m' times. Assuming $\overline{\text{Bad1}} \cap \overline{\text{Bad2}} \cap \overline{\text{Bad3}}$ happens, for each $i \in [\tau]$, the probability of a plaintext/ciphertext collision is at most $\sum_{j=0}^{q_i-1} m'/(2^{n_{\mathsf{bc}}}-j) + \sum_{j=0}^{m'-1} q_i/(2^{n_{\mathsf{bc}}}-j) \leq \sum_{j=0}^{q_i-1} 2m'/2^{n_{\mathsf{bc}}} + \sum_{j=0}^{m'-1} 2q_i/2^{n_{\mathsf{bc}}} \leq 4m'q_i/2^{n_{\mathsf{bc}}}$, where $q_i \leq 2^{n_{\mathsf{bc}}-1}$ and $m' \leq 2^{n_{\mathsf{bc}}-1}$. Therefore, $\Pr[\text{Bad4} \mid \overline{\text{Bad1}} \cap \overline{\text{Bad2}} \cap \overline{\text{Bad3}}] \leq \sum_{i=1}^{\mathsf{Cnc}} 4m'q_i/2^{n_{\mathsf{bc}}} = 4m'q/2^{n_{\mathsf{bc}}}$ holds. We evaluate Theorem 1 by summing up the four probabilities of bad events. □

Tightness of Theorem 1. The bound in Theorem 1 is tight, as we have two matching attacks. We here present two distinguishing attacks to show the tightness of Theorem 1. The attacks try to invoke the events corresponding to Bad1 and Bad4 defined in the proof.

The first attack shows the tightness of the term $dq_L^2/2^{n_E+1}$, and it corresponds to Bad1. The attacker first repeats queries to LR4-L.$\mathcal{E}$ with the same plaintexts and distinct counters. With a sufficient number of queries, the attacker can find a key collision defined in Bad1 with a high probability by obtaining Leak other than $\perp$ or finding collisions of some ciphertexts. Once the attacker finds the key collision, it can distinguish two construction oracles LR4 and $\widetilde{\mathsf{P}}$ by querying twice to a non-leakage oracle with the same plaintext and the counters where the key collision occurs. If ciphertext collision occurs, the attacker figures out that it queries LR4 with a high probability; otherwise, it does that it queries $\widetilde{\mathsf{P}}$. This attack requires $q = O(1)$ and sufficiently large $q_L \approx O(2^{n_E/2})$ since the probability of the key collision is at most $dq_L^2/2^{n_E+1}$. Note that we can show the tightness of the term $dq^2/2^{n_E+1}$ in the same manner as the above attack. The attacker repeats construction queries with the same plaintexts and distinct counters. If ciphertext collision occurs in some counters, the attacker can distinguish two worlds by checking if other plaintexts also collide with the counters.

The second attack shows the tightness of the term $4m'q/2^{n_{\mathsf{bc}}}$, and it corresponds to Bad4. The attacker repeats construction queries and leakage queries with the same counters and distinct plaintext. If the attacker queries to the real world, the output ciphertexts cannot collide. However, if it queries to the ideal world, the probability of ciphertext collision is at most $4m'q/2^{n_{\mathsf{bc}}}$, as aforementioned.

4.3 Applications of LR4

As a primary application of fresh rekeying, Medwed *et al.* considered a challenge-response (CR) protocol with low-cost devices (*e.g.*, RFID) [MSGR10]. LR4 can be used for CR protocols, utilizing counters instead of random challenge values. A more practically useful application is LR-AE (*e.g.*, [DJS19, KS20, Men20, BGP$^+$19]). However, one line of research has presented various LR-AEs based on different leakage models for different security goals, utilizing different (LR and non-LR) primitives. Pinpointing how known LR-AEs can benefit from our proposal is not easy due to this wide variety of problem settings. In a very general sense, if an LR-AE scheme uses a nonce-based rekeying component which is assumed to be leak-free (*e.g.*. ISAP [DEM$^+$20]) it could be replaced with LR4 (but again it depends on the details of the scheme and needs ad-hoc security analysis).

Some examples. Given the aforementioned limitations in mind, we describe example applications of LR4 to existing LR-AEs in more detail. The first is the proposal of Mennink at Asiacrypt 2020 [Men20]. Mennink proposed a class of LR-AEs based on ΘCB (a TBC-based idealized version of OCB) [KR11]. He proposes to instantiate ΘCB by replacing the internal TBC with a fresh rekeying scheme, where the input value r of Figure 1 is used as a tweak. Mennink presented the black-box security of the proposal but did not clearly show what leakage-resilience security would be possible[3]. Still, the crucial point of his argument is that any TBC-based AE can use a fresh rekeying scheme as long as each encryption takes distinct tweaks determined by the nonce and the length of input variables (rather than the value of an input variable itself). If this holds and the nonce is a counter, each tweak is unique and determined incrementally, and we can use LR4 as the underlying TBC of ΘCB efficiently. As a result, the encryption of ΘCB does not leak anything from the TBC up to the bound of Theorem 3[4]. We should point out that Mennink's proposal does not strictly follow the rule mentioned above of tweak update in the processing of AD, as a TBC encrypts each AD block without taking a nonce (see *e.g.*, [Men20, Fig. 5]). However, this can be easily fixed by involving the nonce for each AD block encryption. This fix does not harm the security under the standard model, namely, without leakage. Unlike many existing LR-AEs, the resulting scheme is parallelizable and the rate is one; namely, it needs just one TBC (realized by LR4) call to process one input block, and thus it is quite efficient.

[3]The main purpose of [Men20] was to show the conceptual similarities between rekeying-based TBC modes and fresh rekeying schemes, not to present a concrete LR-AE.

[4]We also need to protect outside the TBC so that the mode does not leak anything. In the case of ΘCB, this is much cheaper than protecting the primitives because it consists of simple linear operations (XORs).

ΘCB has a relatively large state size due to its parallel structure. If we want to reduce the implementation size, serial counterparts such as PFB_plus and PFB$_\omega$ by Naito *et al.* [NSS20] could be used instead. These modes are specifically designed with (higher-order) masking implementations in mind. Romulus [IKMP20], a finalist of the NIST Lightweight cryptography project [Nat23], could also be used, with a similar modification to the AD processing as mentioned above to the Mennink's scheme. Meanwhile, some TBC-based LR-AEs do not follow the condition mentioned above on the tweak values used in encryption, such as HOMA [NSS22] and TEDT [BGP+19]. LR4 is not suitable for these because the tweak cannot be updated in an incremental manner. Designing efficient LR rekeying schemes (or TBCs) that suit these LR-AEs remains an interesting open problem.

The second example is FGHF [DJS19] or its improvement [KS20]. These LR-AEs are encryption-then-MAC composition, where the encryption and MAC function uses single-block leak-free PRFs. The first is directly replaceable with LR4 as it takes nonce N as an input (which will be a counter of LR4; the input block of E could be a fixed constant). The second leak-free PRF takes the (key-less) hash value V of the tuple of (ciphertext, nonce, associated data), and does not take a nonce N. Using LR4, this PRF can be modified so that it takes V as E's input and the next nonce $(N + 1)$ as the counter of LR4. We do not go into the details here, however, the security proof with leakage could be obtained in a similar manner to that of [KS20] (given certain restrictions on the decryption leakege imposed by m').

5 Quantitative success rate evaluation methodology for rekeying schemes

5.1 SCA backgrounds and success rate

Notations for SCA. We introduce notations for the discussion about SCA. An uppercase letter (*e.g.*, X) denotes a random variable/vector on a set denoted by the calligraphic character (*e.g.*, $\mathcal{X}$), and a lowercase character (*e.g.*, x) denotes an element of the set (*i.e.*, $x \in \mathcal{X}$), unless otherwise defined. Let Pr be the probability measure and p be the density or mass function. A side-channel trace is defined as $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^\ell$, where ℓ is the number of sample points. Let m be the number of traces available for an attack. Let $\boldsymbol{X}$ and T be the random variables for side-channel trace and n_b-bit partial plaintext/ciphertext, respectively. We suppose that m side-channel traces $\boldsymbol{X}^m = (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_m)$ and plaintexts/ciphertexts $T^m = (T_1, T_2, \ldots, T_m)$ are independent and identically distributed (i.i.d). A secret variable utilized in SCA is denoted by Z. If we need to specify a secret key k, we denote it by $Z^{(k)}$. For example, $Z^{(k)} = \mathrm{Sbox}(T \oplus k)$ and $n_b = 8$ for major AES software implementations.

Optimal SCA. In SCA on symmetric ciphers/primitives, we usually compute the rank of key candidates from side-channel traces and partial plaintexts/ciphertexts, and estimate the correct key according to the score. Let $S : \mathcal{K} \times \mathcal{X}^m \times \mathcal{T}^m \to \mathbb{R}$ be a score function and let $\delta_S : \mathcal{X}^m \times \mathcal{T}^m \to \mathcal{K}$ be an SCA distinguisher using S. For example, the correlation power analysis (CPA) utilizes the absolute value of Pearson's correlation coefficient as a score function, combined with a leakage function [BCO04]. Deep-learning based SCA (DL-SCA) utilizes a negative-log likelihood (NLL) calculated using a probability distribution represented by a neural network (NN) as a score function [MHM14, ZBHV19, ISUH21]. If a true probability distribution of a secret intermediate value Z given a side-channel trace $\boldsymbol{X}$, denoted by $p_{Z|\boldsymbol{X}}$, is available, it is proven to provide an optimal attack, which theoretically maximizes the success rate [HRG14, IUH21]. Concretely, a score function L

for a key candidate k defined as

$$L(k; \boldsymbol{X}^m, T^m) = -\frac{1}{m}\sum_{i=1}^{m} \log p_{Z|\boldsymbol{X}}(Z_i^{(k)} \mid \boldsymbol{X}_i)$$

is proven to provide an optimal attack, where $p_{Z|\boldsymbol{X}}$ denotes the true conditional probability distribution of Z given $\boldsymbol{X}$. In other words, $\delta_L(\boldsymbol{X}^m, T^m) = \arg\min_k L(k; \boldsymbol{X}^m, T^m)$ is an optimal distinguisher[5]. The function L is called NLL. For a given device, considering such an optimal attack is sufficient to evaluate the SCA resistance (and leakage resilience) against all possible SCAs.

Success rate. Success Rate (SR) has been commonly used for evaluating SCA performance and the validity of SCA countermeasures for various cryptographic implementations [SMY09]. SR given from m traces, denoted by SR_m, is defined as the probability that the rank of correct key k^* becomes one, as

$$\mathrm{SR}_m = \Pr\left[\mathrm{rank}(k^*, m) = 1\right], \tag{2}$$

where $\mathrm{rank}(k^*, m)$ denotes the correct key rank in an optimal SCA, defined as

$$\mathrm{rank}(k^*, m) = 1 + \sum_{k \in \mathcal{K} \setminus \{k^*\}} \mathbb{1}_{\{L(k^*) \geq L(k)\}}.$$

Here, $\mathbb{1}$ denotes the indicator function. Note that, for a simplified notation, we here omit the inputs of $\boldsymbol{X}^m$ and T^m to rank and L; therefore, rank and NLLs are random variables in this context.

A sound SCA countermeasure should exponentially increase the number of traces required to achieve an SR by an increase of security parameter(s). For example, masking schemes are proven to satisfy this property: the SR of SCA on masked implementations exponentially decreases by an increase of the masking order, which corresponds to the number of traces to achieve the SR, under a condition about mutual information [DFS15, IUH22a, MRS22].

SR upper-bound evaluation. As the true probability distribution is usually unknown and unavailable, it is quite difficult to evaluate the SR of an optimal SCA in practice. Currently, one of the most popular methods for evaluating an optimal SR would be to use DL-SCA: the evaluator profiles the device under test (*i.e.*, trains an NN to imitate the true probability distribution $p_{Z|\boldsymbol{X}}$) and repeats an attack with m traces using the trained NN to evaluate the SR_m empirically [ZBHV19, PHJ+19, dCGRP19]. However, an empirical approach like this incurs a non-negligible computational cost, and the soundness and validity of the evaluation result are sometimes uncertain due to NN approximation error, NN hyperparameter variations, and the stochastic aspects of learning. Alternatively, an inequality evaluation is sometimes useful for estimating the theoretically achievable SR from a quantity (*e.g.*, mutual information and SNR) for a given device/implementation, as stated in Theorem 2.

Theorem 2 (SR upper-bound [dCGRP19, IUH22a])**.** *Let $I(Z; \boldsymbol{X})$ be the mutual information between the secret intermediate value Z and side-channel trace $\boldsymbol{X}$. Let* SR_m *be the success rate of SCA with m traces.* SR_m *is upper-bounded as*

$$\xi(\mathrm{SR}_m) \leq m I(Z; \boldsymbol{X}), \tag{3}$$

[5]There are various constructions of optimal distinguisher [IUH22b]. For example, template attack [CRR02], which uses $p_{\boldsymbol{X}|T,K}$, is also optimal [HRG14]. Our discussion/methodology is valid regardless of the construction of optimal distinguishers, although we here focus on the optimal distinguisher based on $p_{Z|\boldsymbol{X}}$ for ease of explanation.

where $\xi(\mathrm{SR}_m)$ is a function $\xi : [0,1] \to \mathbb{R}_0^+$, defined as

$$\xi(\mathrm{SR}_m) = H(K) - (1 - \mathrm{SR}_m)\log(|\mathcal{K}| - 1) - H_2(\mathrm{SR}_m). \tag{4}$$

In Equation (4), $H(K)$ denotes the entropy of K, log *is the binary logarithm, and H_2 is the binary entropy function; namely, $H_2 : [0,1] \to [0,1] : p \mapsto -p\log p - (1-p)\log(1-p)$.*

Usually, it holds $H(K) = n_b$ and $|\mathcal{K}| = 2^{n_b}$, where n_b denotes the bit length of a partial secret key targeted by the SCA. Inequality (3) evaluates the SR of an optimal attack, and every SCA (on Z) must satisfy Inequality (3)[6]. Note that an SR upper-bound conversely represents the lower-bound of the number of traces required to achieve a given SR.

5.2 Formal analysis on success rate of SCA on LR4

5.2.1 Formalization

In the following, we consider the cache-based LR4. This section presents a methodology to evaluate the overall success rate in attacking LR4 (AR in short) in a quantitative manner. Our methodology is derived as a combination/unification of trace complexity bound and bounded leakage from underlying primitive(s) through an information-theoretical SR evaluation to quantify the attack cost and AR, whereas the existing studies utilize only the trace complexity bound as a threshold value for attack success/failure. For this purpose, we consider the bounded leakage as the mutual information $I(Z; \boldsymbol{X})$, which is a common leakage representation, as in many previous studies, and extend Inequality (3) for the numerical evaluation of attack cost/AR. This unification is essential for the practical usage of LR4 with a guarantee of quantitative security. Hereafter, we refer to the success rates of overall attack on LR4 and a partial key recovery as AR and SR, respectively.

In this paper, as a common case, we suppose that the rekeying components for LR4 have a construction similar to an AES-like block cipher; that is, its round function consists of n_s parallel evaluations of an n_b-bit Sbox for key-plaintext XOR (corresponding to SubBytes following AddRoundKey in AES). We also suppose that the LR4 instantiates rekeying components using an identical primitive. Recall that the d-th order LR4 consists of d rekeying components. with m-bounded traces (*i.e.*, rekeying interval of m). Here, the computations of the rekeying components under an m-bounded traces model (meaning rekeying interval of m) are performed for $\sum_{i=1}^{d} m^{i-1}$ different keys, as the i-th rekeying component generates m^i different temporal intermediate keys for the $(i+1)$-th rekeying component. This indicates that the attacker has m^{i-1} chances/trials for key-recovery SCA with m traces on the i-th rekeying component. It would be sufficient for the attacker to recover at least one key among all the SCA trials. Here, we consider AR as a probability that an attacker can achieve the full recovery of at least one intermediate/temporal key of a rekeying component by all possible SCA trials with m traces. Using the rank metric as well as SR, AR is formally defined as follows.

Definition 1 (Success rate of SCA on LR4)**.** *Let $\mathrm{AR}_{d,m}$ be the probability that an attacker succeeds in at least one full key recovery by attacking the d-th order m-bounded* LR4 *instantiated using rekeying component(s) with n_s parallel Sboxes. Using the rank metric, $\mathrm{AR}_{d,m}$ is defined by*

$$\mathrm{AR}_{d,m} = \Pr\left[\bigcup_{i=1}^{d}\bigcup_{j=1}^{m^{i-1}}\bigcap_{h=1}^{n_s} \mathrm{rank}(k^*_{i,j,h}, m) = 1\right], \tag{5}$$

[6]Let $\mathrm{SR}_m(\delta_S) = \Pr[k^* = \delta_S(\boldsymbol{X}^m, T^m)]$ be the SR of m-trace SCA with distinguisher δ_S (which is equivalent to Equation (2) with score function S). SR_m in Inequality (3) means $\mathrm{SR}_m = \sup_{\delta\in\mathcal{D}} \mathrm{SR}_m(\delta) = \mathrm{SR}_m(\delta_L)$, where $\mathcal{D}$ is a set of all possible SCA distinguishers.

where $k^*_{i,j,h}$ *denotes the* h*-th* n_b*-bit correct partial key of the* j*-th temporal key at the* i*-th rekeying component of* LR4.

In Definition 1, the right-most intersection $\bigcap_{h=1}^{n_s}$ means that the attacker should recover a full key from SCA on n_s parallel Sboxes; the center union $\bigcup_{j=1}^{m^{i-1}}$ means that the attacker can mount an m-bounded-trace SCA (*i.e.*, a trial) on the i-th rekeying component m^{i-1} times with different keys; and the left-most union $\bigcup_{i=1}^{d}$ means that the attacker can mount the trial m^{i-1} times for d different rekeying components. It is sufficient for the attacker to succeed in at least one full-key recovery among all trials. Hence, we should take the union of events $\bigcap_{h=1}^{n_s} \mathrm{rank}(k^*_{i,j,h}, m) = 1$ in terms of i and j, whereas the full-key recovery of a trial is represented as the intersection in terms of h.

Remark 2 (On payload encryption). LR4 generates m^d temporal keys, and we call the payload encryption $m' \times m^d$ times in total. Definition 1 considers SCAs on rekeying components, excluding the payload encryption. If we use a primitive for the payload encryption same as the rekeying component (implying that $m = m'$), it is sufficient for the evaluation including the payload encryption calls to consider $\mathrm{AR}_{d,m}$, as in the numerical evaluation in Section 6.1. Even if we use a distinct primitive for the payload component, we can readily evaluate its AR by an union of full-key recovery events for the primitive involved in m' in a similar manner, which is extendable to Theorem 3 below.

Remark 3 (Relation to multi-user security). Our definition is similar to the security notion in the multi-user encryption setting [DLMS14, BT16, LMP17, HTT18, DGGP21, NSSY22], which has been used to determine the rekeying frequency in real-world cryptographic protocols such as (D)TLS and QUIC [Res18, RTM18, TT21]. Security analysis of LR4 is related to a cryptanalysis in the multi-user setting. This is because we perform multiple rekeying component evaluations using different keys, which would correspond to the case that multiple users evaluate a block cipher with distinct keys. Note that, in attacking LR4, rekeying component outputs are available only through leakage, different from common cryptanalyses. The numbers of queries and users correspond to the trace bound m and key lifetime (or the number of SCA trials) described below as $\sigma_{d,m}$, respectively. In other words, the above definition and the following theorem(s) are used to evaluate the success rate (or advantage) and the rekeying frequency in a setting similar to multi-user encryption with an SCA leakage.

5.2.2 Information-theoretical evaluation

We next formally provide the relation between AR and SR to derive a concrete and quantitative AR evaluation method, under some standard and realistic assumptions.

Lemma 1 (Relation between AR and SR). *Let* $\mathrm{AR}_{d,m}$ *be the overall success rate of SCA on the* d*-th order* m*-bounded* LR4. *Suppose that all temporal keys are mutually independent. Suppose that, for rekeying components, the SR of SCA on* n_b*-bit partial key recovery is identical for all Sboxes/partial keys. Let* SR_m *be the partial key recovery success rate of an SCA with* m *traces. It holds that*

$$\mathrm{AR}_{d,m} = 1 - (1 - (\mathrm{SR}_m)^{n_s})^{\sigma_{d,m}}, \tag{6}$$

$$\mathrm{SR}_m = \left(1 - (1 - \mathrm{AR}_{d,m})^{1/\sigma_{d,m}}\right)^{1/n_s}, \tag{7}$$

where $\sigma_{d,m} = \sum_{i=1}^{d} m^{i-1}$ *denotes the number of SCA trials available for the attacker.*

Proof. We first show that Equation (6) holds. Let $(\Omega, \mathcal{F}, \Pr)$ be a probability space. Let $[A]^c$ denote the complement of a set $A \in \mathcal{F}$ (*i.e.*, $[A]^c = \Omega \setminus A$). Let $A^m_{i,j,h}$ denote an event of $\mathrm{rank}(k^*_{i,j,h}, m) = 1$. According to De Morgan's law, Equation (5) is transformed into

$$\begin{aligned}
\mathrm{AR}_{d,m} &= \Pr\left[\left[\bigcap_{i=1}^{d}\left[\bigcup_{j=1}^{m^{i-1}}\bigcap_{h=1}^{n_s} A^m_{i,j,h}\right]^c\right]^c\right] \\
&= \Pr[\Omega] - \Pr\left[\bigcap_{i=1}^{d}\bigcap_{j=1}^{m^{i-1}}\left[\bigcap_{h=1}^{n_s} A^m_{i,j,h}\right]^c\right]. \qquad (8)
\end{aligned}$$

Here, events of $\bigcap_{h=1}^{n_s} A^m_{i,j,h}$ (*i.e.*, success of full-bit temporal key recovery in a trial) for all i and j are mutually independent, as temporal keys are supposed to be mutually independent owing to RO. In addition, the rekeying components are performed on an identical device, which indicates that the events $\bigcap_{h=1}^{n_s} A^m_{i,j,h}$ are i.i.d in terms of i and j. Therefore, Equation (8) is followed by

$$\begin{aligned}
\mathrm{AR}_{d,m} &= \Pr[\Omega] - \prod_{i=1}^{d}\prod_{j=1}^{m^{i-1}} \Pr\left[\left[\bigcap_{h=1}^{n_s} A^m_{i,j,h}\right]^c\right] \\
&= \Pr[\Omega] - \prod_{i=1}^{d}\prod_{j=1}^{m^{i-1}}\left(\Pr[\Omega] - \Pr\left[\bigcap_{h=1}^{n_s} A^m_{i,j,h}\right]\right).
\end{aligned}$$

Due to the assumption on SR of SCA on n_b-bit partial key recovery, events $A^m_{i,j,h}$ for all h are also mutually independent, which indicates that it holds

$$\Pr\left[\bigcap_{h=1}^{n_s} A^m_{i,j,h}\right] = \prod_{h=1}^{n_s} \Pr\left[A^m_{i,j,h}\right]$$

for any i and j. In addition, owing to the assumptions, we consider an identical SR for all i, j, and h because $\mathrm{SR}_m = \Pr[A^m_{i,j,h}]$ as well as Equation (2). Thus, we conclude

$$\begin{aligned}
\mathrm{AR}_{d,m} &= \Pr[\Omega] - \prod_{i=1}^{d}\prod_{j=1}^{m^{i-1}}\left(\Pr[\Omega] - \prod_{h=1}^{n_s} \Pr\left[A^m_{i,j,h}\right]\right) \\
&= 1 - \prod_{i=1}^{d}\prod_{j=1}^{m^{i-1}}\left(1 - \prod_{h=1}^{n_s} \mathrm{SR}_m\right) \\
&= 1 - \left(1 - (\mathrm{SR}_m)^{n_s}\right)^{\sigma_{d,m}},
\end{aligned}$$

as required. Equation (7) is derived from Equation (6) as

$$\begin{aligned}
1 - \left(1 - (\mathrm{SR}_m)^{n_s}\right)^{\sigma_{d,m}} = \mathrm{AR}_{d,m} \quad &\Leftrightarrow \quad \left(1 - (\mathrm{SR}_m)^{n_s}\right)^{\sigma_{d,m}} = 1 - \mathrm{AR}_{d,m} \\
&\Leftrightarrow \quad 1 - (\mathrm{SR}_m)^{n_s} = (1 - \mathrm{AR}_{d,m})^{1/\sigma_{d,m}} \\
&\Leftrightarrow \quad (\mathrm{SR}_m)^{n_s} = 1 - (1 - \mathrm{AR}_{d,m})^{1/\sigma_{d,m}},
\end{aligned}$$

and finally we conclude

$$\mathrm{SR}_m = \left(1 - (1 - \mathrm{AR}_{d,m})^{1/\sigma_{d,m}}\right)^{1/n_s},$$

as required. □

Lemma 1 states how much/little SR is required to achieve an AR, and Equation (7) is used for deriving the SR corresponding to a given AR. In designing a cryptographic module with SCA countermeasure(s), an acceptable AR is determined in advance. For LR4, the parameters (rekeying order d and interval m) should be determined for a required AR and given device with mutual information $I(Z; \boldsymbol{X})$ as a leakage amplitude (or an SNR, which upper-bounds $I(Z; \boldsymbol{X})$ *via* the Shannon–Hartley theorem). For the evaluation, we introduce an upper-bound of AR as Theorem 3.

Theorem 3 (AR upper-bound)**.** *Let* $\mathrm{AR}_{d,m}$ *be the success rate of* m*-bounded-trace attack on the* d*-th order* LR4*, as in Definition 1. Let* $I(Z; \boldsymbol{X})$ *be the mutual information between the secret intermediate value* Z *and side-channel trace* $\boldsymbol{X}$*. With the same assumption as Lemma 1, the* $\mathrm{AR}_{d,m}$ *is upper-bounded as*

$$\xi\left(\left(1-(1-\mathrm{AR}_{d,m})^{1/\sigma_{d,m}}\right)^{1/n_s}\right) \leq mI(Z; \boldsymbol{X}), \tag{9}$$

where $\sigma_{d,m} = \sum_{i=1}^{d} m^{i-1}$ *and* ξ *is the function defined as Equation (4) in Theorem 2.*

Proof. It is obvious from Inequality (3) in Theorem 2 and Equation (7) in Lemma 1. □

Theorem 3 states the relation between AR with m-bounded traces and the mutual information $I(Z; \boldsymbol{X})$ considered as a leakage from each rekeying component; thus, this is a unified security metric of the bounded trace model and underlying primitive leakage. Using Theorem 3, we can determine the security parameters d and m for a required AR and given device with $I(Z; \boldsymbol{X})$ or SNR.

Meanings of Theorem 3. According to [dCGRP19,IUH22b,IUH22a], function ξ represents the number of bits required to achieve an SR. For example, $\mathrm{SR} = 1/2^{n_b}$ implies that the attacker has no advantage in the attack, as represented by by $\xi(1/2^{n_b}) = 0$. Conversely, $\mathrm{SR} = 1$ implies that the attacker obtains the full-bit information of a secret key, as represented by $\xi(1) = n_b$. In contrast, $mI(Z; \boldsymbol{X})$ represents the amount of information that the attacker receives through m traces. Note that $n_s I(Z; \boldsymbol{X}) = \lambda$, where λ is the bounded leakage and n_s denotes the number of parallel S-boxes, and m means the traces bound. Thus, Theorem 3 reveals the relation between the bounded leakage function and trace complexity bound in an analytical and quantitative manner. In practice, Inequality (9) evaluates an upper-bound of SR_m to achieve a given $\mathrm{AR}_{d,m}$ (*i.e.*, attack cost), to determine the appropriate trace bound m (*i.e.*, the rekeying interval).

Remark 4 (On SR range). In Inequality (9), SR_m is defined in the range of $[0, 1]$. However, the minimum value of SR_m should be $1/2^{n_b}$, which means that the attacker has no advantage in guessing the secret key. In other words, it makes no sense to consider the case that $\mathrm{SR}_m \in [0, 1/2^{n_b})$, because any attacker trivially achieves $\mathrm{SR}_m = 1/2^{n_b}$ by a random guess. Therefore, we should determine d and m such that they satisfy $\mathrm{SR}_m \geq 1/2^{n_b}$ in addition to Inequality (9). Conversely, if $\mathrm{SR}_m \geq 1/2^{n_b}$ is not achievable for a given AR and $I(Z; \boldsymbol{X})$, such AR cannot be reached by the device.

Remark 5 (Relation between Theorem 1 and Theorem 3). Theorem 1 proves the security bound against an SCA attacker under a theoretical (yet reflecting the idea of practical protection methods, *e.g.*, high-order masking) leakage model with idealized primitives. In contrast, Theorem 3 states a bound of overall success rate of an optimal and real SCA on actual symmetric primitive(s). Each theorem captures different and essential aspects of LR4.

5.3 Practical usage of LR4

We can utilize LR4 as an SCA countermeasure with a guarantee of quantitative security evaluated by our methodology in Section 5.2. The proposed design flow is as follows.

Step 1: Device profiling. We first need to know the value of mutual information $I(Z;\boldsymbol{X})$ or achievable SNR of side-channel measurement by profiling the target device. For example, a deep-learning based profiling method in [IUH22a, Section 6] is useful to evaluate a tight upper-bound of $I(Z;\boldsymbol{X})$. In addition, according to the Shannon–Hartley theorem, $I(Z;\boldsymbol{X})$ is upper-bounded by SNR as $I(Z;\boldsymbol{X}) \leq 1/2\log(1+\mathrm{SNR})$ (assuming that noise is additive Gaussian). This indicates that it would be sufficient to evaluate the SNR, which may be easier than $I(Z;\boldsymbol{X})$ evaluation.

Step 2: Determination of master key lifetime and acceptable key recovery success rate. In this paper, we define the master key lifetime as the number of temporal keys generated under a given $I(Z;\boldsymbol{X})$ or SNR. The master key lifetime should be considered with the number of calls the target cipher or LR-AE required for the application. At the same time, we determine an acceptable full-key recovery success rate as a threshold value of $\mathrm{AR} \in (0,1]$.

Step 3: Determination of security parameters. We then determine the security parameters including the rekeying order d and rekeying interval m (*i.e.*, appropriate trace complexity bound for the situation) using Theorem 3 such that, for a given AR, the key lifetime exceeds the desired value determined in Step 2. Namely, for a given AR, we should determine d and the corresponding maximum value of m with satisfying Inequality (9) and $\mathrm{SR}_m \geq 1/2^{n_b}$ such that the master key lifetime requirement is met. If the requirement cannot be met with practical d and m (see also Remark 4), we need to mitigate/reduce the leakage by other SCA countermeasures such as masking and hiding.

Here, if we adopt a masking scheme, we do not have to profile masking gadgets because we can precisely estimate the resulting leakage from masked implementation using the aforementioned profiling method in [IUH22a]. Alternatively, under some conditions (see [IUH22a]), we can use another inequality instead of Inequality (9) in the following corollary:

Corollary 1. *Let e be the masking order. Let $I(S;\boldsymbol{L})$ be the mutual information between a masking share S and its corresponding leakage $\boldsymbol{L}$. It holds*

$$\xi\left(\left(1-(1-\mathrm{AR}_{d,m})^{1/\sigma_{d,m}}\right)^{1/n_s}\right) \leq m\log\left(1+(2^{n_b}-1)\left(2\ln(2)I(S;\boldsymbol{L})\right)^{e+1}\right), \quad (10)$$

where $\log$ *and* $\ln$ *are the binary and natural logarithms, respectively.*

Proof. It is proven by combining Theorem 3 and a lemma in [IUH22a, MRS22] stating that $I(Z;\boldsymbol{X}) \leq \log\left(1+(2^{n_b}-1)\left(2\ln(2)I(S;\boldsymbol{L})\right)^{e+1}\right)$. □

Here, $I(S;\boldsymbol{L})$ is equal to the $I(Z;\boldsymbol{X})$ of non-masked implementation in some settings; therefore, Inequality (10) can be used to evaluate the AR on masked implementation from the profiling result on non-masked implementation, without actual evaluation on masking gadgets [IUH22a].

Remark 6 (Conditions for Inequality (10)). Inequality (10) is meaningful for a non-trivially low $I(S;\boldsymbol{L})$ (*i.e.*, worse SNR) and/or large masking order e, as mentioned in [IUH22a, Remark 5.1]. At least, it should hold $I(S;\boldsymbol{L}) < 2\ln(2) \approx 0.72$ to use Inequality (10). If $I(S;\boldsymbol{L})$ is relatively high and e is relatively small, we need to actually profile the adopted masking gadgets or to use the aforementioned method in [IUH22a]. It should be noted that Béguinot *et al.* recently proved another bound in [BCG+23], in which they claim a more precise evaluation than [IUH22a, MRS22]. It would be useful for the practical and more precise evaluation, although we used Corollary 1 based on [IUH22a, MRS22] for the proof-of-concept evaluation in this paper. In other words, for masked implementation, we can achieve a more precise evaluation if we use a precise inequality about masked implementations.

Step 4: Actual design/implementation. After determining the parameters that satisfy the master key lifetime requirement, we conduct an actual design and implementation for LR4. Here, if we adopt no SCA countermeasure other than LR4, it is sufficient to use a common non-protected implementation like naïve or reference implementations as it is, which may have been used in Step 1. Otherwise, a sound masked implementation should be utilized. The masking scheme used here should be provably secure under a practical leakage model (*e.g.*, [NRS11, RBN+15, GMK16, GM17, BBD+16]), and implementation should be done by carefully considering the physical defaults that cause security order degradation (*e.g.*, coupling, cross-share interaction, and glitches), which have been shown and analyzed in many studies [RSVC+11, BGG+14, dCBG+17, DCEM18, FGP+18, GMPO20, SCS+21, SSB+21, MKSM22]. Usage of leakage detection/verification tools, design automation tools and/or open-source implementations is promising to achieve such a provably secure masked implementation (*e.g.*, [Rep16, UHMA17, UHMA21, BBC+19, KSM20, SCS+21, SSB+21, KMMS22, BMRT22]).

6 Validity evaluation

6.1 Numerical evaluation

We show the validity of LR4 through a numerical evaluation of the key lifetime for a given d and mutual information.

First, we virtually determine the mutual information in Step 1. We set the acceptable full-key recovery success rate as 1% as an example[7] and then evaluate the master key lifetime for various rekeying orders d (and masking orders e) using Inequality (9) (or Inequality (10)) with achievable trace bound m for $\mathrm{AR}_{d,m} = 0.01$. Here, we assume that each rekeying component in LR4 is implemented using one AES encryption call (namely, $G_i(k_i, \cdot) = E_{k_i}(\cdot)$ where E is AES for any i) for a proof-of-concept evaluation, although such a plain AES encryption is not an RO. See Section 6.2 for a discussion about the instantiation of an RO using AES or other symmetric primitives. Note that the evaluation result under the assumption of one AES encryption call would be consistent with actual RO instantiations. In addition, we suppose that AES is utilized for encrypting payload data using a temporal key generated by LR4. The AES encryption for payloads should be trace-bounded by m, so the key lifetime is given by $m \times m^d = m^{d+1}$, as m^d corresponds to the number of generatable temporal keys.

Table 1 and Table 2 list the evaluation results of key lifetime lower-bounds of LR4 with non-masked and masked implementations, respectively, where $I(Z; \boldsymbol{X})$ is the virtually determined mutual information value; SR denotes the SR required to satisfy $\mathrm{AR}_{d,m} = 0.01$ with a maximum value of m for a given d (evaluated using Equation (7)); m denotes the maximum value of m under the conditions evaluated using Inequalities (9) and (10) for non-masked and masked implementations, respectively; and m^{d+1} denotes a lower-bound of the maximum number of secure encryption calls with generated temporal keys. "N/A" means that we cannot derive the value due to the computational difficulty (that is, the evaluation requires extremely high-precision floating-point arithmetic). Note that rekeying and masking are not applied if d and e are zero, respectively (if $d = 0$, the master key is used for the payload encryption as it is).

The results demonstrate the validity of LR4 as an SCA countermeasure: the key lifetime increases exponentially (*i.e.*, digit-wisely) by an increase of the rekeying order d in most parts of the tables. In addition, from Table 2, we confirm that a combination with masking is more effective for improving key lifetimes if $I(Z; \boldsymbol{X})$ is sufficiently smaller (*i.e.*, the leakage is sufficiently noisy). It should be noted that, in $I(Z; \boldsymbol{X}) = 0.01$ of Table 2, the

[7]Even for $\mathrm{AR} < 0.01$, the resulting key lifetime is not very different. Empirically, the key lifetime seems to be linear to $\log \mathrm{AR}$ in our experiment.

Table 1: Key lifetime lower-bound evaluation results of non-masked LR4 for $\mathrm{AR}_{d,m} = 0.01$

d	$I(Z;\boldsymbol{X}) = 1$			$I(Z;\boldsymbol{X}) = 0.1$			$I(Z;\boldsymbol{X}) = 0.01$		
	SR	m	m^{d+1}	SR	m	m^{d+1}	SR	m	m^{d+1}
0	0.75	6	**6**	0.75	52	**52**	0.75	519	**519**
1	0.67	5	**25**	0.60	39	**1,521**	0.52	319	**101,761**
2	0.61	5	**125**	0.49	30	**27,000**	0.38	212	**9,528,128**
3	0.57	4	**256**	0.41	24	**331,776**	0.29	149	**492,884,401**
4	0.52	4	**1,024**	0.35	20	**3,200,000**	0.23	109	**15,386,239,549**

d	$I(Z;\boldsymbol{X}) = 0.001$			$I(Z;\boldsymbol{X}) = 0.0001$		
	SR	m	m^{d+1}	SR	m	m^{d+1}
0	0.75	5,190	**5,190**	0.75	51,892	**51,892**
1	0.46	2,673	**7,144,929**	0.40	22,405	**501,984,025**
2	0.30	1,524	**3,539,605,824**	0.23	10,949	**1,312,572,700,349**
3	0.21	932	**754,507,653,376**	0.15	5,821	**1,148,128,234,489,681**
4	0.15	603	**79,723,537,443,243**	0.10	3,393	**449,696,283,350,000,192**

Table 2: Key lifetime lower-bound evaluation results of masked LR4 for $\mathrm{AR}_{d,m} = 0.01$

d	e	$I(S;\boldsymbol{L}) = 0.01$			$I(S;\boldsymbol{L}) = 0.001$			$I(S;\boldsymbol{L}) = 0.0001$		
		SR	m	m^{d+1}	SR	m	m^{d+1}	SR	m	m^{d+1}
0	0	0.75	519	**519**	0.75	5 K	**5 K**	0.75	51 K	**51 K**
	1	0.75	76	**76**	0.75	7 K	**7 K**	0.75	733 K	**734 K**
	2	0.75	5 K	**5 K**	0.75	5 M	**5 M**	0.75	5 G	**5 G**
1	0	0.52	319	**101 K**	0.46	2 K	**7 M**	0.40	22 K	**501 M**
	1	0.58	54	**2 K**	0.45	3 K	**13 M**	0.34	258 K	**66 G**
	2	0.46	2 K	**7 M**	0.31	1 M	**2 T**	0.23	938 M	**881 P**
2	0	0.38	212	**9 M**	0.30	1 K	**3 M**	0.23	10 K	**1 T**
	1	0.47	41	**68 K**	0.29	2 K	**8 G**	0.18	105 K	**1 P**
	2	0.30	1 K	**3 G**	0.14	569 K	**184 P**	N/A	N/A	**N/A**
3	0	0.29	149	**492 M**	0.21	932	**754 G**	0.15	5 K	**1 P**
	1	0.39	32	**11 M**	0.20	1 K	**2 T**	0.10	47 K	**5 E**
	2	0.21	948	**858 G**	N/A	N/A	**N/A**	N/A	N/A	**N/A**

key lifetime of first-order masked implementation is smaller than non-masked one. This is because Inequality (10) evaluates a *lower-bound* of key lifetime for a given $\mathrm{AR}_{d,m}$, but does not necessarily precisely/tightly represent an actual value under some conditions. As mentioned in Remark 6, Inequality (10) cannot provide a meaningful evaluation of the bound if $I(S;\boldsymbol{L})$ is too low and e is too small. $I(S;\boldsymbol{L}) = 0.01$ and $e = 1$ may be such a condition, but an actual key lifetime would be longer than the non-masked implementation.

6.2 Discussion

6.2.1 Practical instantiations

Standard instantation using SHA-3. The provable security analysis of LR4 assumes that G_i is an RO and E is an IC. A straightforward instantiations would be *e.g.*, SHA-3 for G (as $G_i(k_i, \mathsf{ctr}_i) = \text{SHA-3}(k_i \parallel \mathsf{ctr}_i \parallel i)$) and AES for E. If one wants to avoid multiple distinct primitives for implementation efficiency, E can be also permutation-based, say using Keccak-p with an adequate domain separation and output truncation (but the resulting function is non-invertible so it limits the applications). In principle, instead of E we can use more complex functions, such as nonce-based encryption or AE, possibly using a permutation. What security/efficiency benefit is expected depends on the scheme we use, and exploring such combinations would be an interesting future direction.

Instantiation using AES. Our SR evaluation in Section 6.1 assumes a naïve use of AES for G for ease of evaluation, but this instantiation has a gap from the proof. It is important to consider secure instantiations using AES owing to its ubiquity and maturity as a symmetric primitive. We briefly discuss secure instantiations of E and G based solely on an ideal cipher E_{base}.

For simplicity, let E_{base} have a key length one-bit longer than E so that we can generate two independent ideal ciphers, E' and E'', from E_{base} by using this extra key bit for domain separation. The problem is how to instantiate G from E'. What we need for G is indifferentiability [MRH04] from the fixed-length RO. Note that classical block cipher-based compression functions (*e.g.*, Davies–Mayer), are not indifferentiable [KM07]. We present two secure examples here. First, if E' is a block cipher of n-bit block and n-bit key, we can instantiate G by Mennink's F^3 construction [Men17], which is $n/2$-bit indifferentiability from the (fixed-length) RO and needs three calls of E'. Second, if E' is a block cipher of n-bit block and $2n$-bit key, G can be a $2n$-bit indifferentiable hash function using Hirose's double-block-length compression function [Hir06] with a proper domain extension. For example, we can use MDPH [Nai19], which has $(n - \log n)$-bit indifferentiability [GIM22]. Both examples utilize different primitives and offer different security levels, so the proper choice will depend on the security goal, application, and the available primitives.

We should point out that the average computation cost of LR4 is at most two G calls plus one E call (Section 3), which means the impact of G's cost on the total computation is limited. In addition, security evaluation for the aforementioned (secure) instantiations could be done in the same manner as Section 6.1 as long as we use AES as E_{base}.

6.2.2 Conditions for exponential increase of key lifetime.

Interestingly, in Table 1 and Table 2, SR and the trace bound m get more severe for larger d. For larger d, the attacker can have the larger number of SCA trials (*i.e.*, $\sigma_{d,m}$ increases), which enables at least one full-key recovery with a smaller value of SR. This is also represented as Equation (7), which shows a monotonic decrease in terms of $\sigma_{d,m}$ for a fixed $\mathrm{AR}_{d,m}$. In other words, the key lifetime gets longer by increasing d only if the gain of m^{d+1} is greater than the decrease of SR and m. In fact, if d is very large, the key lifetime does not (exponentially) increase and AR decreases anymore by increasing d, which implies that LR4 is valid as an SCA countermeasure for sufficiently small d.

In contrast, the security of masking is guaranteed as the SR decreases exponentially by increasing the masking order e. In particular, the security of masking is *asymptotically* proven; that is, it holds $\mathrm{SR}_m \to 1/2^{n_b}$ as $e \to \infty$, although the exponential increase may not be guaranteed for a small e [IUH22a]. Thus, the rekeying and masking have opposite features to each other. A higher-order SCA countermeasure usually incurs a large performance overhead. If LR4 is available, its adoption can be one of the best choices to counter SCAs, as it is very efficient for small d and its overhead is practically small (as shown in Section 3).

6.2.3 On upper-bound of master key lifetime for general rekeying schemes

In the above, we mentioned that the master key lifetime does not increase exponentially by increasing d if d is (very) large, although it is effective for practical values of d. In general, the increasing rate gets slower for larger d, and the increase of master key lifetime eventually stops for a certain value of d, depending on $I(Z; \boldsymbol{X})$. Intuitively, this is because the minimum value of SR_m should be $1/2^{n_b}$ (as mentioned in Remark 4), although SR_m is monotonically decreasing in terms of d and $\mathrm{SR}_m \in [0, 1]$ to the definition. In fact, any rekeying scheme including LR4 has an upper-bound of master key lifetime according to Proposition 2.

Proposition 2 (Attacker with infinite trials *almost surely* succeeds in at least one full-key recovery)**.** *Let* $\mathrm{TR}_{\sigma,m}$ *be the probability of at least one success during* σ *SCA trials with* m*-bounded-trace, defined as*

$$\mathrm{TR}_{\sigma,m} = \Pr\left[\bigcup_{v=1}^{\sigma} \bigcap_{h=1}^{n_s} \mathrm{rank}(k^*_{v,h}, m) = 1\right],$$

where $\mathrm{rank}(k^*_{v,h}, m)$ *denotes the correct key rank of the h-th partial key at the v-th trial with m traces. With the same assumption as Lemma 1, it holds* $\mathrm{TR}_{\sigma,m} \to 1$ *as* $\sigma \to \infty$*; namely,*

$$\Pr\left[\lim_{\sigma\to\infty} \bigcup_{v=1}^{\sigma} \bigcap_{h=1}^{n_s} \mathrm{rank}(k^*_{v,h}, m) = 1\right] = 1,$$

if $\mathrm{SR}_m \neq 0$.

Proof. If $\mathrm{SR}_m \neq 0$, then it holds $\Pr[\bigcap_{h=1}^{n_s} \mathrm{rank}(k^*_{v,h}, m) = 1] = (\mathrm{SR}_m)^{n_s} > 0$ for any v, according to the assumption on the SR. Therefore, as the SCA trials are mutually independent of each other, it holds

$$\sum_{v=1}^{\infty} \Pr\left[\bigcap_{h=1}^{n_s} \mathrm{rank}(k^*_{v,h}, m) = 1\right] = \infty. \tag{11}$$

According to the Borel–Cantelli lemma [Fel91, pp. 201–202], Equation (11) is followed by

$$\Pr\left[\limsup_{v\to\infty} \bigcap_{h=1}^{n_s} \mathrm{rank}(k^*_{v,h}, m) = 1\right] = 1,$$

which means that events $\bigcap_{h=1}^{n_s} \mathrm{rank}(k^*_{v,h}, m) = 1$ (*i.e.*, successful full-key recoveries) *infinitely often* occur with probability one. This implies Proposition 2. □

Corollary 2. *Let* $\mathrm{AR}_{d,m}$ *be the success rate of attack on the d-th order m-bounded-trace* LR4 *defined in Definition 1. With the same assumption as Lemma 1, it holds* $\mathrm{AR}_{d,m} \to 1$ *as* $d \to \infty$ *or* $m \to \infty$.

Proof. It is proven[8] by Proposition 2 as the case that $\mathrm{TR}_{\sigma,m} = \mathrm{AR}_{d,m}$ with $\sigma = \sigma_{d,m}$, where it holds $\sigma_{d,m} \to \infty$ as $d \to \infty$ or $m \to \infty$. Note that $\mathrm{SR}_m \to 1$ as $m \to \infty$ usually holds. □

Proposition 2 implies that, for a given TR value τ, there always exists the number of SCA trials σ such that $\mathrm{TR}_\sigma \geq \tau$, and the implementation cannot achieve a master key lifetime of more than σ with regard to the overall success rate of τ. In other words, Proposition 2 implies that, for any given AR, we should make SR_m approach to zero when $d \to \infty$, although SR_m should be greater than $1/2^{n_b}$. Here, the master key life time $\sigma_{d,m}$ is maximized by d and m which tightly satisfies Inequality (9) with $\mathrm{SR}_m = 1/2^n$. Thus, for a given $I(Z; \boldsymbol{X})$ (or $I(S; \boldsymbol{L})$ and masking order) and AR, there exists an upper-bound of the master key lifetime, distinctly from a brute-force/cryptanalysis on the master key. Theorem 3 is an upper-bound of the master key lifetime with regard to SCA[9], and it is a case study of LR4. Our discussion emphasizes a (trivial) fact that *an ultimate goal of SCA countermeasures including rekeying is to achieve a (master) key lifetime as long as lifetime against pure cryptanalysis, such that SCA leakage is no longer useful for the attacker.*

Related to Proposition 2, the convergence rate of TR $\to 1$ is very important and represents the achievable security by rekeying schemes. It depends on the value of $I(Z; \boldsymbol{X})$. Fortunately, for LR4, we experimentally confirmed that the convergence is slow for practical d and a wide range of $I(Z; \boldsymbol{X})$, which indicates the validity of LR4 security for many

[8]Note that the proof of Lemma 1 does *not* include the case where $d \to \infty$ or $m \to \infty$ as the proof of Proposition 2 requires an interchange between Pr and lim, which indicates that we *cannot* conclude $\mathrm{AR}_{d,m} \to 1$ as $d \to \infty$ or $m \to \infty$ from Equation (6).

[9]Abdalla and Bellare also discussed the key lifetime in [AB00]. Their work considered neither the side-channel adversary nor leakage resilience, whereas this paper focuses on the rekeying security in the presence of leakage.

practical conditions. In contrast, for very high $I(Z; \boldsymbol{X})$ (*e.g.*, $I(Z; \boldsymbol{X}) = 1$), the leakage is not sufficiently bounded at all, and it is impossible for any SCA countermeasure to protect such a leaky device (see also [BS21]). Discussion on the convergence rate for a wider range of $I(Z; \boldsymbol{X})$ would be useful for achievable security of rekeying schemes. The purpose/goal of the rekeying scheme is to improve master key lifetime in general; hence, investigating (the existence of) tighter upper-bounds and the convergence rate for rekeying schemes is an important future work for making the rekeying security more concrete.

6.2.4 Resilience against fault attacks

We here briefly discuss the resilience of LR4 against fault attacks [BDL97]. A major fault attack would be differential fault analysis (DFA) [BS97]. DFA recovers the secret key from pair(s) of correct and faulty ciphertexts for an identical input, where faulty ciphertext means that an error (*e.g.*, bit flip) is induced to an intermediate value of the encryption/decryption. For example, in the case of AES encryption, one-bit fault in the eighth-round input is sufficient for the full-key recovery if the corresponding correct ciphertext is available [PQ03]. However, the DFA attacker should observe the output of the symmetric primitive to obtain the pair(s) of correct and faulty ciphertexts. As the output of ROs of LR4 is not (directly) available for the attacker, DFA is inapplicable to LR4 implementation (except for the payload encryption). In addition, DFA must require to query an identical plaintext twice to obtain pair of correct and faulty ciphertexts. If the LR4 is correctly implemented in such a way as to detect the replayed queries as mentioned in Section 3.2 (and the payload encryption is nonce-based), ROs (and payload encryption) never evaluate an identical input more than once, which also indicates the inapplicability of DFA.

Some other fault attacks have been also developed, such as fault sensitivity analysis (FSA) [LSG+10, MMP+11], differential fault intensity analysis (DIFA) [GYTS14], and persistent fault analysis (PFA) [ZLZ+18, ZHF+23]. These fault attacks utilize a statistical mean for the key recovery like DPAs. Hence, LR4 can offer a leakage resilience by determining an appropriate trace bound m and m' similarly to Section 5 (which may be trivial for some attacks). Moreover, we want to stress that the FSA requires to query an identical plaintext many times to observe the leakage of fault sensitivity; the DFIA utilizes the output ciphertext; in addition, PFA is a chosen-plaintext fault attack. Neither such chosen-plaintext strategies nor RO outputs are avaialble in attacking ROs in LR4; thus, it would be difficult to apply these fault attacks to ROs in LR4 (in a naïve manner). Meanwhile, the payload encryption part can be protected using an appropriate trace bound m', a fault attack resilient mode of operation, and/or countermeasure against fault attacks (*e.g.*, fault detection schemes).

Note that fault attacks on hash functions (*e.g.*, SHA-3), which may be a natural choice for RO instantiation, would be frequently more difficult than block ciphers, although we basically discuss fault attacks on AES in this section.

7 Relation to existing LR Schemes

7.1 Comparison to LR-PRG/stream cipher

LR-PRG and stream cipher may be used for generating a key stream as a temporal key. For example, Pietrzak's LR stream cipher [Pie09] is based on a weak PRF F (whose outputs are pseudorandom as long as inputs are random) and its initial state is one uniformly random input of F, in addition to two keys of F which are the master key, and the random input is sent in clear. F is called in an alternating manner, using an internal state consisting of two keys and one input to F. Leakage model is different from ours, namely it is assumed that F leaks a certain amount of bits for each invocation via a leakage function restricted on the output size. The security is proved in terms of the pseudorandomness of single (possibly long) output sequence with leakage (hence the game does not consider multiple initializations).

One of the advantages of LR4 over LR-PRG/stream cipher is that LR4 offers an explicit synchronization. LR4 can immediately generate arbitrary temporal keys, which indicates that LR4 can redeem the communication whenever the synchronization fails (maliciously or accidentally) and can start a new session without reset. In contrast, LR-PRG and stream cipher require a reset with a new initialization vector in cases of synchronization failure or new session beginning. As mentioned in Section 1, an attacker may mount an SCA on LR-PRG/stream cipher during some first state updates, if the attacker can trigger resets repeatedly. Thus, an explicit synchronization is essential for an LR rekeying, which makes LR4 more suitable.

7.2 Relation and comparison to LR-PRFs

As mentioned, known LR-PRFs [MSJ12, FPS12] have structural similarities to our approach in terms of the use of GGM, but LR4 and [MSJ12, FPS12] are basically incomparable due to the different leakage models and the assumptions on the primitives. In [FPS12], the authors assume non-adaptive bounded leakage and a weak PRF as an underlying primitive. They show how to construct leakage resilient *non-adaptive* PRF, in which the adversary non-adaptively chooses the inputs of PRF. In [MSJ12], a formal security proof is not given, but the authors show that parallel implementation improves the security and efficiency of GGM-like LR-PRF. In terms of the constructions, two LR-PRFs [MSJ12, FPS12] use independent public randomness for each node on the path. These public random values (IVs) are crucial for their security proofs and significantly increase the bandwidth. Moreover, the generation of these random values must be secure even under leakage, which can be quite costly in practice. For completeness, we briefly describe GGM and [MSJ12, FPS12] in Appendix A.

7.3 Applicability of LR4 to LR-AEs

As discussed in Section 4.3, many LR-AE proposals are designed with *leveled implementation* [BBC+20] in mind [PSV15, BGP+19, DJS19, KS20, DEM+20, BBB+20, SPS+22]. Although their security assumptions and leakage models vary (as discussed in [BBC+20]), they share the core idea of combining a leak-free/DPA-resistant component for, for example, the derivation of a temporal key, and SPA-resistant component(s) using the derived temporal key for the rest of the encryption routine. As we have discussed in Section 4.3, LR4 could be used as a component of existing LR-AEs if they meet certain conditions. However, these conditions are not always met, particularly when it comes to the *tag-generation function (TGF)* (see [BBC+20]). As discussed in Section 4.3, extending LR4 to handle such case would be an interesting future direction.

Ultimately, the goal of LR-AEs is to *improve the temporal key lifetime and change the key lifetime unit from the number of (tweakable) block cipher calls to AE calls.* When applicable, an LR rekeying scheme contributes to this goal.

8 Conclusion

8.1 Summary

This paper studied rekeying as a power/EM SCA countermeasure and presented a new higher-order and LR rekeying scheme named LR4. We developed a leakage model for rekeying to formally prove the security of LR4, and analyzed its performance overhead and practical usecases. In addition, we defined the success rate of attack on rekeying schemes and developed a methodology for evaluating the success rate quantitatively through a unification of bounded trace complexity and bounded leakage. This is useful for determining the rekeying frequency for a bounded leakage (defined as a mutual information value for a given device here) regarding a success rate, which is mandatory for the practical usage of LR4. Through a numerical evaluation, we confirmed the validity and effectiveness of LR4 as an SCA countermeasure (as well as masking), as the number of secure encryption/decryption calls increases exponentially by an increase of rekeying order under practical conditions.

8.2 Future works

Relaxing security assumption. Our current security proof relies on the idealized primitives. A standard model-based proof would provide additional confidence (see *e.g.*, [BGPS21]). It might be possible to remove a (pseudo)random property for G as observed by MSGR. Moreover, it should be noted that our SR definition is related to the multi-user analysis [DLMS14, BT16, LMP17, HTT18, DGGP21, NSSY22] (as in Remark 3). Clarifying the relationship would be an interesting future direction.

Investigation of other possible rekeying construction. In this paper, we discussed the evaluation of LR4 in Section 5.2, but Definition 1 and our evaluation methodology are readily and naturally generalizable and extendable to other rekeying schemes (as in Proposition 2 in Section 6.2.3). It is an important future work to investigate efficient rekeying constructions that make the master key lifetime longer.

Extension to SCAs other than power/EM attack. The focus of this paper was power/EM SCAs, for which we developed the models, security proofs, and an evaluation methodology. It would be valuable to extend our theory and methodology to utilize rekeying in a provable secure manner against other SCAs such as timing and cache attacks.

Acknowledgments

We are grateful to anonymous reviewers and shepherd for their comments and considerations, which were very useful for improving our paper. This work has been supported by JSPS Kakanhi Grant No. 19H21526 and JST CREST No. JPMJCR19K5.

A Existing LR-PRFs

Goldreich–Goldwasser–Micali (GGM) scheme [GGM86]. Existing LR-PRFs are based on the classical Goldreich–Goldwasser–Micali (GGM) PRF that is built on a PRG. Let $f : \{0,1\}^{n_f} \to \{0,1\}^{2n_f}$ be a PRG, and let $f(K) = (f_0(K), f_1(K))$, where $|f_0(K)| = |f_1(K)| = n_f$. To implement a PRF with n_f-bit key and d-bit input, GGM builds a binary tree of depth d. Each node at depth i represents the i-th input bit and the entire input determines the path from the root to the leaf node, where each edge specifies whether f_0 or f_1 is used to derive the key for the child node. More specifically, for input $X = (X_1, \ldots, X_d) \in \{0,1\}^d$ and the key k, GGM PRF output is

$$f_{X_d}(f_{X_{d-1}}(\ldots f_{X_3}(f_{X_2}(f_{X_1}(k)))\ldots)).$$

Faust–Pietrzak–Schipper (FPS) scheme [FPS12]. The FPS scheme consists in weak PRF (wPRF), whose outputs are indistinguishable from random only for random inputs. Let n_k denote the secret key bit-length. Let $F_k(\cdot)$ denote an n_F-bit-input and $n_{F'}$-bit-output LR-PRF by FPS ($n_{F'} = 2n_k$). Let $f_k(\cdot)$ denote an n_E-bit-input and $n_{F'}$-bit-output wPRF. Let $\bar{f}_k(r;0)$ and $\bar{f}_k(r;1)$ be the lower and upper n_k bits of $f_k(r)$, respectively. Using n_F n_E-bit public randomness $r_1, r_2, \dots, r_{n_F}$, the FPS scheme evaluates $F_k(t)$ as

$$\begin{cases} F_k(t) = f_{k_{n_F}}(r_{n_F}), \\ k_{i+1} = \bar{f}_{k_i}(r_i; t[i]) \quad (1 \le i < n_F), \\ k_1 = k. \end{cases}$$

Faust *et al.* proved that the above PRF is leakage resilient if both the leakage function and inputs are non-adaptive.

Medwed–Standaert–Joux (MSJ) scheme [MSJ12]. The MSJ scheme consists in a multi-ary tree for an improved computational cost, whereas GGM and FPS employ a binary tree. Let n_b denote a positive integer divisible n_E, and let $n_s = n_E/n_b$. Typically, n_b is defined as the bit-length of Sbox (*e.g.*, $n_b = 8$ and $n_s = 16$ for AES). Let $t_{(n_b)}[i]$ $(1 \le i \le n_s)$ denote the i-th digit of t in the 2^{n_b}-ary number representation. For example, $t_{(n_b)}[i]$ is typically given by two hexadecimal numbers for AES as $2^{n_b} = 16^2$. Using 2^{n_b} distinct public values $r_0, r_1, \dots, r_{2^{n_b}-1}$, the MSJ scheme evaluates $F_k(t)$ using n_s iterative block cipher calls, as

$$\begin{cases} F_k(t) = k_{n_s+1}, \\ k_{i+1} = E_{k_i}(r_{t_{(n_b)}[i]}) \quad (1 \le i \le n_s), \\ k_1 = k. \end{cases}$$

The MSJ scheme performs one $F_k(t)$ evaluation with n_s encryption calls, which is a significant reduction from $n_E = n_b n_s$ of the GGM scheme. For the MSJ scheme, Medwed *et al.* suggested determining the i-th public value r_i as $r_i = i \parallel i \parallel \cdots \parallel i$, which increases the trace complexity bound (*i.e.*, decreases the SR or increases the number of traces). Some improvements and practical evaluations of MSJ have been devoted in [MSNF16, USS$^+$20, BMPS21].

B Proof of Theorem 1

B.1 H-coefficient technique

Assume that computationally-unbounded adversary $\mathcal{A}$ queries to the two worlds: real and ideal, denoted by $\mathcal{O}_{\mathsf{re}}$ and $\mathcal{O}_{\mathsf{id}}$, and tries to distinguish them. The H-coefficient [Pat08, CS14] is a general technique to evaluate the distinguishing probability of $\mathcal{A}$. We define a *transcript* as a set of input/output values that $\mathcal{A}$ obtains during the interaction with the world. Let T_{re} (resp. T_{id}) denote the probability distribution of the transcript induced by the real world (resp. the ideal world). By extension, we also use the same notation to refer to a random variable distributed according to each distribution. We say that a transcript τ is *attainable* if $\Pr[\mathsf{T}_{\mathsf{id}} = \tau] > 0$ holds with respect to $\mathcal{A}$. Let Θ denote the set of attainable transcripts. The following is the fundamental lemma of H-coefficient technique; see *e.g.* [CS14] for the proof.

Lemma 2. *Let $\Theta = \Theta_{\mathrm{good}} \sqcup \Theta_{\mathrm{bad}}$ be a partition of the set of attainable transcripts. Assume that there exists $\varepsilon_1 \ge 0$ such that for any $\tau \in \Theta_{\mathrm{good}}$, one has*

$$\frac{\Pr[\mathsf{T}_{\mathsf{re}} = \tau]}{\Pr[\mathsf{T}_{\mathsf{id}} = \tau]} \ge 1 - \varepsilon_1,$$

and that there exists $\varepsilon_2 \ge 0$ such that $\Pr[\mathsf{T}_{\mathsf{id}} \in \Theta_{\mathrm{bad}}] \le \varepsilon_2$. Then $|\Pr[\mathcal{A}^{\mathcal{O}_{\mathsf{re}}} \to 1] - \Pr[\mathcal{A}^{\mathcal{O}_{\mathsf{id}}} \to 1]| \le \varepsilon_1 + \varepsilon_2$.

B.2 Evaluation of good transcript probability ratio

In Section 4, we defined bad events to show how to partite the set of attainable transcripts and then showed the evaluation of ε_2, *i.e.*, $\Pr[\mathsf{T}_{\mathsf{id}} \in \Theta_{\mathrm{bad}}] \le d(q+q_L)^2/2^{n_k+1} + (q+q_L)(p+p_I)/2^{n_k} + 4m'q/2^{n_{\mathsf{bc}}}$. All that remains is evaluating a good transcript probability ratio, *i.e.*, ε_1 in Lemma 2.

Lemma 3. *For any $\tau \in \Theta_{\mathrm{good}}$, we obtain the following evaluation:*

$$\frac{\Pr[\mathsf{T}_{\mathsf{re}} = \tau]}{\Pr[\mathsf{T}_{\mathsf{id}} = \tau]} \ge 1.$$

Proof. Let $\tau = \{\mathcal{Q}_C, \mathcal{Q}_{G_1}, \ldots, \mathcal{Q}_{G_d}, \mathcal{Q}_E, \mathcal{Q}_L, \mathcal{Q}_K\}$ and $\tau \in \Theta_{\mathrm{good}}$. Let T_C, T_{G_1}, …, T_{G_d}, T_E, T_L, T_K denote the random variables of each transcript. Let $* \in \{\mathsf{re}, \mathsf{id}\}$. In both real and ideal worlds, we obtain the following evaluation:

$$\begin{aligned}
&\Pr[\mathsf{T}_* = \tau] \\
&\quad = \Pr[\mathsf{T}_C = \mathcal{Q}_C, \mathsf{T}_{G_1} = \mathcal{Q}_{G_1}, \ldots, \mathsf{T}_{G_1} = \mathcal{Q}_{G_d}, \mathsf{T}_E = \mathcal{Q}_E, \mathsf{T}_L = \mathcal{Q}_L, \mathsf{T}_K = \mathcal{Q}_K] \\
&\quad = \underbrace{\Pr[\mathsf{T}_{G_1} = \mathcal{Q}_{G_1}, \ldots, \mathsf{T}_{G_1} = \mathcal{Q}_{G_d}, \mathsf{T}_E = \mathcal{Q}_E]}_{:=\mathsf{P1}_*} \\
&\qquad \times \underbrace{\Pr[\mathsf{T}_K = \mathcal{Q}_K \mid \mathsf{T}_{G_1} = \mathcal{Q}_{G_1}, \ldots, \mathsf{T}_{G_1} = \mathcal{Q}_{G_d}, \mathsf{T}_E = \mathcal{Q}_E]}_{:=\mathsf{P2}_*} \\
&\qquad \times \underbrace{\Pr[\mathsf{T}_C = \mathcal{Q}_C, \mathsf{T}_L = \mathcal{Q}_L, \mid \mathsf{T}_{G_1} = \mathcal{Q}_{G_1}, \ldots, \mathsf{T}_{G_1} = \mathcal{Q}_{G_d}, \mathsf{T}_E = \mathcal{Q}_E, \mathsf{T}_K = \mathcal{Q}_K]}_{:=\mathsf{P3}_*} \\
&\quad := \mathsf{P1}_* \times \mathsf{P2}_* \times \mathsf{P3}_*.
\end{aligned}$$

To prove Lemma 3, we evaluate the lower bound of $(\mathsf{P1}_{\mathsf{re}} \cdot \mathsf{P2}_{\mathsf{re}} \cdot \mathsf{P3}_{\mathsf{re}})/(\mathsf{P1}_{\mathsf{id}} \cdot \mathsf{P2}_{\mathsf{id}} \cdot \mathsf{P3}_{\mathsf{id}})$.

Evaluation of $\mathsf{P1}_{\mathsf{re}}$ and $\mathsf{P1}_{\mathsf{id}}$. We first obtain $\mathsf{P1}_{\mathsf{re}} = \mathsf{P1}_{\mathsf{id}}$ because the probability distribution of transcripts defined by the interactions with the oracles G_1, …, G_d, and $E^{\pm}$ are identical in both worlds.

Evaluation of $\mathsf{P2}_{\mathsf{re}}$ and $\mathsf{P2}_{\mathsf{id}}$. In the ideal world, the keys revealed by the construction oracle (*i.e.*, $k^{(1)}_{\cdot,\cdot}$) are chosen at random and independently from $\{0,1\}^{n_k}$. Regarding keys revealed by the leakage oracle (*i.e.*, $k^{(0)}_{\cdot,\cdot}$), recall that the transcript τ is *good*; thus, there is no collision between the revealed keys in the same depth in $\mathcal{Q}_K$ (*i.e.*, $\overline{\mathrm{Bad1}}$), and there is no collision between in revealed keys of depth i, $k^{(0)}_{i,\cdot}$, and the input keys of the RO G_i where $i \in [d]$ (*i.e.*, $\overline{\mathrm{Bad2}}$). Therefore, we obtain $\mathsf{P2}_{\mathsf{id}} = (1/2^{n_k})^{\Sigma_{i=1}^{d+1} \mathsf{nk}_i}$, where nk_i is the number of elements $k^{(\cdot)}_{i,\cdot}$ in $\mathcal{Q}_K$ (*i.e.*, the number of revealed keys in i-th depth).

In the real world, keys revealed from the construction oracle are all real, unlike in the ideal world. However, due to $\overline{\mathrm{Bad1}}$ and $\overline{\mathrm{Bad2}}$, we obtain $\mathsf{P2}_{\mathsf{re}} = (1/2^{n_k})^{\Sigma_{i=1}^{d+1} \mathsf{nk}_i}$ in the same manner as the above discussion of keys revealed by the leakage oracle in the ideal world. Therefore, we obtain $\mathsf{P2}_{\mathsf{re}} = \mathsf{P2}_{\mathsf{id}}$.

Evaluation of $\mathsf{P3}_{\mathsf{re}}$ and $\mathsf{P3}_{\mathsf{id}}$. In the ideal world, the construction oracle is TURP $\widetilde{\mathsf{P}}^{\pm}$; thus, T_C and T_L are independent. Then we obtain the following equations.

$$\begin{aligned}\mathsf{P3}_{\mathsf{id}} &:= \Pr[\mathsf{T}_C = \mathcal{Q}_C, \mathsf{T}_L = \mathcal{Q}_L, \mid \mathsf{T}_{G_1} = \mathcal{Q}_{G_1}, \ldots, \mathsf{T}_{G_1} = \mathcal{Q}_{G_d}, \mathsf{T}_E = \mathcal{Q}_E, \mathsf{T}_K = \mathcal{Q}_K] \\ &= \underbrace{\Pr[\mathsf{T}_C = \mathcal{Q}_C \mid \mathsf{T}_{G_1} = \mathcal{Q}_{G_1}, \ldots, \mathsf{T}_{G_1} = \mathcal{Q}_{G_d}, \mathsf{T}_E = \mathcal{Q}_E, \mathsf{T}_K = \mathcal{Q}_K]}_{:=\mathsf{P4}_{\mathsf{id}}} \\ &\quad \times \underbrace{\Pr[\mathsf{T}_L = \mathcal{Q}_L, \mid \mathsf{T}_{G_1} = \mathcal{Q}_{G_1}, \ldots, \mathsf{T}_{G_1} = \mathcal{Q}_{G_d}, \mathsf{T}_E = \mathcal{Q}_E, \mathsf{T}_K = \mathcal{Q}_K]}_{:=\mathsf{P5}_{\mathsf{id}}} \\ &:= \mathsf{P4}_{\mathsf{id}} \times \mathsf{P5}_{\mathsf{id}}.\end{aligned}$$

Recall that Cnc is the number of distinct counters in construction queries, and $\widetilde{\mathsf{ctr}^d_1}$, …, $\widetilde{\mathsf{ctr}^d_{\mathsf{Cnc}}}$ are the distinct counters. Also recall that q_1, …, q_{Cnc} are the number of construction queries whose counter is $\widetilde{\mathsf{ctr}^d_1}$, …, $\widetilde{\mathsf{ctr}^d_{\mathsf{Cnc}}}$, respectively. Since the adversary queries to $\widetilde{\mathsf{P}}^{\pm}$ which is independent from other oracles, we obtain

$$\mathsf{P4}_{\mathsf{id}} = \prod_{i=1}^{\mathsf{Cnc}} \prod_{j=0}^{q_i - 1} \frac{1}{(2^{n_{\mathsf{bc}}} - j)}.$$

Similarly, we define Lnc as the number of distinct counters in leakage queries, and $\widetilde{\mathsf{Lctr}^d_1}$, …, $\widetilde{\mathsf{Lctr}^d_{\mathsf{Lnc}}}$ as the distinct counters. Also, let $q_{L,1}$, …, $q_{L,\mathsf{Lnc}}$ be the number of leakage queries whose counter is $\widetilde{\mathsf{Lctr}^d_1}$, …, $\widetilde{\mathsf{Lctr}^d_{\mathsf{Lnc}}}$, respectively; thus, $q_{L,i} \le m'$ for $i \in [\mathsf{Lnc}]$ and $\sum_{i=1}^{\mathsf{Lnc}} q_{L,i} = q_L$. Here, temporal key values inputted into E in LR4-L, which are derived from $\widetilde{\mathsf{Lctr}^d_1}$, …, $\widetilde{\mathsf{Lctr}^d_{\mathsf{Lnc}}}$, are all distinct due to $\overline{\text{Bad1}}$. Also, there is no collision between the temporal keys in LR4-L and input keys of the IC due to $\overline{\text{Bad3}}$. Thus, we obtain

$$\mathsf{P5}_{\mathsf{id}} = \prod_{i=1}^{\mathsf{Lnc}} \prod_{j=0}^{q_{L,i} - 1} \frac{1}{(2^{n_{\mathsf{bc}}} - j)}.$$

In the real world, unlike the case of $\mathsf{P3}_{\mathsf{id}}$, we cannot divide the evaluation of $\mathsf{P3}_{\mathsf{re}}$ into two evaluations about T_C and T_L since they are not independent in the real world. However, we can discuss the evaluation of $\mathsf{P3}_{\mathsf{re}}$ in almost the same manner as $\mathsf{P5}_{\mathsf{id}}$. We define CLnc as the number of distinct counters throughout the construction and leakage queries (*i.e.*, $\mathsf{CLnc} \le \mathsf{Cnc} + \mathsf{Lnc}$), and $\widetilde{\mathsf{CLctr}^d_1}$, …, $\widetilde{\mathsf{CLctr}^d_{\mathsf{CLnc}}}$ as the distinct counters throughout $\mathcal{Q}_C$ and $\mathcal{Q}_L$. Also, let $q_{CL,1}$, …, $q_{CL,\mathsf{CLnc}}$ be the summation number of construction and leakage queries whose counter is $\widetilde{\mathsf{CLctr}^d_1}$, …, $\widetilde{\mathsf{CLctr}^d_{\mathsf{CLnc}}}$, respectively; thus, $\sum_{i=1}^{\mathsf{CLnc}} q_{CL,i} = q + q_L$. As in the case of $\mathsf{P5}_{\mathsf{id}}$, all the temporal keys derived from $\widetilde{\mathsf{CLctr}^d_1}$, …, $\widetilde{\mathsf{CLctr}^d_{\mathsf{CLnc}}}$ are distinct and have no collision with the input key of the IC due to $\overline{\text{Bad1}}$ and $\overline{\text{Bad3}}$. Thus, we obtain

$$\mathsf{P3}_{\mathsf{re}} = \prod_{i=1}^{\mathsf{CLnc}} \prod_{j=0}^{q_{CL,i} - 1} \frac{1}{(2^{n_{\mathsf{bc}}} - j)}.$$

We next show $\mathsf{P3}_{\mathsf{re}} \ge \mathsf{P4}_{\mathsf{id}} \cdot \mathsf{P5}_{\mathsf{id}}$ holds. For $\forall i \in [\mathsf{CLnc}]$, the query of $\widetilde{\mathsf{CLctr}^d_i}$ in $\mathcal{Q}_C$ and $\mathcal{Q}_L$ can be classified into any of the following three cases: (Case 1) $\widetilde{\mathsf{CLctr}^d_i}$ is queried only in $\mathcal{Q}_C$, (Case 2) $\widetilde{\mathsf{CLctr}^d_i}$ is queried only in $\mathcal{Q}_L$, (Case 3) $\widetilde{\mathsf{CLctr}^d_i}$ is queried in both $\mathcal{Q}_C$ and

$\mathcal{Q}_L$. When (Case 1), there exists $i' \in [\mathsf{Cnc}]$ *s.t.* $\widetilde{\mathsf{CLctr}_i^d} = \widetilde{\mathsf{ctr}_{i'}^d}$, and $q_{CL,i} = q_{i'}$ holds; thus, we obtain

$$\prod_{j=0}^{q_{CL,i}-1} \frac{1}{(2^{n_{\mathsf{bc}}} - j)} = \prod_{j=0}^{q_{i'}-1} \frac{1}{(2^{n_{\mathsf{bc}}} - j)}. \tag{12}$$

In the same manner as (Case 1), when (Case 2), we obtain the following evaluation:

$$\prod_{j=0}^{q_{CL,i}-1} \frac{1}{(2^{n_{\mathsf{bc}}} - j)} = \prod_{j=0}^{q_{L,i''}-1} \frac{1}{(2^{n_{\mathsf{bc}}} - j)}, \tag{13}$$

where $i'' \in [\mathsf{Lnc}]$ *s.t.* $\widetilde{\mathsf{CLctr}_i^d} = \widetilde{\mathsf{Lctr}_{i''}^d}$, and $q_{CL,i} = q_{L,i''}$. When (Case 3), there exists $i' \in [\mathsf{Cnc}]$ and $i'' \in [\mathsf{Lnc}]$ *s.t.* $\widetilde{\mathsf{CLctr}_i^d} = \widetilde{\mathsf{ctr}_{i'}^d} = \widetilde{\mathsf{Lctr}_{i''}^d}$, and $q_{CL,i} = q_{i'} + q_{L,i''}$ holds; thus, we obtain

$$\prod_{j=0}^{q_{CL,i}-1} \frac{1}{(2^{n_{\mathsf{bc}}} - j)} > \prod_{j=0}^{q_{i'}-1} \frac{1}{(2^{n_{\mathsf{bc}}} - j)} \cdot \prod_{j=0}^{q_{L,i''}-1} \frac{1}{(2^{n_{\mathsf{bc}}} - j)}. \tag{14}$$

By multiplying any of the (in)equations (12), (13), (14) for all $i \in [\mathsf{CLnc}]$, we obtain $\mathsf{P3}_{\mathsf{re}} \geq \mathsf{P4}_{\mathsf{id}} \cdot \mathsf{P5}_{\mathsf{id}}$.

Wrapping up of the proof. From the above three paragraphs, we obtain the following inequality:

$$\frac{\Pr[\mathsf{T}_{\mathsf{re}} = \tau]}{\Pr[\mathsf{T}_{\mathsf{id}} = \tau]} = \frac{\mathsf{P1}_{\mathsf{re}} \cdot \mathsf{P2}_{\mathsf{re}} \cdot \mathsf{P3}_{\mathsf{re}}}{\mathsf{P1}_{\mathsf{id}} \cdot \mathsf{P2}_{\mathsf{id}} \cdot \mathsf{P3}_{\mathsf{id}}} = \frac{\mathsf{P3}_{\mathsf{re}}}{\mathsf{P3}_{\mathsf{id}}} = \frac{\mathsf{P3}_{\mathsf{re}}}{\mathsf{P4}_{\mathsf{id}} \cdot \mathsf{P5}_{\mathsf{id}}} \geq 1.$$

This completes the proof. □

References

[AB00] Michel Abdalla and Mihir Bellare. Increasing the lifetime of a key: A comparative analysis of the security of re-keying techniques. In *Advances in Cryptology—ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 546–559. Springer, 2000.

[ADW09] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *Advances in Cryptology—CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 36–54, 2009.

[ASB14] Janaka Alawatugoda, Douglas Stebila, and Colin Boyd. Modelling after-the-fact leakage for key exchange. In *ACM Symposium on Information, Computer, and Communications Security (ASIA CCS 2014)*, pages 207–216, 2014.

[BBB+20] Davide Bellizia, Francesco Berti, Olivier Bronchain, Gaëtan Cassiers, Sébastien Duval, Chun Guo, Gregor Leander, Gaëtan Leurent, Itamar Levi, Charles Momin, Olivier Pereira, Thomas Peters, François-Xavier Standaert, Balazs Udvarhelyi, and Friedrich Wiemer. Spook: Sponge-based leakage-resistant authenticated encryption with a masked tweakable block cipher. *IACR Transactions on Symmetric Cryptology*, 2020(S1):295–349, 2020.

[BBC$^+$19] Gilles Barthe, Sonia Balaïd, Gaëtan Cassiers, Pierre-Alan Fouque, Benjamin Grégoire, and François-Xavier Standaert. maskVerif: Automated verification of higher-order masking in presence of physical defaults. In *European Symposium on Research in Computer Security (ESORICS 2019)*, volume 11735 of *Lecture Notes in Computer Science*, pages 300–318. Springer, 2019.

[BBC$^+$20] Davide Bellizia, Olivier Bronchain, Gaëtan Cassiers, Vincent Grosso, Chun Guo, Charles Momin, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Mode-level vs. implementation-level physical security in symmetric cryptography: A practical guide through the leakage-resistance jungle. In *Advances in Cryptology—CRYPTO 2022*, volume 12170 of *Lecture Notes in Computer Science*, pages 369–400, 2020.

[BBD$^+$16] Gilles Barthe, Sonia Balaïd, Fraonçois Dupressoir, Pierre-Alan Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. In *ACM SIGSAC Conference on Computer Communications Security (CCS 2016)*, pages 116–129, 2016.

[BCG$^+$23] Julien Béguinot, Wei Cheng, Sylvain Guilley, Yi Liu, Loïc Masure, Olivier Rioul, and François-Xavier Standeart. Removing the field size loss from Duc et al.'s conjectured bound for masked encodings. In *International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE 2023)*, volume 13979 of *Lecture Notes in Computer Science*, pages 86–104, 2023.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004)*, Lecture Notes in Computer Science, pages 16–29, Berlin, Heidelberg, 2004. Springer.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults. In *Advances in Cryptology—EUROCRYPT '97*, pages 37–51, 1997.

[BDSH$^+$14] Sonia Belaïd, Fabrizio De Santis, Johann Heyszl, Stefan Mangard, Marcel Medwed, Jørn-Marc Schmidt, François-Xavier Standaert, and Stefan Tillich. Towards fresh re-keying with leakage-resilient PRFs: Cipher design principles and analysis. *Journal of Cryptographic Engineering*, 4(3):157–171, 2014.

[BGG$^+$14] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In *International Conference on Smart Card Research and Advanced Applications (CARDIS 2014)*, volume 8968 of *Lecture Notes in Computer Science*, pages 64–81, 2014.

[BGP$^+$19] Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. TEDT, a leakage-resist AEAD mode for high physical security applications. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):256–320, 2019.

[BGPS21] Francesco Berti, Chun Guo, Thomas Peters, and François-Xavier Standaert. Efficient leakage-resilient MACs without idealized assumptions. In *Advances in Cryptology—ASIACRYPT 2021 (2)*, volume 13091 of *Lecture Notes in Computer Science*, pages 95–123. Springer, 2021.

[BMOS17] Guy Barwell, Daniel P. Martin, Elisabeth Oswald, and Martijn Stam. Authenticated encryption in the face of protocol and side channel leakage. In *Advanced in Cryptology—ASIACRYPT 2017*, volume 10624 of *Lecture Notes in Computer Science*, pages 693–723. Springer, 2017.

[BMPS21] Olivier Bronchain, Charles Momin, Thomas Peters, and François-Xavier Standaert. Improved leakage-resistant authenticated encryption based on hardware AES coprocessors. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):641–676, 2021.

[BMRT22] Sonia Belaïd, Darius Mercadier, Matthieu Rivain, and Abdul Rahman Taleb. IronMask: Versatile verification of masking security. In *IEEE Symposium on Security and Privacy (SP)*, pages 142–160, 2022.

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer, 2000.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *Advances in Cryptology—CRYPTO '97*, pages 513–525, 1997.

[BS21] Olivier Bronchain and François-Xavier Standeart. Breaking masked implementations with many shares on 32-bit software platforms: or when the security order does not matter. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 3:202–234, 2021.

[BSW12] Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. *Journal of Cryptology*, 26:513–558, 2012.

[BT16] Mihir Bellare and Björn Tackmann. The multi-user security of authenticated encryption: AES-GCM in TLS 1.3. In *Advances in Cryptology—CRYPTO 2016*, Lecture Notes in Computer Science, pages 247–276, 2016.

[CLS15] Benoit Cogliati, Rodolphe Lampe, and Yannick Seurin. Tweaking Even-Mansour Ciphers. In *Advances in Cryptology—CRYPTO 2015 (1)*, volume 9215 of *Lecture Notes in Computer Science*, pages 189–208. Springer, 2015.

[CMY$^+$16] Rongmao Chen, Yi Mu, Guomin Yang, Willy Susilo, and Fuchun Guo. Strongly leakage-resilient authenticated key exchange. In *Topics in Cryptology—CT-RSA 2016*, volume 9610 of *Lecture Notes in Computer Science*, pages 19–36, 2016.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2002)*, LNCS, pages 13–28, 2002.

[CS14] Shan Chen and John P. Steinberger. Tight security bounds for key-alternating ciphers. In *EUROCRYPT*, pages 327–350. Springer, 2014.

[dCBG$^+$17] Thomas de Cnudde, Begül Bilgin, Benedikt Gierlichs, Ventzislav Nikov, Svetla Nikova, and Vincent Rijmen. Does coupling affect the security of masked implementations? In *International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE 2017)*, volume 10348 of *Lecture Notes in Computer Science*, pages 1–18, 2017.

[DCEM18] Thomas De Cnudde, Maik Ender, and Amir Moradi. Hardware masking, revisited. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, (2):123–148, 2018.

[dCGRP19] Eloi de Chérisey, Sylvain Guilley, Olivier Rioul, and Pablo Piantanida. Best information is most successful: Mutual information and success rate in side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(2):49–79, 2019.

[DEM+20] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennik, Robert Primas, and Thomas Unterlaggauer. Isap v2.0. *IACR Transactions on Symmetric Cryptology*, 2020(S1):390–416, 2020.

[DEMM14] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, and Florian Mendel. On the security of fresh re-keying to counteract side-channel and fault attacks. In *International Conference on Smart Card Research and Advanced Applications (CARDIS 2014)*, volume 8968 of *Lecture Notes in Computer Science*, pages 233–244. Springer, 2014.

[DEMS21] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. ASCON: Lightweight authentication & hashing, 2021.

[DFH+16] Stefan Dziembowski, Sebastian Faust, Gottfried Herold, Anthony Journault, Daniel Masny, and François-Xavier Standaert. Towards sound fresh re-keying with hard (physical) learning problems. In *Advances in Cryptology—CRYPTO 2016*, volume 9815 of *Lecture Notes in Computer Science*, pages 272–301, 2016.

[DFS15] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete: Or how to evaluate the security of any leakage device. In *Advances in Cryptology—EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 401–429. Springer, 2015.

[DGGP21] Jean Paul Degabriele, Jërôme Govinden, Felix Günther, and Kenneth G. Paterson. The security of ChaCha20-Poly1305 in the multi-user setting. In *ACM SIGSAC Conference on Computer Communications Security (CCS 2021)*, pages 1981–2003, 2021.

[DHLAW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In *Advances in Cryptology—ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 613–631, 2010.

[DJS19] Jean Paul Degabriele, Christian Janson, and Patrick Struck. Sponges resist leakage: The case of authenticated encryption. In *Advances in Cryptology—ASIACRYPT 2019*, volume 11922 of *Lecture Notes in Computer Science*, pages 209–240, 2019.

[DLMS14] Yuanxi Dai, Jooyoung Lee, Bart Mennik, and John Steinberger. The security of multiple encryption in the ideal cipher model. In *Advances in Cryptology—CRYPTO 2014*, Lecture Notes in Computer Science, pages 20–38, 2014.

[DM19] Christoph Dobraunig and Bart Mennink. Leakage resilience of the duplex construction. In *Advances in Cryptology—ASIACRYPT 2019*, volume 11923 of *Lecture Notes in Computer Science*, pages 225–255. Springer, 2019.

[DMMS21] Sébastien Duval, Pierrick Méaux, Charles Momin, and François-Xavier Standaert. Exploring crypto-physical dark matter and learning with physical rounding: Towards secure and efficient fresh re-keying. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 1:373–401, 2021.

[DMP22] Christoph Dobraunig, Bart Mennink, and Robert Primas. Leakage and tamper resilient permutation-based cryptography. In *ACM SIGSAC Conference on Computer Communications Security (CCS 2022)*, pages 859–873, 2022.

[DP08] Stefan Dziembowski and Krzysztof Pietzrak. Leakage-resilient cryptography. In *Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008)*, pages 293–302, 2008.

[DP10] Yavgeniy Dodis and Kryzsztof Pietzrak. Leakage-resilient pseudorandom functions and side-channel attacks on Feistel networks. In *Advances in Cryptology—CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 21–40, 2010.

[Fel91] Willilam Feller. *An Introduction to Probability Theory and Its Applications, Volume 1, 3rd Edition.* Wiley, 1991.

[FGP$^+$18] Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François-Xavier Standaert. Composable masking schemes in the presence of physical defaults & the robust probing model. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, (3):89–120, 2018.

[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.

[GFM13] Berndt Gammel, Wieland Fischer, and Stefan Mangard. Generating a session key for authentication and secure data transfer. US Patent App. 14/074, 279, 2013. https://patents.google.com/patent/US20100316217.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.

[GIM22] Chun Guo, Tetsu Iwata, and Kazuhiko Minematsu. New indifferentiability security proof of MDPH hash function. *IET Inf. Secur.*, 16(4):262–281, 2022.

[GM17] Hannes Gross and Stefan Mangard. Reconciling $d+1$ masking in hardware and software. In *International Conference on Cryptographic Hardware and Embedded Systems (CHES 2017)*, volume 10529 of *Lecture Notes in Computer Science*. Springer, 2017.

[GMK16] Hannes Gross, Stefan Mangard, and Thomas Korak. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. In *ACM Workshop on Theory of Implementation Security (TIS 2016)*, page 3, 2016.

[GMPO20] Si Gao, Ben Marshall, Dan Page, and Elisabeth Oswald. Share-slicing: Friend or foe? *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):152–174, 2020.

[GSWY20] Chun Guo, François-Xavier Standaert, Weijia Wang, and Yu Yu. Efficient side-channel secure message authentication with better bounds. *IACR Transactions on Symmetric Cryptology*, 2019, Issue 4:23–53, 2020.

[GYTS14] Nahid Farhady Ghalaty, Bilgiday Yuce, Mostafa Taha, and Patrick Schaumont. Differential fault intensity analysis. In *Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 49–58, 2014.

[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.

[Hir06] Shoich Hirose. Some plausible constructions of double-block-length hash functions. In *Fast Software Encryption (FSE 2006)*, volume 4047 of *Lecture Notes in Computer Science*, pages 210–225. Springer, 2006.

[HRG14] Annelie Heuser, Olivier Rioul, and Sylvain Guilley. Good is not good enough: Deriving optimal distinguishers from communication theory. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2014)*, pages 55–74, 2014.

[HTT18] Viet Tung Hoang, Stefano Tessaro, and Aishwarya Thiruvengadam. The multi-user security of GCM, revisited: Tight bounds for nonce randomization. In *ACM SIGSAC Conference on Computer Communications Security (CCS 2018)*, pages 1429–1440, 2018.

[IKMP20] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Duel of the titans: The romulus and remus families of lightweight AEAD algorithms. *IACR Trans. Symmetric Cryptol.*, 2020(1):43–120, 2020.

[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.

[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.

[IUH22a] Akira Ito, Rei Ueno, and Naofumi Homma. On the success rate of side-channel attacks on masked implementations: Information-theoretical bounds and their practical usage. In *ACM SIGSAC Conference on Computer and Communications Security (CCS 2022)*, pages 1521–1535, 2022.

[IUH22b] Akira Ito, Rei Ueno, and Naofumi Homma. Perceived information revisited: New metrics to evaluate success rate of side-channel attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 4:228–254, 2022.

[JB17] Bernhard Jungk and Shivam Bhasin. Don't fall into a trap: Physical side-channel analysis on ChaCha20-Poly1305. In *IEEE/ACM Design, Automation and Test in Europe Conference and Exhibition (DATE 2017)*, pages 1110–1115, 2017.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology—CRYPTO' 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397, 1999.

[KM07] Hidenori Kuwakado and Masakatu Morii. Indifferentiability of single-block-length and rate-1 compression functions. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 90-A(10):2301–2308, 2007.

[KMMS22] David Knichel, Amir Moradi, Nicolai Müller, and Pascal Sasdrich. Automated generation of masked hardware. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 1:589–629, 2022.

[Koc98] Paul C. Kocher. Leak-resistant cryptographic indexed key update. US Patent US6539092B1, 1998. https://patents.google.com/patent/US6539092B1/en.

[KR11] Ted Krovetz and Phillip Rogaway. The software performance of authenticated-encryption modes. In *Fast Software Encryption (FSE 2011)*, volume 6733 of *Lecture Notes in Computer Science*, pages 306–327. Springer, 2011.

[KS20] Juliane Krämer and Patrick Struck. Leakage-resilient authenticated encryption from leakage-resilient pseudorandom functions. In *International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE 2020)*, volume 12244 of *Lecture Notes in Computer Science*, pages 315–337, 2020.

[KSM20] David Knichel, Pascal Sasdrich, and Amir Moradi. SILVER—statistical independence and leakage verification. In *Advances in Cryptology—ASIACRYPT 2020*, volume 12491 of *Lecture Notes in Computer Seience*, pages 787–816, 2020.

[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.

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In *Advances in Cryptology—ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 703–720, 2009.

[LKO$^+$21] Moritz Lipp, Andreas Kogler, David Oswald, Michael Schwarz, Catherine Easdon, Claudio Canella, and Daniel Gruss. PLATYPUS: Software-based power side-channel attacks on x86. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 355–371, 2021.

[LMP17] Atul Luykx, Bart Mennink, and G. Kenneth Paterson. Analyzing multi-key security degradation. In *Advances in Cryptology—ASIACRYPT 2017*, volume 10625 of *Lecture Notes in Computer Seience*, pages 575–605, 2017.

[LSG$^+$10] Yang Li, Kazuo Sakiyama, Shigeto Gomisawa, Toshinori Fukunaga, Junko Takahashi, and Kazuo Ohta. Fault sensitivity analysis. In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 320–334, 2010.

[Men17] Bart Mennink. Optimal collision security in double block length hashing with single length key. *Des. Codes Cryptogr.*, 83(2):357–406, 2017.

[Men20] Bart Mennink. Beyond birthday bound secure fresh rekeying: Application to authenticated encryption. In *Advances in Cryptology—ASIACRYPT 2020*, volume 12491 of *Lecture Notes in Computer Science*, pages 630–661. Springer, 2020.

[MHM14] Zdenek Martinasek, Jan Hajny, and Lukas Malina. Optimization of power analysis using neural network. In *International Conference on Smart Card Research and Advanced Applications (CARDIS 2014)*, Lecture Notes in Computer Science, pages 94–107. Springer International Publishing, 2014.

[MKSM22] Nicolai Müller, David Knichel, Pascal Sasdrich, and Amir Moradi. Transitional leakage in theory and practice: Unveiling security flaws in masked circuits. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2:266–288, 2022.

[MMP$^+$11] Amir Moradi, Oliver Mischke, Christof Paar, Yang Li, Kazuo Ohta, and Kazuo Sakiyama. On the power of fault sensitivity analysis and collision side-channel attacks in a combined setting. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, pages 292–311, 2011.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer New York, 2007.

[MOS11] S. Mangard, E. Oswald, and F.-X. Standaert. One for all – all for one: Unifying standard differential power analysis attacks. *IET Information Security*, 5(2):100–110, June 2011.

[MPR$^+$11] Marcel Medwed, Christophe Petit, Francesco Regazzoni, Mathieu Renauld, and François-Xavier Standaert. Fresh re-keying II: securing multiple parties against side-channel and fault attacks. In *International Conference on Smart Card Research and Advanced Applications (CARDIS 2011)*, volume 7079 of *Lecture Notes in Computer Science*, pages 115–132. Springer, 2011.

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *Theory of Cryptography Conference (TCC 2004)*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In *Theory of Cryptography Conference (TCC 2004)*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004.

[MRS22] Loïc Masure, Olivier Rioul, and François-Xavier Standaert. A nearly tight proof of Duc et al.'s conjectured security bound for masked implementations. In *International Conference on Smart Card Research and Advanced Applications (CARDIS)*, volume 13820 of *Lecture Notes in Computer Science*. Springer, 2022.

[MS14] Taha Mostafa and Patrick Schaumont. Key updating for leakage resiliency with application to AES modes of operation. *IEEE Transactions on Information Forensics and Security*, 10(3):519–528, 2014.

[MSGR10] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In *Progress in Cryptology—AFRICACRYPT 2010*, volume 6055 of *Lecture Notes in Computer Science*, pages 279–296. Springer, 2010.

[MSJ12] Marcel Medwed, François-Xavier Standaert, and Antoine Joux. Towards super-exponential side-channel security with efficient leakage-resilient PRFs. In *International Workshop on Cryptographic Hardware and Embedded Systems*

(CHES 2012), volume 7428 of *Lecture Notes in Computer Science*, pages 193–212. Springer, 2012.

[MSNF16] Marcel Medwed, François-Xavier Standaert, Ventzislav Nikov, and Martin Feldhofer. Unknown-input attacks in the parallel setting: Improving the security of the CHES 2012 leakage-resilient PRF. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology—ASIACRYPT 2016*, pages 602–623. Springer Berlin Heidelberg, 2016.

[Nai19] Yusuke Naito. Optimally indifferentiable double-block-length hashing without post-processing and with support for longer key than single block. In *Progress in Cryptology—LATINCRYPT 2019*, volume 11774 of *Lecture Notes in Computer Science*, pages 65–85. Springer, 2019.

[Nat23] National Institute of Standards and Technology. Lightweight cryptography, March 2023.

[NRS11] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure hardware implementation of nonlinear functions in the presence of glitches. *Journal of Cryptology*, 24(2):292–321, 2011.

[NS09] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *Advances in Cryptology—CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 18–35, 2009.

[NSS20] Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Lightweight authenticated encryption mode suitable for threshold implementation. In *Advances in Cryptology—EUROCRYPT 2020*, volume 12106 of *Lecture Notes in Computer Science*, pages 705–735, 2020.

[NSS22] Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Secret can be public: Low-memory AEAD mode for high-order masking. In *Advances in Cryptology—CRYPTO 2022*, Lecture Notes in Computer Science, July 2022. https://eprint.iacr.org/2022/812.

[NSSY22] Yusuke Naito, Yu Sasaki, Takeshi Sugawara, and Kan Yasuda. The multi-user security of triple encryption, revisited: Exact security, strengthening, and application to TDES. In *ACM SIGSAC Conference on Computer Communications Security (2022)*, pages 2323–2336, 2022.

[Pat08] Jacques Patarin. The "coefficients H" technique. In *Selected Areas in Cryptography*, volume 5381 of *Lecture Notes in Computer Science*, pages 328–345. Springer, 2008.

[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.

[Pie09] Kryzsztof Pietzrak. A leakage-resilient mode of operation. In *Advances in Cryptology—EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 462–482, 2009.

[PM16] Peter Pessl and Stefan Mangard. Enhancing side-channel analysis of binary-field multiplication with bit reliability. In *Topics in Cryptology—CT-RSA 2016*, volume 9610 of *Lecture Notes in Computer Science*, pages 255–270. Springer, 2016.

[PQ03] Gilles Piret and Jean-Jacques Quisquater. A differential fault attack technique against SPN structures, with application to the AES and Khazad. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, pages 77–88, 2003.

[PSV15] Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In *ACM SIGSAC Conference on Computer and Communications Security (CCS 2015)*, pages 96–108, 2015.

[RBN$^+$15] Oscar Reparaz, Begül Bilgin, Svelta Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating masking schemes. In *Advances in Cryptology—CRYPTO 2015*, volume 9215 of *Lecture Notes in Computer Science*, pages 764–783. Springer, 2015.

[Rep16] Oscar Reparaz. Detecting flawed masking schemes with leakage detection tests. In *Fast Software Encryption (2016)*, volume 9783 of *Lecture Notes in Computer Science*, pages 204–222, 2016.

[Res18] Eric Rescorla. The Transport Layer Security (TLS) protocol version 1.3, 2018. https://doi.org/10.17487/RFC8446.

[RSVC$^+$11] Mathieu Renauld, François-Xavier Standeart, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In *Advances in Cryptology—EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 109–128, 2011.

[RTM18] Eric Rescorla, Hannes Tschofenig, and Nagendra Modadugu. The Datagram Transport Layer Security (DTLS) protocol version 1.3—draft-ietf-tls-dtls13-43, 2018. https://tools.ietf.org/html/draft-ietf-tls-dtls13-43.

[SCS$^+$21] A. Madura Shelton, Łukasz Chmielewski, Niels Samwel, Markus Wagner, Lejla Batina, and Yuval Yarom. Rosita++: Automatic higher-order leakage elimination from cryptographic code. In *ACM SIGSAC Conference on Computer Communications Security (CCS 2021)*, pages 685–699, 2021.

[SMY09] François-Xavier Standeart, Tal G. Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Advances in Cryptology—EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461, 2009.

[SPS$^+$22] Yaobin Shen, Thomas Peters, François-Xavier Standaert, Gaëtan Classiers, and Corentin Verhamme. Triplex: an efficient and one-pass leakage-resistant mode of operation. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 4:135–162, 2022.

[SPY$^+$10] François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. *Leakage Resilient Cryptography in Practice*, pages 99–134. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[SSAQ02] David Samyde, Sergei Skorobogatov, Ross Anderson, and Jean-Jacques Quisquater. On a new way to read data from memory. In *First International IEEE Security in Storage Workshop*, pages 65–69, 2002.

[SSB+21] A. Madura Shelton, Niels Samwel, Lejla Batina, Francesco Regazzoni, Markus Wagner, and Yuval Yarom. Rosita: Towards automatic elimination of power-analysis leakage in ciphers. In *Network and Distributed System Security Symposium (NDSS 2021)*, 2021.

[TT21] Martin Thomson and Sean Turner. Using TLS to secure QUIC. RFC 9001, 2021.

[TV04] Kris Tiri and Ingrid Verbauwhede. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In *IEEE/ACM Design, Automation and Test in Europe Conference and Exhibition (DATE 2004)*, pages 246–251, 2004.

[UHMA17] Rei Ueno, Naofumi Homma, Sumio Morioka, and Takafumi Aoki. Automatic generation of formally-proven tamper-resistant Galois-field multipliers based on generalized masking scheme. In *Design, Automation and Test in Europe Conference and Exhibition (DATE 2017)*, pages 978–983. IEEE, 2017.

[UHMA21] Rei Ueno, Naofumi Homma, Sumio Morioka, and Takafumi Aoki. A systematic design methodology of formally proven side-channel-resistant cryptographic hardware. *IEEE Design & Test*, 38(3):84–92, 2021.

[USS+20] Florian Unterstein, Marc Schink, Thomas Schamberger, Lars Tebelmann, Manuel Ilg, and Johann Heyszl. Retrofitting leakage resilient authenticated encryption to microcontrollers. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(4):365–388, 2020.

[YSPY10] Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In *ACM SIGSAC Conference on Computer and Communications Security (CCS 2010)*, pages 141–151, 2010.

[ZBHV19] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient CNN architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):1–36, 2019.

[ZHF+23] Fan Zhang, Run Huang, Tianxiang Feng, Xue Gong, Yulong Tao, Kui Ren, Xinjie Zhao, and Shize Guo. Efficient persistent fault analysis with small number of chosen plaintexts. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(2):519–542, 2023.

[ZLZ+18] Fan Zhang, Xiaoxuan Lou, Xinjie Zhao, Shivam Bhasin, Wei He, Ruyi Ding, Samiya Qureshi, and Kui Ren. Persistent fault analysis on block ciphers. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(3):150–172, 2018.

[ZS18] Mark Zhao and G. Edward Suh. FPGA-based remote power side-channel attacks. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 229–244, 2018.