Exploring the Golden Mean Between Leakage and Fault Resilience and Practice

Christoph Dobraunig[1] and Bart Mennink[1] and Robert Primas[2]

[1] Digital Security Group, Radboud University, Nijmegen, The Netherlands
`cdobraunig@cs.ru.nl`, `b.mennink@cs.ru.nl`
[2] Graz University of Technology, Austria
`rprimas@gmail.com`

Abstract. The area of leakage resilient cryptography aims to provide proofs under the assumption that the side channel leakage of implementations behaves in a certain way, e.g., the leakage is bounded, hard-to-invert, or simulatable. On the other hand, it is often hard to show that a practical implementation has such a behavior. Moreover, these models are typically targeted exclusively towards side channel attacks and hence, other implementation attacks like fault attacks are excluded. In this paper, we provide an alternative approach that we call accumulated leakage. In our model, no a priori restriction or assumption on the leakage is made. Instead, leakage resilience bounds are expressed in terms of an accumulated gain, which is a function of the leakage obtained by an attacker. In particular, we express the accumulated gain as a function of the number of computations of a primitive using a secret that an attacker can observe, one of the major restrictions that determines whether a certain implementation attack is possible or not. Having the advantage of a scheme expressed with the help of accumulated leakage, we have two roads to go. One option is to stick to the a priori bounding made in, e.g., the bounded leakage model and put an a priori restriction on the maximum allowed leakage per primitive call. Another option is to compute the accumulated gain based on measurements a posteriori. As a proof of concept, we apply the accumulated leakage concept to a sponge-based stream encryption scheme called asakey: first, a formal leakage resilience analysis is delivered as a function of the accumulated gain, and second, leakage measurements on permutations are performed to demonstrate how the accumulated gain can be estimated a posteriori.

Keywords: leakage resilience, accumulated leakage, sponge-based encryption, side channel measurements, fault attacks, symmetric cryptography

1 Introduction

With the rise of the Internet of Things (IoT), devices performing cryptographic tasks become ubiquitous. As a consequence, an increasing number of devices operates in environments where attackers do not only have remote access to their interfaces, but also physical access. In such a scenario, implementation attacks, like side channel attacks [25] and fault attacks [9], are a major threat to the security that such devices aim to provide. Hence, research for protection against implementation attacks nowadays plays an important part in cryptography.

One approach in this direction is the concept of leakage resilient cryptography [20]. The goal of this research direction is to design modes of operation that remain secure under specific assumptions on the leakage an adversary can receive. Leakage resilience gave rise to cryptographic schemes with very strong security guarantees, for example, modes of operation that are provably secure against all side channel attacks assuming that the leakage in each round is bounded [36]. Hence, it has attracted the interest of a lot of researchers proposing several leakage resilient symmetric cryptographic schemes [2–4, 17, 19, 21, 29, 35, 46, 47].

On the downside, the promises delivered by leakage resilience come with an assumption on the leakage. A typical example is the assumption that the leakage is bounded [20] or hard-to-invert [18]. Showing that an assumption on the leakage actually holds turns out to be quite hard, and in practice, side channel analysis of leakage resilient schemes typically just considers which side channel attacks can be performed. See, e.g., work on evaluating the security of a leakage resilient pseudo random function [30, 31, 44, 45]. One attempt that has been made to bring the theory of leakage resilient cryptography closer to practice is simulatable leakage [39]. The high-level idea of simulatable leakage is to consider the distance of a cryptographic scheme from a simulator that does not possess the key, but that still generates leakage that is indistinguishable from the device using the actual key. However, Longo et al. [26] pointed out some obstacles with the practical realization of such simulators, and — to the best of our knowledge — the instantiation of simulators is still an open problem. In this light, the requirement to have implementations that ultimately leak as specified by a model might be asking too much.

However, if we look at leakage resilient constructions, we see that independent of the modeling of the leakage, all these constructions aim to limit the number of observations an attacker can make per evaluation of an underlying primitive using a certain key. Considering this from an implementation attack perspective, such limitations on the number of observations make a lot of sense. Attacks like statistical fault attacks (SFA) [22], statistical ineffective fault attacks (SIFA) [12, 13], or differential power analysis (DPA) [25] get better with increasing data complexity an attacker is able to exploit per secret state it tries to recover. Complementing that, these attacks can also get better with the number of observations where the input, as well as the secret of the underlying primitive, remain the same. For instance, such observations can be used to reduce the noise of traces used in a DPA, simple power analysis (SPA), or template attack [28]. Furthermore, also fault attacks can utilize different faults on executions with the same input, like differential fault attacks (DFA) [9].

In this paper, we take a different approach and aim to model the impact of implementation attacks more broadly. We do so by introducing the concept of *accumulated leakage* that allows us to abstract *side channel attacks and fault* *attacks* in the leakage resilient analysis. In a nutshell, the goal of the analysis of our leakage resilient scheme in the accumulated leakage model is to provide a limit on the data complexity for the underlying primitives. This way of modeling leakage is fully backwards compatible and we can easily transform the resulting bounds to the bounded leakage model in the non-adaptive or adaptive leakage setting. Furthermore, accumulated leakage allows us to focus on evaluating implementations of the underlying building block using standard side channel and fault attacks. The results of these evaluations can then be used in the bound for the leakage resilient scheme to express the advantage of an adversary. We will discuss our contributions in more detail below.

1.1 Accumulated Leakage

First, we introduce and formalize the concept of *accumulated leakage* in Section 2. The core idea of accumulated leakage is that one expresses an *accumulated gain* during the experiment. This is a function in terms of all information a side channel attacker has observed so far, and it changes in the course of the attack. The final estimation of the accumulated gain, i.e., the actual side channel effect on a scheme, can be measured *a posteriori*. This approach allows us to discard many restrictions imposed by typical leakage resilience analysis. As a pleasant bonus, this also evades the discussion on whether adaptive or non-adaptive leakage must be considered: it is covered within the accumulated gain.

Note that this stands in sharp contrast with bounded or hard-to-invert leakage, where the leakage *per query* is generously bounded. For example, in the bounded leakage model, one assumes that each *evaluation leaks* at most $\lambda \geq 0$ bits of secret data, and these linearly add up for multiple evaluations. For the sake of comparison, consider the following example. Take the AES block cipher, and assume that an adversary can learn evaluations of $\mathrm{AES}_K(\cdot)$ for secret K. In the bounded leakage model, one assumes that each evaluation leaks λ bits of data, but this means that after $|K|/\lambda$ evaluations, the security proof becomes meaningless. In contrast, in the accumulated leakage model, one assumes that up to the i-th query, the attacker has learned f_i bits of data, which is a function of all information the attacker has observed so far. These values $f_1, f_2, \ldots$ remain yet undetermined, and must be substantiated with side channel experiments. Comparing both approaches, necessarily $f_i \leq i \cdot \lambda$, but typically, the difference is much larger as we will demonstrate in our practical experiments (see Section 1.4).

However, please note that the accumulated gain is backwards compatible and, if it is wished, it can be used in a similar way to bounded leakage. Analogue to the bounded leakage model, we can a priori assume that *each evaluation leaks* at most $\lambda \geq 0$ bits of secret data, and these linearly add up for multiple evaluations in order to give the accumulated gain. However, in contrast to the bounded leakage model that models the leakage function directly, we only model the effect of an attack, namely that an attacker gets a certain advantage in guessing a secret. Hence, we can also naturally include fault attacks, as detailed next.

1.2 Coverage of Fault Attacks

Typical methods in leakage resilient cryptography aim to precisely model the gain an attacker can get with the help of the physical leakage. For instance, in the bounded leakage model the leakage function can be any arbitrary function of the secret state with bounded output length, or any function that preserves some min-entropy of the secret state. Later, hard-to-invert leakage was introduced, which, on a high level, requires that the leakage has the property that even under knowledge of the leakage, the secret state is hard to guess [18,23]. However, such attempts to model the gain an attacker can get from physical leakage leave out the threat of fault attacks.

This is a weak point in existing approaches, since in scenarios where side channel attacks are applicable, an attacker can typically also apply a wider range of implementation attacks, such as fault attacks. Hence, our concept of accumulated leakage does not aim to model the physical leakage prior to an implementation attack, but rather at the end of the attack. Therefore, accumulated leakage is agnostic to the type of implementation attack and hence our results are naturally applicable to a wide range of attack scenarios, including fault attacks.

1.3 Application of Accumulated Leakage

We demonstrate how the concept of accumulated leakage can be incorporated in leakage resilience analyses.

The first exposition is on a simple key recovery attack game against a keyed random function, in Section 3. The example might sound impractical at first sense, but it appears typically *within* larger security analyses. In addition, a keyed random function is the simplest possible example to demonstrate how our framework functions. The reason for this is that a keyed random function is perfectly indistinguishable from a random function as long as an adversary does not guess the key, and the main target of side channel attacks in this context would be to recover the key.

Our analysis on the keyed random function shows that if an adversary learns q evaluations of the keyed random function, the key entropy equals $|K|$ in the first one, $|K| - f_1$ in the second one, $|K| - f_2$ in the third one, and so on, where f_i denotes the accumulated gain over time.

The second exposition of accumulated leakage is a more complete one: in Section 4 we apply it to a nonce-based sponge-based stream encryption scheme called asakey. The asakey encryption mode is derived from the encryption part of ISAP [14,15]: it initializes a sponge state with a secret key, then it absorbs the nonce bit-by-bit to obtain a secret inner part of the sponge that is used "as a key" to a plain nonce-based sponge encryption mode with high rate. By doing a direct analysis of asakey and by in addition confiding on the accumulated leakage model, we obtain a bound that (i) is simpler than the one that would be obtained by relying on the general leakage resilience of the duplex [17] modularly, and that (ii) is more accurate in the leakage estimation.

1.4 Justification of Accumulated Leakage

Of course, a proper bound on the accumulated gain must still be computed, and this depends on the concrete implementation and the side channel and fault attacks an attacker can do. Simply said, the analysis so far is agnostic to the concrete attacks within the leakage parameters $f_1, f_2, \ldots$. While the naive a priori bound considering bounded leakage $f_i \leq i \cdot \lambda$ would work, a more accurate estimation should give a higher level of confidence. Such an estimation could be made based on dedicated analysis that depends on (i) the primitive that leaks, (ii) the scheme in which it leaks, and (iii) the type of side channel attack.

In Section 5, we perform an exemplary analysis of an implementation of the asakey scheme of Section 4 instantiated with the standardized Keccak-p[1600,12] permutation [33] as used by KangarooTwelve [7] and Keyak [6]. We analyze the implementation using the two attack vectors of differential power analysis (DPA) [25] and statistical ineffective fault attacks (SIFA) [12,13]. Those results not only provide an estimation of the expected loss of security when inserted in the asakey security bound of Theorem 2, but also show that a bounded leakage approach is often way too optimistic on how information of single leakages (experiments) can be combined.

As an example, have a look at Figure 3b in Section 5. The graph shows how security degrades with the number of queries assuming λ-bit leakage per query and how it typically degrades in an implementation attack which can also be expressed in accumulated leakage. It shows that a bound of $f_i \leq i \cdot \lambda$ is generally very loose. Taking different prior models, such as that of hard-to-invert leakage, would yield a line that is close to that of the bounded leakage model.

In order to get the best possible picture on the security in the accumulated leakage model, it is important to utilize the attack vectors as good as possible. Hence, e.g., work that aims to bound model errors in side channel attacks is also relevant in our context [10].

2 A Formalization of Implementation Attacks

Let k, m, n be three natural numbers. We denote by $\{0,1\}^n$ the set of n-bit strings and by $\{0,1\}^*$ the set of arbitrarily long strings. The set of families of 2^k n-bit permutations is denoted $\mathrm{perm}(k,n)$. If $k = 0$, we simply write $\mathrm{perm}(n)$. Similarly, the set of families of 2^k m-to-n-bit functions is denoted $\mathrm{func}(k,m,n)$. Again, if $k = 0$, we simply write $\mathrm{func}(m,n)$. If we consider functions with arbitrary domain and/or range, we replace m and/or n by $*$. We cheat and assume that this set is still finite, assuming that there is always an upper bound on length of arbitrary strings. If $m \leq n$, for string $X \in \{0,1\}^n$ we denote by $\mathrm{left}_m(X)$ the m leftmost bits of X and by $\mathrm{right}_m(X)$ the m rightmost bits. For a finite set $\mathcal{X}$, $X \overset{\$}{\leftarrow} \mathcal{X}$ denotes the event of uniformly randomly sampling of an element X from $\mathcal{X}$.

We will be concerned with adversaries A that are given access to one or more oracles O, and after interaction with O they output a decision bit b: $b \leftarrow \mathsf{A}^{\mathsf{O}}$. For two oracles O and P, the adversarial advantage of distinguishing the two is defined as

$$\Delta_{\mathsf{A}}\left(\mathsf{O}\ ;\ \mathsf{P}\right) = \left|\mathbf{Pr}\left(1 \leftarrow \mathsf{A}^{\mathsf{O}}\right) - \mathbf{Pr}\left(1 \leftarrow \mathsf{A}^{\mathsf{P}}\right)\right| . \tag{1}$$

2.1 Leakage Resilience

Let F be a cryptographic function with key size k. Let ro be a random oracle with the same interface as F_K. In a black-box scenario, one quantifies security of F as the advantage of an adversary A in distinguishing F_K for $K \xleftarrow{\$} \{0,1\}^k$ from a random oracle ro:

$$\mathbf{Adv}_{\mathsf{F}}^{\mathrm{bb}}(\mathsf{A}) = \Delta_{\mathsf{A}}\left(\mathsf{F}_K\ ;\ \mathsf{ro}\right) . \tag{2}$$

The adversary is typically bounded by a certain query complexity and time complexity (memory is usually not considered). A comparable definition occurs in the ideal primitive model. Suppose that F is based on a random function $\mathsf{f} \xleftarrow{\$} \mathrm{func}(m,n)$ for some natural numbers m,n. The adversary would, in addition to the oracle F_K or ro in (2), have access to the random function f:

$$\mathbf{Adv}_{\mathsf{F}}^{\text{i-bb}}(\mathsf{A}) = \Delta_{\mathsf{A}}\left(\mathsf{F}_K^{\mathsf{f}}, \mathsf{f}\ ;\ \mathsf{ro}, \mathsf{f}\right) . \tag{3}$$

The time complexity is then called primitive complexity and measures queries to f. If f would be a function with bi-directional interface, for example if it is drawn uniformly from $\mathrm{perm}(n)$, the adversary would have bi-directional access to it.

In the context of leakage resilient cryptography [2, 19–21, 36, 41, 47], the adversary gets access to a leaky version of the function F_K:

$$\mathsf{L}\left[\mathsf{F}_K\right] .$$

The function evaluates F_K as usual but in addition leaks certain secret information to the adversary. Now, the adversary has to distinguish the challenge oracle F_K from random ro as before, but *in addition* it gets access to $\mathsf{L}\left[\mathsf{F}_K\right]$. The resulting advantages are then expressed as

$$\mathbf{Adv}_{\mathsf{F}}^{\mathrm{lr}}(\mathsf{A}) = \Delta_{\mathsf{A}}\left(\mathsf{L}\left[\mathsf{F}_K\right], \mathsf{F}_K\ ;\ \mathsf{L}\left[\mathsf{F}_K\right], \mathsf{ro}\right) \tag{4}$$

for the standard model, and

$$\mathbf{Adv}_{\mathsf{F}}^{\text{i-lr}}(\mathsf{A}) = \Delta_{\mathsf{A}}\left(\mathsf{L}\left[\mathsf{F}_K^{\mathsf{f}}\right], \mathsf{F}_K^{\mathsf{f}}, \mathsf{f}\ ;\ \mathsf{L}\left[\mathsf{F}_K^{\mathsf{f}}\right], \mathsf{ro}, \mathsf{f}\right) . \tag{5}$$

for the ideal model. Naturally, the adversary is not allowed to query the leaky and the challenge oracle on identical inputs.

2.2 Accumulated Leakage

So far, we did not specify *how* leakage occurs in calls to $\mathsf{L}\left[\mathsf{F}_K\right]$. In the accumulated leakage model, we define the *accumulated gain* that represents the leakage, basically the entropy loss, that has occurred.

Generally, suppose that for a certain secret state that is input to a cryptographic primitive, the adversary has obtained q leakages. These are for r different inputs,

$$\boldsymbol{X} = (X_1, \ldots, X_r) \,,$$

occurring

$$\boldsymbol{q} = (q_1, \ldots, q_r)$$

times respectively. The incurred accumulated gain is defined as

$$\mathrm{AG}_{\mathtt{atk}}(\boldsymbol{X}, \boldsymbol{q}, r) \,,$$

where $\mathtt{atk} \in \{\mathtt{spa}, \mathtt{dpa}, \mathtt{sfa}, \ldots\}$ denotes the attack that the adversary performs.

We remark that the definition is purposely general: the model should apply to many different modes, types of leakages, and types of attacks. In particular, the indication of the attack type $\mathtt{atk}$ is important, as the adversarial advantage differs depending on the performed attack, which might be a DPA attack, a SIFA attack, or anything else. In Section 5, we estimate the function $\mathrm{AG}_{\mathtt{atk}}(\boldsymbol{X}, \boldsymbol{q}, r)$ for different attack types. The generality of our model comes at the price of strictness, which is unavoidable: original leakage resilience models (such as bounded leakage) are very well-defined but hard to apply in practice, whereas accumulated leakage is general but better to apply in practice.

As the name suggests, the leakage *accumulates*. For $i \in \{1, 2, \ldots\}$, we denote by r_i the amount of different inputs up to the i-th query, and likewise define $\boldsymbol{X}_i = (X_1, \ldots, X_{r_i})$ as the inputs, with occurrences $\boldsymbol{q}_i = (q_1, \ldots, q_{r_i})$ respectively.

3 Accumulated Leakage in Random Function Key Recovery

We consider a simple, isolated, security experiment, namely that of key recovery of a random function. This event would never occur in isolation, but it would rather be a part in a larger puzzle. We discuss this particular example as it clearly demonstrates how our accumulated leakage resilience concept occurs on a security proof.

Theorem 1. *Let k, m, n be three natural numbers. Let $\mathsf{rf} \xleftarrow{\$} \mathrm{func}(k, m, n)$ be a random function, and let $\mathsf{ro} \xleftarrow{\$} \mathrm{func}(m, n)$ be a random function. Let $\mathsf{F} \in \mathrm{func}(k, m, n)$ be defined as $\mathsf{F}^{\mathsf{rf}}_K(X) = \mathsf{rf}_K(X)$. For any adversary A with construction complexity q and primitive complexity p,*

$$\begin{aligned} \mathbf{Adv}^{\text{i-lr}}_{\mathsf{F}}(\mathsf{A}) &= \Delta_{\mathsf{A}}\left(\mathsf{L}\left[\mathsf{F}^{\mathsf{rf}}_K\right], \mathsf{F}^{\mathsf{rf}}_K, \mathsf{rf} \,;\, \mathsf{L}\left[\mathsf{F}^{\mathsf{rf}}_K\right], \mathsf{ro}, \mathsf{rf}\right) \\ &\le \sum_{i=1}^{p} \frac{1}{2^{k - \mathrm{AG}_{\mathtt{atk}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)}} \,, \end{aligned}$$

where $\boldsymbol{X}_i$ denotes the inputs of the queries made to $\mathsf{L}\left[\mathsf{F}^{\mathsf{rf}}_K\right]$ up to primitive query i, $\boldsymbol{q}_i$ their occurrences, and r_i the length of these tuples.

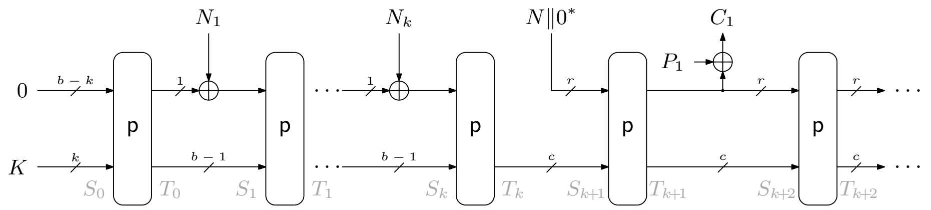


Fig. 1: Encryption scheme. The state parameters (S_i, T_i) will be used in the proof of Theorem 2.

Proof. It is easy to see that the two oracles are perfectly indistinguishable as long as the adversary never queries its unkeyed primitive rf on key K. Consider any primitive query (K_i, X_i) for $i \in \{1, \ldots, p\}$. Suppose that, prior to this i-th primitive query, the adversary has queried its construction oracle for r_i different inputs, namely $\boldsymbol{X}_i$, each occurring $\boldsymbol{q}_i$, respectively. Then, the probability that $K_i = K$ is at most

$$\frac{1}{2^{k-\mathrm{AG}_{\mathtt{atk}}(\boldsymbol{X}_i,\boldsymbol{q}_i,r_i)}} .$$

The bound is obtained by summing over $i = 1, \ldots, p$. □

As we can see from Theorem 1, the attacker is in principle able to utilize the leakage of all previous operations in his attack. This means that both $\max(r_i)$ and $\max(\boldsymbol{q}_i)$ are bounded by the number of primitive queries p. Although this shows that accumulated leakage lets us nicely describe how a side channel attack on an instantiation of a random function, e.g., AES-PRF [32], behaves, we cannot draw any conclusion on the leakage resilience on this level of abstraction.

4 Accumulated Leakage in Sponge-Based Stream Encryption

We consider a nonce-based sponge-based stream encryption called asakey. The scheme is a slight variant of the encryption part of ISAP [14,15], using a bitwise absorption of the nonce similarly as [43]. The asakey encryption scheme is parameterized by natural numbers k, b, c, r, where $k \le \min\{c, r\}$ and $c+r = b$, and it is based on a cryptographic permutation $\mathsf{p} \in \mathrm{perm}(b)$. Instead of initializing the sponge state with a nonce and a key, it first processes the nonce bit-wise in order to obtain a secret state that functions "as a key" (hence the name). The asakey encryption scheme is specified in Algorithm 1 and depicted in Figure 1. As it is a stream encryption scheme, the decryption is identical with P and C swapped.

4.1 Security

A variation of this scheme was already proven leakage resilient by Dobraunig and Mennink [17], but in that work, the result was derived as a corollary of

Algorithm 1 asakey encryption scheme

```
Input: $(K, N, P) \in \{0,1\}^k \times \{0,1\}^k \times \{0,1\}^*$
Output: $C \in \{0,1\}^{|P|}$
  $S \leftarrow \mathsf{p}(0^{b-k} \| K)$
  $N_1 \| \ldots \| N_k \leftarrow N$
  for $i = 1, \ldots, k$ do
      $S \leftarrow \mathsf{p}(S \oplus N_i \| 0^{b-1})$
  $S \leftarrow N \| 0^{b-c-k} \| \mathrm{right}_c(S)$
  $Z \leftarrow \varnothing$
  while $|Z| < |P|$ do
      $S \leftarrow \mathsf{p}(S)$
      $Z \leftarrow Z \| \mathrm{left}_r(S)$
  return $\mathrm{left}_{|P|}(P \oplus Z)$
```

the leakage resilience of the duplex, a versatile permutation-based cryptographic mode. By now performing a *direct analysis*, we obtain a simpler bound and we also more clearly demonstrate how the accumulated leakage model can be used.

The result uses the notion of the multicollision limit function from Daemen et al. [11]. For natural numbers q, c, r, consider the experiment of throwing q balls uniformly at random in 2^r bins, and let μ be the maximum number of balls in a single bin. The multicollision limit function $\nu_{r,c}^q$ is defined as the smallest natural number ν such that

$$\mathbf{Pr}\left(\mu > \nu\right) \leq \frac{\nu}{2^c} \,. \tag{6}$$

Daemen et al. [11] analyzed $\nu_{r,c}^q$ in detail. As a rule of thumb [17], the term behaves as follows:

$$\nu_{r,c}^q \lesssim \begin{cases} (r+c)/\log_2\left(\frac{2^r}{q}\right), \text{ for } q \lesssim 2^r \,, \\ (r+c) \cdot \frac{q}{2^r}, \text{ for } q \gtrsim 2^r \,. \end{cases}$$

We are now ready to state security of the stream encryption algorithm asakey of Algorithm 1. The proof is given in Section 4.2.

Theorem 2. *Let k, b, c, r be four natural numbers such that $k \leq \min\{c, r\}$ and $c+r = b$. Consider asakey of Algorithm 1 instantiated with a random permutation $\mathsf{p} \xleftarrow{\$} \mathrm{perm}(b)$, and let $\mathsf{ro} \xleftarrow{\$} \mathrm{func}(*,*)$ be a random function. For any adversary A with construction complexity q, total number of encrypted message blocks Q, and with primitive complexity p,*

$$\begin{aligned}\mathbf{Adv}_{\mathsf{asakey}}^{\text{i-lr}}(\mathsf{A}) &= \Delta_{\mathsf{A}}\left(\mathsf{L}\left[\mathsf{asakey}_K^{\mathsf{p}}\right], \mathsf{asakey}_K^{\mathsf{p}}, \mathsf{p}^{\pm} \;;\; \mathsf{L}\left[\mathsf{asakey}_K^{\mathsf{p}}\right], \mathsf{ro}, \mathsf{p}^{\pm}\right) \\ &\leq \sum_{i=1}^{p}\left(\frac{1}{2^{k-\mathrm{AG}_{\mathsf{atk}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)}} + \frac{\nu_{r,c}^{Q-q}+1}{2^{c-\mathrm{AG}_{\mathsf{atk}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)}} + \frac{Q+2qk+1}{2^{b-\mathrm{AG}_{\mathsf{atk}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)}}\right) \\ &\quad + \frac{(Q+2qk)q + 2\nu_{r,c}^{Q-q}}{2^c} + \frac{\binom{Q+2qk+1}{2} + \binom{Q+qk+1+p}{2} + \binom{p}{2}}{2^b} \,,\end{aligned}$$

where $\boldsymbol{X}_i$ denotes the primitive evaluations in all queries made to $\mathsf{L}\left[\mathsf{asakey}_K^{\mathsf{p}}\right]$ up to primitive query i with the same inner part as primitive query i, $\boldsymbol{q}_i$ their occurrences, and r_i the length of these tuples.

Simply speaking, the goal of our scheme is to limit the input complexity per evaluation of p with the same secret state. As we will detail in Section 4.2, for all i, each X_j contained in $\boldsymbol{X_i}$ of $\mathrm{AG}_{\mathtt{atk}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)$ shares the same inner part $\mathrm{right}_c(X_j)$. As long as no bad event occurs (specified in the proof), all permutation queries within all construction queries have a differing inner part $\mathrm{right}_c(T_j)$. Since the nonce N is required to be unique (proper evaluations of the mode within an authenticated encryption scheme prevent an attacker to be able to decrypt on new data) and either is absorbed bitwise or as a whole, it follows that $r_i \leq 2$. Hence, the input complexity per permutation evaluation within our scheme is effectively bounded to 2. We evaluate the effects of different bounds on r_i for concrete implementation attacks in Section 5.

4.2 Proof of Theorem 2

Write $\mathsf{E} = \mathsf{asakey}$ for brevity. Without loss of generality, all constructions queries that A makes are for plaintext 0^*; all that matters is the *length* of the queries. So the adversary can make q construction queries of the form (N_j, ℓ_j, Z_j), where $N_j \in \{0,1\}^k$ denotes the nonce, $\ell_j \in \mathbb{N}$ the requested number of key stream blocks, and $Z_j \in (\{0,1\}^r)^{\ell_j}$ the resulting key stream. All nonces N_j are unique, and the lengths ℓ_j sum to Q. For each query j, we define $S_{j,l}$ and $T_{j,l}$ for $l = 0, \ldots, k+\ell_j$ as indicated in Figure 1. Evaluations of $\mathsf{L}\left[\mathsf{E}_K^{\mathsf{p}}\right]$ may leak information upon every evaluation of p, i.e., of every transition from $S_{j,l}$ to $T_{j,l}$. The adversary can additionally make p direct queries to $\mathsf{p}^{\pm}$, which are denoted (X_i, Y_i). Recall that for query i, we defined $\mathrm{AG}_{\mathtt{atk}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)$ as the accumulated gain up to query i, where $\boldsymbol{X}_i$ denotes the primitive evaluations in all queries made to $\mathsf{L}\left[\mathsf{E}_K^{\mathsf{p}}\right]$ up to primitive query i with the same inner part $\mathrm{right}_c(X_i)$, $\boldsymbol{q}_i$ their occurrences, and r_i the length of these tuples.

Our goal is to bound

$$\Delta_{\mathsf{A}}\left(\mathsf{L}\left[\mathsf{E}_K^{\mathsf{p}}\right], \mathsf{E}_K^{\mathsf{p}}, \mathsf{p}^{\pm}\ ;\ \mathsf{L}\left[\mathsf{E}_K^{\mathsf{p}}\right], \mathsf{ro}, \mathsf{p}^{\pm}\right)\,, \tag{7}$$

As Dobraunig and Mennink [17], we first replace p with a function $\mathsf{f} : \{0,1\}^b \to \{0,1\}^b$ that has the same interface, and that simply returns uniform random responses lazily, and aborts if there is a collision. Formally, f maintains an initially empty list $\mathcal{F}$ in which its input-output tuples will be stored. For a query $\mathsf{f}(X)$ with $(X, \cdot) \notin \mathcal{F}$, the function f generates $Y \xleftarrow{\$} \{0,1\}^b$ and returns this value. If $(\cdot, Y) \in \mathcal{F}$, it aborts; otherwise, it stores (X, Y) in $\mathcal{F}$. It operates symmetrically for inverse queries. Clearly, f and p are perfectly indistinguishable as long as the former does not abort, and hence,

$$\Delta_{\mathsf{A}}\left(\mathsf{L}\left[\mathsf{E}_K^{\mathsf{p}}\right], \mathsf{E}_K^{\mathsf{p}}, \mathsf{p}^{\pm}\ ;\ \mathsf{L}\left[\mathsf{E}_K^{\mathsf{f}}\right], \mathsf{E}_K^{\mathsf{f}}, \mathsf{f}^{\pm}\right) \leq \frac{\binom{Q+qk+1+p}{2}}{2^b}\,,$$

$$\Delta_{\mathsf{A}}\left(\mathsf{L}\left[\mathsf{E}_K^{\mathsf{p}}\right], \mathsf{ro}, \mathsf{p}^{\pm}\ ;\ \mathsf{L}\left[\mathsf{E}_K^{\mathsf{f}}\right], \mathsf{ro}, \mathsf{f}^{\pm}\right) \leq \frac{\binom{p}{2}}{2^b}\,,$$

noting that in the former case p is evaluated a total amount of at most $Q+qk+1+p$ times and in the latter case it is evaluated at most p times. As in [17], the transition is a purely probabilistic issue, and it does not concern leakage. We get

$$(7) \leq \varDelta_{\mathsf{A}}\left(\mathsf{L}\left[\mathsf{E}_K^{\mathsf{f}}\right], \mathsf{E}_K^{\mathsf{f}}, \mathsf{f}^{\pm} \; ; \; \mathsf{L}\left[\mathsf{E}_K^{\mathsf{f}}\right], \mathsf{ro}, \mathsf{f}^{\pm}\right) + \frac{\binom{Q+qk+1+p}{2} + \binom{p}{2}}{2^b} \,. \tag{8}$$

To analyze the remaining distance of (8), we define three bad events, where $\nu = \nu_{r,c}^{Q-q}$ is defined using the multicollision limit function:

$$\mathsf{bad_{cc}} \; : \; \exists \text{ distinct and non-trivial } (j,l),(j',l') \text{ such that } S_{j,l} = S_{j',l'} \,, \tag{9}$$

$$\mathsf{bad_{cp}} \; : \; \exists\ (j,l), i \text{ such that } S_{j,l} = X_i \text{ or } T_{j,l} = Y_i \,, \tag{10}$$

$$\mathsf{bad_{mcS}} \; : \; \exists \text{ distinct } (j,l)_1, \ldots, (j,l)_{\nu+1} \text{ with } l_1, \ldots, l_{\nu+1} \geq k+2 \text{ such that } \mathrm{left}_r(S_{(j,l)_1}) = \cdots = \mathrm{left}_r(S_{(j,l)_{\nu+1}}) \,, \tag{11}$$

$$\mathsf{bad_{mcT}} \; : \; \exists \text{ distinct } (j,l)_1, \ldots, (j,l)_{\nu+1} \text{ with } l_1, \ldots, l_{\nu+1} \geq k+2 \text{ such that } \mathrm{left}_r(T_{(j,l)_1}) = \cdots = \mathrm{left}_r(T_{(j,l)_{\nu+1}}) \,. \tag{12}$$

For $\mathsf{bad_{cc}}$, two tuples are *non-trivial* if either $l \neq l'$ or $l = l' \geq \mathrm{argmin}_{\iota}(N_\iota \neq N'_\iota)$. This condition is needed to exclude trivial collisions for different queries whose nonces share a prefix. The bad events $\mathsf{bad_{mcS}}$ and $\mathsf{bad_{mcT}}$ are strictly seen not needed to bound the remaining distance of (8), $\mathsf{bad_{cc}}$ and $\mathsf{bad_{cp}}$ alone suffice. Yet, they will be used to actually *bound the occurrences of* $\mathsf{bad_{cc}}$ *and* $\mathsf{bad_{cp}}$.

We write $\mathsf{bad_{mc}} = \mathsf{bad_{mcS}} \vee \mathsf{bad_{mcT}}$ and $\mathsf{bad} = \mathsf{bad_{cc}} \vee \mathsf{bad_{cp}} \vee \mathsf{bad_{mc}}$. One can note that, as long as $\neg\mathsf{bad}$ holds, the stream generation part of any evaluation of E^{f} consists of "fresh" evaluations of f, i.e., evaluations that are not defined by $\mathcal{F}$ yet. This means that their responses are randomly drawn from $\{0,1\}^b$, and that the resulting key stream $Z_j \in (\{0,1\}^r)^{\ell_j}$ is perfectly indistinguishable from random. Therefore, we obtain for the remaining distance of (8):

$$\varDelta_{\mathsf{A}}\left(\mathsf{L}\left[\mathsf{E}_K^{\mathsf{f}}\right], \mathsf{E}_K^{\mathsf{f}}, \mathsf{f}^{\pm} \; ; \; \mathsf{L}\left[\mathsf{E}_K^{\mathsf{f}}\right], \mathsf{ro}, \mathsf{f}^{\pm}\right) \leq \mathbf{Pr}\left(\mathsf{A}^{\mathsf{L}\left[\mathsf{E}_K^{\mathsf{f}}\right], \mathsf{E}_K^{\mathsf{f}}, \mathsf{f}^{\pm}} \text{ sets } \mathsf{bad}\right) \,. \tag{13}$$

The remaining probability is bounded by Lemma 1 below. The proof of Theorem 2 is completed by combining this lemma with (7), (8), and (13).

Lemma 1. *We have*

$$\mathbf{Pr}\left(\mathsf{A}^{\mathsf{L}\left[\mathsf{E}_K^{\mathsf{f}}\right], \mathsf{E}_K^{\mathsf{f}}, \mathsf{f}^{\pm}} \textit{ sets } \mathsf{bad}\right) \leq \sum_{i=1}^{p}\left(\frac{1}{2^{k-\mathrm{AG}_{\mathrm{atk}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)}} + \frac{\nu+1}{2^{c-\mathrm{AG}_{\mathrm{atk}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)}} + \frac{Q+2qk+1}{2^{b-\mathrm{AG}_{\mathrm{atk}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)}}\right) + \frac{(Q+2qk)q+2\nu}{2^c} + \frac{\binom{Q+2qk+1}{2}}{2^b} \,.$$

Proof. For brevity, write

$$\mathbf{Pr}\left(\mathsf{bad}\right) := \mathbf{Pr}\left(\mathsf{A}^{\mathsf{L}[\mathsf{E}^{\mathsf{f}}_{K}],\mathsf{E}^{\mathsf{f}}_{K},\mathsf{f}^{\pm}} \text{ sets } \mathsf{bad}\right)\,, \tag{14}$$

and likewise for the sub-events $\mathsf{bad}_{\mathsf{cc}}$, $\mathsf{bad}_{\mathsf{cp}}$, and $\mathsf{bad}_{\mathsf{mc}}$. By basic probability theory,

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{bad}\right) &= \mathbf{Pr}\left(\mathsf{bad}_{\mathsf{cc}} \vee \mathsf{bad}_{\mathsf{cp}} \vee \mathsf{bad}_{\mathsf{mc}}\right) \\ &\leq \mathbf{Pr}\left(\mathsf{bad}_{\mathsf{cc}} \vee \mathsf{bad}_{\mathsf{cp}} \mid \neg\mathsf{bad}_{\mathsf{mc}}\right) + \mathbf{Pr}\left(\mathsf{bad}_{\mathsf{mc}}\right)\,.\end{aligned}$$

In fact, following Dobraunig and Mennink [17], we will introduce one layer of granularity in the reasoning. Note that the adversary can trigger a total amount of at most $Q + qk + 1 + p$ primitive evaluations. These are all evaluated in some sequential order. For $\alpha = 1, \ldots, Q + qk + 1 + p$, one can write $\mathsf{bad}_{\mathsf{x}}(\alpha)$ as the event that the α-th query triggers the particular event $\mathsf{bad}_{\mathsf{x}}$. Then,

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{bad}\right) \leq &\sum_{\alpha=1}^{Q+qk+1+p} \mathbf{Pr}\left(\mathsf{bad}_{\mathsf{cc}}(\alpha) \mid \neg\mathsf{bad}(1 \ldots \alpha-1) \wedge \neg\mathsf{bad}_{\mathsf{mc}}(\alpha)\right) &\text{(15a)}\\ + &\sum_{\alpha=1}^{Q+qk+1+p} \mathbf{Pr}\left(\mathsf{bad}_{\mathsf{cp}}(\alpha) \mid \neg\mathsf{bad}(1 \ldots \alpha-1) \wedge \neg\mathsf{bad}_{\mathsf{mc}}(\alpha)\right) &\text{(15b)}\\ + &\ \mathbf{Pr}\left(\mathsf{bad}_{\mathsf{mc}}\right)\,. &\text{(15c)}\end{aligned}$$

We will henceforth proceed the same way as [17]. We will consider any of the $Q + qk + 1 + p$ evaluations of f that might happen, and analyze the probability that this particular query α sets $\mathsf{bad}_{\mathsf{cc}}(\alpha)$ or $\mathsf{bad}_{\mathsf{cp}}(\alpha)$, assuming that the events were not set prior to this query. One can also adopt a similar reasoning for $\mathsf{bad}_{\mathsf{mc}}$, but for that event, instead, a direct reasoning is more convenient.

Probability that a query sets $\mathsf{bad}_{\mathsf{cc}}$ *(Eq. (15a)).* Consider any two distinct and non-trivial queries (j, l) and (j', l'). Without loss of generality, (j', l') is the newer one, i.e., either $j < j'$ or $(j = j'$ and $l < l')$. We will consider the probability that any new construction query hits an older tuple. Note that $S_{j,0} = 0^{b-k} \| K$ for all j, and this is the first evaluation of f made in construction queries. We have the following cases:

- $l' = 0$. This case would imply that $j' = 1$ and (j', l') cannot be the newer query;
- $l' \in \{1, \ldots, k\}$. In this case, $S_{j',l'}$ is randomly generated and it hits any older state value with $l \neq k + 1$ with probability $1/2^b$; The case of $l = k + 1$ is different: the adversary might have set a nonce as input that matches the leakage that it might have obtained from an earlier evaluation of $S_{j',l'-1}$ (noting that although (j', l') is newer than (j, l), $(j', l' - 1)$ might predate it). In this case, yet, a collision happens with probability at most $1/2^c$;
- $l' = k + 1$. In this case, the attacker has set the outer part, and the state equals any older state with probability $1/2^c$;

– $l' \in \{k+2, k+3, \dots\}$. In this case, $S_{j',l'}$ is randomly generated and it hits any older state value with probability $1/2^b$.

There is 1 unique state with $l = 0$, qk states with $l \in \{1, \dots, k\}$, q states with $l = k+1$, and $Q - q$ states with $l \in \{k+2, k+3 \dots\}$. By summing over all possible choices of distinct (j, l) and (j', l'), we obtain

$$
\begin{aligned}
(15a) \leq & \left(\frac{qk \cdot (1 + \frac{qk-1}{2} + (Q-q))}{2^b} + \frac{qk \cdot q}{2^c} \right) \\
& + \frac{q \cdot (1 + qk + \frac{q-1}{2} + (Q-q))}{2^c} + \frac{(Q-q) \cdot (1 + qk + q + \frac{Q-q-1}{2})}{2^b} \\
\leq & \frac{(Q+2qk)q}{2^c} + \frac{\binom{Q+2qk+1}{2}}{2^b} \, . \qquad (16)
\end{aligned}
$$

Probability that a query sets $\mathsf{bad}_{\mathsf{cp}}$ *(Eq. (15b)).* Consider any construction query (j, l) and any primitive query i. Note that, upon the i-th primitive query, the accumulated gain for states with the same inner part $\mathrm{right}_c(X_i)$ is defined as $\mathrm{AG}_{\mathsf{atk}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)$. This bound represents the entropy loss of guessing that state value. We make the following distinction:

– $l = 0$. Note that $S_{j,0} = 0^{b-k} \| K$ for all j, so there is only one occurrence of this construction query. If the primitive query is a forward query, without loss of generality its first $b-k$ bits are 0. It satisfies $S_{j,0} = X_i$ with probability at most $1/2^{k-\mathrm{AG}_{\mathsf{atk}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)}$. It satisfies $T_{j,0} = Y_i$ with probability $1/2^b$. On the other hand, if the primitive query is an inverse query, it satisfies $T_{j,0} = Y_i$ with probability at most $1/2^{b-\mathrm{AG}_{\mathsf{atk}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)}$ and $S_{j,0} = X_i$ with probability $1/2^b$. Therefore, restricted to the case $l = 0$ (which is just 1 unique construction query), the event happens with probability at most

$$
\frac{1}{2^{k-\mathrm{AG}_{\mathsf{atk}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)}} + \frac{1}{2^{b-\mathrm{AG}_{\mathsf{atk}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)}} \, ;
$$

– $l \in \{1, \dots, k\}$. Note that this involves qk construction queries, but the adversary knows nothing about these states, besides leakage. If the primitive query is a forward query, it satisfies either of $S_{j,l} = X_i$ and $T_{j,l} = Y_i$ with probability at most $1/2^{b-\mathrm{AG}_{\mathsf{atk}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)}$. For inverse queries, the analysis is identical. Therefore, restricted to the case $l \in \{1, \dots, k\}$ (which are at most qk construction queries), the event happens with probability at most

$$
\frac{2qk}{2^{b-\mathrm{AG}_{\mathsf{atk}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)}} \, ;
$$

– $l = k+1$. Note that this involves q construction queries, and the adversary might have set the state to a value $N \| 0^*$. If the primitive query is a forward query, there is at most one construction query whose first r bits are equal to the first r bits of X. It satisfies $S_{j,k+1} = X_i$ for that particular state with probability at most $1/2^{c-\mathrm{AG}_{\mathsf{atk}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)}$. It satisfies $T_{j,k+1} = Y_i$ with probability $1/2^b$. On the other hand, if the primitive query is an inverse query, there is at most one construction query whose first r bits are equal to the first r bits of Y. It satisfies $T_{j,k+1} = Y_i$ for that particular state with probability at most $1/2^{c-\mathrm{AG}_{\mathsf{atk}}(\boldsymbol{X}_i,\boldsymbol{q}_i,r_i)}$ and $S_{j,k+1} = X_i$ with probability $1/2^b$. Therefore, restricted to the case $l = k+1$ (which are q construction queries), the event happens with probability at most

$$\frac{1}{2^{c-\mathrm{AG}_{\mathsf{atk}}(\boldsymbol{X}_i,\boldsymbol{q}_i,r_i)}} + \frac{q}{2^{b-\mathrm{AG}_{\mathsf{atk}}(\boldsymbol{X}_i,\boldsymbol{q}_i,r_i)}}\,;$$

– $l \in \{k+2, k+3, \dots\}$. Note that this involves $Q-q$ construction queries. If the primitive query is a forward query, by $\neg\mathsf{bad}_{\mathsf{mcS}}$ there are at most ν possible construction queries with $\mathrm{left}_r(S_{j,l}) = \mathrm{left}_r(X_i)$. It satisfies $S_{j,l} = X_i$ for any of these ν states with probability at most $1/2^{c-\mathrm{AG}_{\mathsf{atk}}(\boldsymbol{X}_i,\boldsymbol{q}_i,r_i)}$. It satisfies $T_{j,l} = Y_i$ with probability $1/2^b$. For inverse queries, the analysis is identical, relying on $\neg\mathsf{bad}_{\mathsf{mcT}}$. Therefore, restricted to the case $l \in \{k+2, k+3, \dots\}$ (which are at most $Q-q$ construction queries), the event happens with probability at most

$$\frac{\nu}{2^{c-\mathrm{AG}_{\mathsf{atk}}(\boldsymbol{X}_i,\boldsymbol{q}_i,r_i)}} + \frac{Q-q}{2^{b-\mathrm{AG}_{\mathsf{atk}}(\boldsymbol{X}_i,\boldsymbol{q}_i,r_i)}}\,.$$

By summing over all possible types of queries, we obtain

$$(15b) \le \sum_{i=1}^{p} \left(\frac{1}{2^{k-\mathrm{AG}_{\mathsf{atk}}(\boldsymbol{X}_i,\boldsymbol{q}_i,r_i)}} + \frac{\nu+1}{2^{c-\mathrm{AG}_{\mathsf{atk}}(\boldsymbol{X}_i,\boldsymbol{q}_i,r_i)}} + \frac{Q+2qk+1}{2^{b-\mathrm{AG}_{\mathsf{atk}}(\boldsymbol{X}_i,\boldsymbol{q}_i,r_i)}} \right)\,. \tag{17}$$

Probability that a query sets $\mathsf{bad}_{\mathsf{mc}}$ *(Eq. (15c)).* Recall that $\mathsf{bad}_{\mathsf{mc}} = \mathsf{bad}_{\mathsf{mcS}} \vee \mathsf{bad}_{\mathsf{mcT}}$. We start with $\mathsf{bad}_{\mathsf{mcT}}$. The state values $T_{i,j}$ are randomly generated using a random function, and $Q-q$ values are generated. Thus, $\mathsf{bad}_{\mathsf{mcT}}$ is a balls-and-bins experiment with $Q-q$ balls thrown into 2^r bins. The event $\mathsf{bad}_{\mathsf{mcT}}$ is set for the $T_{i,j}$'s if there is a bin with more than $\nu = \nu_{r,c}^{Q-q}$ balls. By (6), this happens with probability at most $\nu/2^c$. The analysis of $\mathsf{bad}_{\mathsf{mcS}}$ is symmetric, noting that $S_{i,j+1}$ has the same distribution as $T_{i,j}$ for $j \neq 0, k+1$. Thus,

$$(15c) \le \frac{2\nu}{2^c}\,. \tag{18}$$

Conclusion. The proof is completed by a direct combination of (14), (15), (16), (17), and (18). □

4.3 On the Usage with Bounded Leakage

Our approach of accumulated leakage is backwards compatible with bounded leakage. In other words, our model allows to provide a bound for leakage in the spirit of bounded leakage. Let us consider asakey and bound the leakage to λ bits per execution of the underlying permutation. Then, we get for non-adaptive bounded leakage $\mathrm{AG}_{\texttt{bounded}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i) = (\max(r_i) + 1)\lambda = 3\lambda$, where the addition of 1 comes from the fact that we have one matching Y to all X with the same inner part that can leak. In the case of adaptive bounded leakage, we get $\mathrm{AG}_{\texttt{bounded}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i) = (2\max(\boldsymbol{q}_i))\lambda \leq 2q\lambda$, where the multiplication by 2 comes also from the fact that each leaking input X has one leaking output Y of a permutation.

Note that bounded leakage puts restrictions on the leakage function, and hence is only valid for side channel attacks. Our way of modeling stays valid even in the case of fault attacks. More generally, one of the significant ideas in accumulated leakage is to provide a leakage model that allows for direct compatibility with practical side-channel and fault evaluations of implementations of primitives. Hence, by knowing the bounds on the data complexity imposed by the leakage resilient mode, an evaluation of the security boils down to focusing on attacks on the isolated implemented primitive. The results of these attacks then paint a clear picture of the security of the investigated implementation under the current attacks. In addition, by increasing the data complexity beyond what would be covered by the proven bound gives an indication of how much more data would be needed for a break. This leaves an implementation with some kind of security margin that is comparable to the security margin we have in the cryptanalysis of (round reduced) symmetric primitives. In particular, it gives a rough estimate of how much an attacker has to improve its techniques until it breaks the implementation. We will show the employment of this concept in the following section.

5 Practical Results on Accumulated Leakage

In this section, we exemplify the usability of the model of Section 2. We give practical results on the amount of information an attacker can get about the secret permutation state in asakey. To make the evaluation less time consuming, we assume that the secret part is always 128 bits and that the attacker is in control of the rest of the input bits and knows the output except 128 bits. Clearly, such a simplified scenario strongly favors the attacker and, as a consequence, the achieved results drastically underestimate the actual security of asakey, which is fine as long as we can show that the simpler construction is already sufficiently hard to attack. For a more thorough evaluation, more experiments on all the possible configurations of the permutation (see Figure 2) would need to be done.

We evaluate the information an attacker can extract via multiple different attack vectors. For each attack vector, we get a certain guessing advantage that an attacker can achieve using a certain amount of measurements. Our construction in Section 4 efficiently bounds the data complexity ($\max(r_i) = 2$) an attacker is able to exploit per secret state it tries to recover. Hence, for attacks like statistical fault attacks (SFA), statistical ineffective fault attacks (SIFA), or differential power analysis (DPA), that perform better with increasing data complexity, we effectively end up with security margin. In other words, we get some impression on how much an attacker has to improve over our already optimistically performed attacks.

In essence, this is a situation that is well-known in symmetric cryptography, where round-reduced variants of symmetric primitives are attacked and the security margin up to the full round version gives some measure how much an attacker has to improve to threaten the whole scheme. As in our case here, there is usually also no guarantee in symmetric cryptography that a certain attack vector (e.g., differential cryptanalysis [8]) is the best attack vector on a certain scheme, nor that one cannot exploit a certain attack vector better than previously demonstrated.

Note that our construction can limit $\max(\boldsymbol{q}_i)$, the number of calls to the permutation with the same input, only during the key stream generation phase of encryption operations by using fresh nonces. During decryption, or the rekeying phase of encryption operations, the attacker can observe permutation calls with constant inputs that our construction can only loosely limit by $\max(\boldsymbol{q}_i) \leq q$. Consequently, asakey does not provide mode-level protection against certain attacks that naturally improve by repeated measurements of the same input to reduce noise, like simple power analysis (SPA) and template attacks [28]. Protection against such attacks thus has to be achieved by either (i) using a stateful mode that can effectively limit $\max(\boldsymbol{q}_i)$, or (ii) using implementation countermeasures against only those attacks that cannot be prevented on mode-level. Stateful modes can be built by employing techniques such as [36,40] where sender and receiver operate on a "synchronized" state and hence, are able to limit $\max(\boldsymbol{q}_i)$ to a small constant, e.g., 2 or 4. If this is not an option then one could use asakey and only employ rather simple countermeasures like first-order masking or shuffling to deal with SPA and template attacks.

At the same time, asakey can still offer some protection against fault attacks that utilize different faults on executions with the same input, like differential fault attacks (DFA) [9] since DFA needs to be mounted on the key stream generation phase and (i) fresh nonces prevent the repeated usage of keys during encryption, and (ii) asakey's hard-to-invert key derivation significantly increases the difficulty DFA-based key recovery attacks during decryption (see Section 5.5).

In the first part of this section, we discuss the simplified attack scenario that we subsequently use to optimistically estimate the effectiveness of various kinds of implementation attacks on asakey. We then present concrete attack evaluations, starting with attacks that are limited by $\max(r_i)$, and interpret the influence of these attacks on our bound. At the end, we discuss attacks that are limited by $\max(\boldsymbol{q}_i)$.

5.1 Primitive Under Attack

We will give an impression on how $\mathrm{AG}_{\mathtt{dpa}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)$, and $\mathrm{AG}_{\mathtt{sifa}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)$ of Theorem 2 behave. Before doing so, we have to settle the implementation that we consider. We assume an implementation that for each new encryption always starts with the first permutation call that involves the key. Hence, we know from

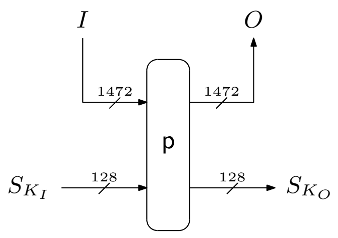


Fig. 2: The Keccak-p permutation as scrutinized in current analysis.

Section 4.1 that for such an implementation, $\max(r_i) = 2$ and $\max(\boldsymbol{q}_i) \leq q$. In order to limit the amount of experiments we have to do, we can perform all experiments with the Keccak-p[1600,12] permutation in a best-case configuration for an attacker as shown in Figure 2.

In this setting, the goal of an attacker is to recover as much information as possible about the secret states, which are 128 bits placed at the last two lanes of Keccak-p[1600,12]. The attacker has ultimate control over the 1472-bit input I and is able to observe O. Please note that the degrees of freedom available to an attacker in the scenario shown in Figure 2 are always strictly better compared to our mode in Figure 1. Hence, we consider experiments based on the configuration shown in Figure 2 to be a valid estimation of the security we can expect from our mode shown in Figure 1.

For DPA and SIFA, the goal is to recover the secret state S_{K_I} at the input of the permutation, and we show how $\mathrm{AG}_{\mathtt{atk}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)$ changes for increasing r_i. We do not take advantage in the attack that due to the construction of our scheme, there exists also a previous permutation call, where the inner part of this previous call $S_{K_O}^{\mathrm{prev}}$ matches the inner part of the next permutation call S_{K_I}, simply because it is unclear how to take this into account if in a scheme all permutations are only evaluated in forward direction. Also note that for fault attacks, the data complexity at the input for some secret state S_{K_I} is not strictly bounded by r_i. An attacker could artificially increase the data complexity by placing additional faults, e.g., at the input of the permutation and exploit that in this case $\max(\boldsymbol{q}_i)$ is not strictly bounded. However, this requires a quite powerful attacker and also complicates and often prohibits the detection on whether a fault was ineffective or not.

In an implementation attack, we usually end up with a list of candidates for the secret bits we aim to recover. In the spirit of Selçuk [38], we express the guessing advantage in bits as $\log_2$ (#total candidates) $-$ $\log_2$ (#candidates to test), where the number of candidates to test are the candidates states, which are ranked equal or higher compared to the correct secret state.

5.2 Differential Power Analysis

In our power analysis experiment, we consider a DPA attack on a microprocessor implementation of the Keccak-p[1600,12] permutation. The target software implementation is the AVR assembler optimized version of Keccak-p from the Extended Keccak Code Package (XKCP) [5]. The power measurements were conducted on a ChipWhisperer-Lite side channel evaluation board [34] featuring an XMEGA 128D4 microprocessor as the victim. As described in the previous section we assume that the initial state of the permutation is initialized with a 128-bit key that is located in the last two lanes, the remaining input bits are under the control of the attacker.

During the experiment we send random inputs I to the device and measure the power consumption of the following Keccak-p[1600,12] permutation. The resulting output O is not needed in this attack. The strategy of the power analysis follows the principles presented by Taha and Schaumont [42] who thoroughly analyzed different DPA attack strategies for various instantiations of MAC-Keccak schemes. In our case each column of the state contains at most one key bit at the beginning of the permutation. We can hence use a rather simple strategy based on the prediction of the Hamming weight of intermediate values during the computation of θ in the first round. In software, θ is usually implemented in two steps. First, the parity of each column in the state $S(x, y, z)$ is calculated which results in θ_{plane}:

$$\theta_{\text{plane}}(x, z) = \bigoplus_{i=0}^{4} S(x, i, z)\,.$$

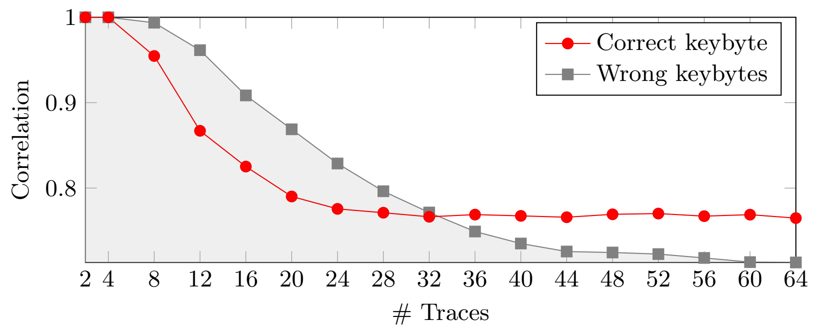
The output of θ then corresponds to the XOR of every state bit with 2 bits from θ_{plane}:

$$S(x, y, z) = S(x, y, z) \oplus \theta_{\text{plane}}(x-1, z) \oplus \theta_{\text{plane}}(x+1, z-1)\,.$$

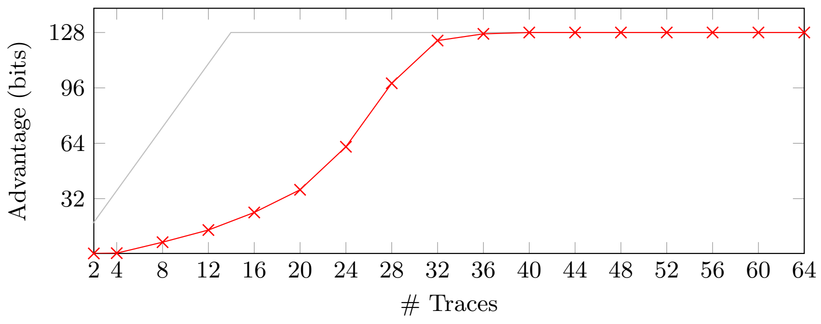
In our experiment we focus on the values of the last two lanes of θ_{plane} since only these have key bits in the corresponding sheet. While predicting values of entire 64-bit lanes is not really practical, we can exploit that fact that the 8-bit microprocessors need to calculate operations using 8-bit registers. We can hence simply guess 8 key bits, predict the Hamming weight of the corresponding 8 bits in θ_{plane}, and check whether or not our prediction correlates with the recorded power traces for multiple runs using differing I.

The results of the experiments, depicted in Figure 3, show how many power traces are needed until key bits can be reliably extracted. After having observed about 36 power traces the correlation of predictions using the correct key guess reliably scores the highest. The corresponding advantage of an attacker at this point is therefore 8 bits. Using the same set of power traces all remaining key bits can be recovered by simply adjusting the location of the key guess in the state.

We expect experiments on other platforms with larger architectures (such as 16-, 32-, 64-bit) to perform increasingly worse, mainly due to the fact that leakage becomes less informative with increasing register sizes. Additionally, especially on 32-bit or 64-bit platforms, checking all possible values of a 32 or 64-bit key guess is not really practical anymore. Here one can fall back to guessing, e.g.,



(a) Correlation of the correct key byte vs. highest correlation of all wrong key bytes.



(b) The expected attacker advantage in guessing key bits when targeting 8 key bits for 16 times in parallel. The grey line shows the gain of $i \cdot \lambda$, where λ is estimated by the biggest slope gotten from the DPA attack.

Fig. 3: DPA results for KECCAK-p[1600,12] on an 8-bit microcontroller. After having observed about 36 power traces the 128-bit key can be retrieved, 8-bit at a time. The experiment was repeated 100 times, the results are averaged.

only 16 key bits, thereby treating the remaining bits in a register simply as noise. This however comes also at the cost of a increased amount of required power traces.

5.3 Statistical Ineffective Fault Attacks

Thanks to our new model of accumulated leakage, we are also able to express the gain an attacker gets from a fault attack within a leakage resilient framework. In this section we will, for the sake of example, focus on statistical ineffective fault attacks (SIFA) [12, 13].

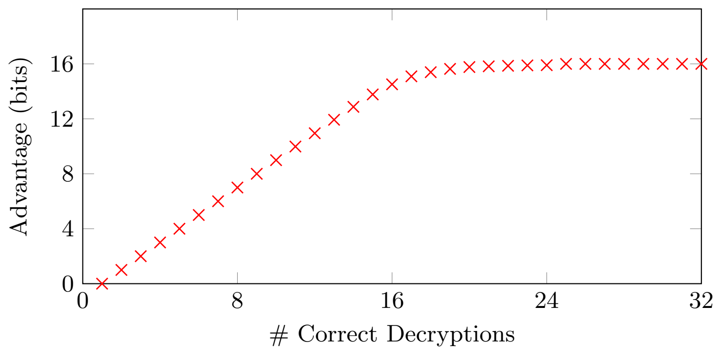
SIFA has been proven to be a very versatile attack vector that can be applied to permutation based schemes in a very straightforward way [16]. Although our scheme in Section 4 is not an authenticated encryption scheme, it can still be viewed as the initialization phase of a sponge-based stream cipher, similar as in Keyak or Ketje.

In our fault attack evaluation, we again consider the Keccak-p[1600,12] permutation as used in Figure 2 where the input I is randomly chosen and known by the attacker. We again target an AVR assembler optimized version of Keccak-p from the Extended Keccak Code Package (XKCP) [5], running on an 8-bit XMEGA 128D4 microprocessor. This time the ChipWhisperer-Lite evaluation board [34] is used to generate a clock signal for the victim that can additionally contain purpose-built glitches for causing erroneous computations.

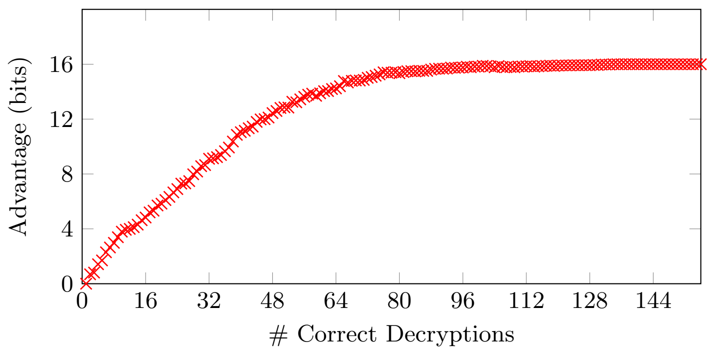
Following a similar strategy as in [16], our goal is to collect certain I's for which an induced fault during the computation of χ in round 2 leads to a correct computation of O. The location of the fault is hereby chosen such that an affected state bit roughly depends on 25 bits of the initial state. Since not all bits influence the target bit in a non-linear way, we can get a guessing advantage of at most 16 bits (see [16] for details). The main observation of SIFA (and Safe-Error Attacks in general) is that the condition whether or not a fault is ineffective can depend on the actual values that are used in the faulted computation. An ineffective fault induction can hence be used to filter out a specific set of I's that show a biased distribution of certain state bits in the faulted location. Once such a set of I's is collected by the attacker the corresponding key bits can be enumerated and the distribution of bits in χ in round 2 calculated. A key guess corresponding to a strong bias in some state bits, when measured, e.g., using the Squared Euclidean Imbalance (SEI), then indicates a correct key guess and vice versa. The more I's are collected the easier it is to distinguish the correct key guess from all wrong key guesses.

A quick visual inspection of a power trace reveals that the computation of χ in round 2 starts around clock cycle 9 550 when measured from the start of the permutation. During the experiment we hence send random inputs I to the device and inject a glitch into the clock signal once the victim starts to compute χ in round 2. Each I is used twice, once with clock glitch, once without, so we can check (by comparing O) whether or not the clock glitch affected the victim's computation. Note that in a more realistic attacker scenario the repetition of each I is not really needed since either a redundancy-based fault countermeasure or the authenticity check during authenticated decryption could indicate whether or not an induced fault was ineffective.

For the key recovery we guess 25 key bits and calculate the distribution of one affected state bit at the input of χ in round 2. If the exact location of the affected state bits is not known to the attacker the previous step can simply be repeated for all possible bit locations in the state. Again, a high SEI indicates a correct key guess while a low SEI indicates either an incorrect key guess or (if all key guesses result in low SEI) a wrong prediction of the location of the affected state bits. In our experimental setup a clock glitch always affects 8 bits of the state at once, probably because an instruction was skipped in χ in round 2. As a result we only find about one I per 256 faulted permutations where the induced fault was ineffective. The total number of required faulted permutations,



(a) No hiding countermeasure.



(b) With hiding countermeasure: 50% of the executions use a dummy key.

Fig. 4: Result of the SIFA evaluation on an 8-bit XMEGA 128D4 microprocessor using clock glitches. The advantage of exploiting the bias in one bit is plotted.

is thus comparably high, but, since every biased bit has a different dependency to the key bits, we can recover significantly more key bits at once. The results of our evaluation are depicted in Figure 4. They show how many ineffective faults were needed until the bias in one affected state bit was high enough so that the correct key guess can be distinguished with high confidence. We looked at two scenarios, one with the plain implementation of the permutation and one with a simple additional hiding countermeasure where the permutation is using a random dummy key in 50% of the cases. Depending on the scenario either about 20 I's (5 120 faulted encryptions) or about 96 I's (24576 faulted encryptions) are needed until a correct key guess can be distinguished with high confidence. Do note however that, in contrast to the DPA attack, reducing the amount of I's at the cost of a reduced attacker advantage is more plausible here. Since each state bit at the input of χ in round 2 has a partially different dependency on the key bits, we can run the key recovery 8 times, thereby recovering significantly more than 16 key bits using one set of I's.

5.4 Interpretation of Results

Our evaluation of the DPA attack shows that having about 36 traces for different inputs allows us to recover 8-bit of the secret state. Since nothing restricts us to also recover the other state bytes using the same data, we can expect that it is also possible to recover the whole 128-bit key. Not surprisingly, the evaluation of the DPA attack shows furthermore that the advantage of an attacker that is limited to the observation of just two inputs to the KECCAK-p permutation is rather negligible. Here, we remark that in asakey the data complexity is, indeed, limited to 2. Also, we want to stress that these results are for an implementation without countermeasures, using an evaluation setup that extremely favors the attacker, and assuming that the attacker has control over 1472 bits of I (instead of just one bit as in asakey).

If we have a look at fault attacks, and more specifically SIFA, the gain of an attacker could be even bigger, assuming the unlikely but ideal case that all I lead to ineffective faults. In the case of an attack on a completely unprotected implementation, the guessing advantage grows almost linearly with the data complexity (Figure 4a). In the presence of a simple hiding countermeasure like dummy rounds the required data complexity for achieving the same guessing advantage is however already impacted quite significantly (Figure 4b). Even in the ideal case the advantage of an attacker is only 1 bit per targeted bit. Since we targeted one complete byte in our experiments, we assume that the combined advantage is 8 bit on average. Again, we stress that these results assume an attacker that has control over 1472 bits of I (instead of just one bit as in asakey).

If we map our optimistic evaluation results to the accumulated leakage analysis of Section 4, we get $\max\big(\mathrm{AG}_{\mathtt{dpa}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)\big) = 0.34$ (from Figure 3b, advantage for 2 traces ($\max(r_i) = 2$)), considering DPA as attack vector. If we consider SIFA, we get one bit of advantage (from Figure 4a, advantage for 2 correct ciphertexts ($\max(r_i) = 2$)) per biased bit. Since in our SIFA attack a whole byte is biased, we simply scale to $\max(\mathrm{AG}_{\mathtt{sifa}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)) = 8$. Since all experiments have been performed for a secret portion of the state of 128 bits, we extrapolate these results for secrets bigger/smaller than 128 bits for the sake of simplicity. For instance, for a k-bit state, we assume an attacker has $\max\big(\mathrm{AG}_{\mathtt{dpa,sifa}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)\big) = 9\lceil \frac{k}{128} \rceil$. Since in a typical instance of asakey the parameters satisfy $b > c \geq k$, we have $9\lceil \frac{b}{128} \rceil > 9\lceil \frac{c}{128} \rceil \geq 9\lceil \frac{k}{128} \rceil$. To get more precise numbers, the experiments need to be redone with the secret states in use. We get

$$\begin{aligned}\mathbf{Adv}^{\text{i-lr}}_{\mathsf{asakey}}(\mathsf{A}) \leq \frac{p}{2^{k-9\lceil \frac{k}{128} \rceil}} + \frac{\nu_{r,c}^{Q-q} p + p}{2^{c-9\lceil \frac{c}{128} \rceil}} + \frac{pQ + 2pqk + p}{2^{b-9\lceil \frac{b}{128} \rceil}} \\ + \frac{(Q+2qk)q + 2\nu_{r,c}^{Q-q}}{2^c} + \frac{\binom{Q+2qk+1}{2} + \binom{Q+qk+1+p}{2} + \binom{p}{2}}{2^b},\end{aligned}$$

for an adversary A with construction complexity q, total number of encrypted message blocks Q, and with primitive complexity p, only considering DPA and SIFA.

5.5 Discussion on Attacks Limited by $\max(\boldsymbol{q}_i)$

As already mentioned, asakey as a mode strictly limits $\max(r_i)$ for encryption and decryption, but can limit $\max(\boldsymbol{q}_i)$ only during encryption. Hence, the mode itself only provides full protection and a security margin in case of attacks that scale with increasing $\max(r_i)$, such as DPA. For attacks that scale with increasing $\max(\boldsymbol{q}_i)$, the decryption implementation itself has to provide protection, so that, e.g., $\mathrm{AG}_{\mathsf{spa}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i)$ for an SPA is small, although $\max(\boldsymbol{q}_i)$ is essentially unbounded. In the case of passive side channel attacks, like SPA or template attacks, this requirement is comparable to a non-adaptive leakage assumption [19, 21, 47]. This means that for non-adaptive leakage, it is usually assumed that an implementation provides enough protection in case of repeated measurements of a primitive that processes the same input, so that an attacker cannot learn more from repeated measurements than from the first one. Also note that, form a practical perspective, achieving protection against SPA attacks is still quite a bit cheaper to realize than protection against DPA attacks, that are prevented by asakey and would otherwise usually require the usage of expensive higher-order masking.

Since we introduce with accumulated leakage a concept that also covers fault attacks, we can observe a similar behavior for fault attacks. Roughly speaking, in the case of fault attacks, we have also attacks that mainly scale for increasing $\max(r_i)$, like SFA and SIFA, but also attacks that mainly scale with increasing $\max(\boldsymbol{q}_i)$ like DFA. Here, in contrast to passive side channel attacks, a notion of non-adaptive fault attacks would make limited sense, since this basically assumes an attacker that always makes the same fault at the same position within an algorithm. Hence, in general, if $\max(\boldsymbol{q}_i)$ is not limited by a mode, we have to shift protection against attacks that scale well with $\max(\boldsymbol{q}_i)$ to the implementation. To see how attacks that scale well with $\max(\boldsymbol{q}_i)$ impact unprotected implementations of asakey, we give a discussion of DFA next.

To estimate the performance of Differential Fault Attacks (DFA) on unprotected implementations of our studied Keccak-p[12,1600] permutation we mainly refer to existing works by Bagheri et al. [1] and Luo et al. [27] who thoroughly analyzed the applicability of DFA attacks against SHA-3, and discuss differences to our attack scenario. Since DFA requires the observation of faulty computations, it can only be used to directly infer information about the later rounds of the attacked permutation. Hence, when performing DFA, we first aim to recover S_{K_O} which could then be used to also recover S_{K_I}. Following the methods of Bagheri et al. [1] who studied the effects of single-bit faults within Keccak-p, a single pair of correct and faulted computation can directly leak 22 bits of the state before the S-box layer of the penultimate round. By repeating this procedure about 80 times using different fault locations the entire state before the S-box layer of the penultimate round can be recovered. Luo et al. followed up on this work by analyzing fault injections with byte granularity (instead of bit granularity) and concluded that: (i) about 120 repetitions are necessary if timing but not location of the fault can be precisely controlled by the attacker, and (ii) as few as 17 repetitions are required if both timing and location of the fault can be precisely controlled by the attacker [27].

The main difference between the setting of, e.g., SHA3-512 by Bagheri et al. [1] and our scenario is the amount of bits of O that are directly observable by the attacker. While 576 bits can be observed by the attacker in SHA3-512, up to 1472 bits can be observed in our scenario. This does not immediately lead to a reduced amount of necessary faulted executions since the state recovery via DFA happens in the penultimate round. Some improvements can however be expected since the available information about large parts of the final state can still be used to infer some prior knowledge about state bits in the penultimate round, which then reduces the required number of faulted executions.

Please note that for asakey this attack approach cannot be directly used to extract the long term key, since the encryption part of asakey works with a session key from a re-keying function that is hard to invert. To extract the long term key, a multi-step approach is necessary. Using the notation from Figure 1, it is necessary to:

1. Use DFA to recover a correct S_{k+1};
2. While injecting a fault in the penultimate round of the last permutation call of the re-keying function, use DFA to recover a faulty S_{k+1};
3. Repeat step 2 until sufficiently many faulty S_{k+1}'s have been recovered. Then, by also using the correct S_{k+1}, perform a DFA-style state recovery within the re-keying function.

Therefore, extending DFA from Keccak-p to asakey requires roughly a squared amount of faulted executions and has the additional requirement of performing precise combinations of two faults per execution.

To sum up, to recover the 128-bit secret state using a DFA, significantly more than one computation of the permutation using the same input is required to get in our case $\mathrm{AG}_{\mathtt{dfa}}(\boldsymbol{X}_i, \boldsymbol{q}_i, r_i) = 128$. Hence, a mode that puts a small limit on $\max(\boldsymbol{q}_i)$ would provide protection against this attack vector.

6 Conclusion

In this paper, we complemented existing approaches in leakage resilient cryptography and gave another method that aims to bring theory and practice closer together. Similar to the cryptanalysis of symmetric cryptographic primitives, our approach can also be purely driven by attacks. This allows for some benefits, most importantly that our approach is not only restricted to side channel attacks but is also able to include other implementation attacks, like fault attacks. However, our approach also has some limitations, some of them inherited from leakage resilience in general.

As our approach only focuses on the leakage resilience of the scheme, it does not cover all sources of information leakage that can occur within a system. A typical pitfall that is excluded is the loading of the secret key, that eventually happens in software implementations. Furthermore, our coverage of potential side channel attacks and fault attacks in Section 5 is far from exhaustive and should just give an impression of the underlying idea of our approach on accumulated leakage. Dependent on the capabilities of an attacker, template attacks are an attack vector that has been proven to threat leakage resilient implementations [31,44,45]. Nevertheless, we think that the ideas presented in this paper provide a nice complement to existing models in leakage resilient cryptography.

The mode asakey presented in this paper only focuses on putting a limit on the data complexity per permutation call ($\max(r_i) = 2$), but leaves the permutation calls with same inputs unrestricted ($\max(\boldsymbol{q}_i) \leq q$). An interesting research topic are modes that also put a strict limit on $\max(\boldsymbol{q}_i)$. For instance, asakey could be converted to such a mode, if we consider a synchronized sender and receiver that always increase the nonce N by one. If sender and receiver then always store S_k (see Figure 1) and utilize the inverse of p, the permutations within the scheme have to be only called once in forward and inverse direction processing the same inputs. Essentially, the “re-keying” function of asakey would be traversed in a tree-like fashion as in [24].

Acknowledgements. Christoph Dobraunig is supported by the Austrian Science Fund (FWF): J 4277-N38. Robert Primas is supported by the Austrian Research Promotion Agency (FFG) via the K-project DeSSnet, which is funded in the context of COMET – Competence Centers for Excellent Technologies by BMVIT, BMWFW, Styria and Carinthia, and via the project ESPRESSO, which is funded by the province of Styria and the Business Promotion Agencies of Styria and Carinthia.

References

1. Bagheri, N., Ghaedi, N., Sanadhya, S.K.: Differential Fault Analysis of SHA-3. In: Progress in Cryptology - INDOCRYPT 2015 - 16th International Conference on Cryptology in India, Bangalore, India, December 6-9, 2015, Proceedings. pp. 253–269 (2015), https://doi.org/10.1007/978-3-319-26617-6_14
2. Barwell, G., Martin, D.P., Oswald, E., Stam, M.: Authenticated Encryption in the Face of Protocol and Side Channel Leakage. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I. LNCS, vol. 10624, pp. 693–723. Springer (2017), https://doi.org/10.1007/978-3-319-70694-8_24
3. Berti, F., Pereira, O., Peters, T., Standaert, F.X.: On Leakage-Resilient Authenticated Encryption with Decryption Leakages. IACR Trans. Symmetric Cryptol. 2017(3), 271–293 (2017), https://doi.org/10.13154/tosc.v2017.i3.271-293
4. Berti, F., Pereira, O., Standaert, F.X.: Reducing the Cost of Authenticity with Leakages: a CIML2-Secure AE Scheme with One Call to a Strongly Protected Tweakable Block Cipher. In: Buchmann, J., Nitaj, A., eddine Rachidi, T. (eds.) Progress in Cryptology - AFRICACRYPT 2019 - 11th International Conference on Cryptology in Africa, Rabat, Morocco, July 9-11, 2019, Proceedings. LNCS, vol. 11627, pp. 229–249. Springer (2019), https://doi.org/10.1007/978-3-030-23696-0_12

5. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: eXtended Keccak Code Package. https://github.com/XKCP/XKCP, [Online; accessed 11-July-2019]
6. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: Keyak. Submission to the CAESAR competition: http://competitions.cr.yp.to (2014)
7. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R., Viguier, B.: KangarooTwelve: Fast Hashing Based on Keccak-p. In: Preneel, B., Vercauteren, F. (eds.) Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings. LNCS, vol. 10892, pp. 400–418. Springer (2018), https://doi.org/10.1007/978-3-319-93387-0_21
8. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings. LNCS, vol. 537, pp. 2–21. Springer (1990), https://doi.org/10.1007/3-540-38424-3_1
9. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In: Kaliski Jr., B.S. (ed.) Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings. LNCS, vol. 1294, pp. 513–525. Springer (1997), https://doi.org/10.1007/BFb0052259
10. Bronchain, O., Hendrickx, J.M., Massart, C., Olshevsky, A., Standaert, F.X.: Leakage Certification Revisited: Bounding Model Errors in Side-Channel Security Evaluations. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I. LNCS, vol. 11692, pp. 713–737. Springer (2019), https://doi.org/10.1007/978-3-030-26948-7_25
11. Daemen, J., Mennink, B., Van Assche, G.: Full-State Keyed Duplex with Built-In Multi-user Support. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II. LNCS, vol. 10625, pp. 606–637. Springer (2017), https://doi.org/10.1007/978-3-319-70697-9_21
12. Dobraunig, C., Eichlseder, M., Groß, H., Mangard, S., Mendel, F., Primas, R.: Statistical Ineffective Fault Attacks on Masked AES with Fault Countermeasures. In: Peyrin, T., Galbraith, S.D. (eds.) Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II. LNCS, vol. 11273, pp. 315–342. Springer (2018), https://doi.org/10.1007/978-3-030-03329-3_11
13. Dobraunig, C., Eichlseder, M., Korak, T., Mangard, S., Mendel, F., Primas, R.: SIFA: Exploiting Ineffective Fault Inductions on Symmetric Cryptography. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2018(3), 547–572 (2018), https://doi.org/10.13154/tches.v2018.i3.547-572
14. Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Mennink, B., Primas, R., Unterluggauer, T.: ISAP v2. Submission to NIST Lightweight Cryptography (2019)
15. Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Unterluggauer, T.: ISAP - Towards Side-Channel Secure Authenticated Encryption. IACR Trans. Symmetric Cryptol. 2017(1), 80–105 (2017), https://doi.org/10.13154/tosc.v2017.i1.80-105

16. Dobraunig, C., Mangard, S., Mendel, F., Primas, R.: Fault Attacks on Nonce-Based Authenticated Encryption: Application to Keyak and Ketje. In: Cid, C., Jacobson Jr., M.J. (eds.) Selected Areas in Cryptography - SAC 2018 - 25th International Conference, Calgary, AB, Canada, August 15-17, 2018, Revised Selected Papers. LNCS, vol. 11349, pp. 257–277. Springer (2018), https://doi.org/10.1007/978-3-030-10970-7_12
17. Dobraunig, C., Mennink, B.: Leakage Resilience of the Duplex Construction. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III. LNCS, vol. 11923, pp. 225–255. Springer (2019), https://doi.org/10.1007/978-3-030-34618-8_8
18. Dodis, Y., Kalai, Y.T., Lovett, S.: On cryptography with auxiliary input. In: Mitzenmacher, M. (ed.) ACM Symposium on Theory of Computing, STOC 2009. pp. 621–630. ACM (2009), https://doi.org/10.1145/1536414.1536498
19. Dodis, Y., Pietrzak, K.: Leakage-Resilient Pseudorandom Functions and Side-Channel Attacks on Feistel Networks. In: Rabin, T. (ed.) Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings. LNCS, vol. 6223, pp. 21–40. Springer (2010), https://doi.org/10.1007/978-3-642-14623-7_2
20. Dziembowski, S., Pietrzak, K.: Leakage-Resilient Cryptography. In: 49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA. pp. 293–302. IEEE Computer Society (2008), https://doi.org/10.1109/FOCS.2008.56
21. Faust, S., Pietrzak, K., Schipper, J.: Practical Leakage-Resilient Symmetric Cryptography. In: Prouff and Schaumont [37], pp. 213–232, https://doi.org/10.1007/978-3-642-33027-8_13
22. Fuhr, T., Jaulmes, É., Lomné, V., Thillard, A.: Fault attacks on AES with faulty ciphertexts only. In: Fischer, W., Schmidt, J. (eds.) FDTC 2013. pp. 108–118. IEEE Computer Society (2013), https://doi.org/10.1109/FDTC.2013.18
23. Fuller, B., Hamlin, A.: Unifying Leakage Classes: Simulatable Leakage and Pseudoentropy. In: Lehmann, A., Wolf, S. (eds.) Information Theoretic Security - ICITS 2015. LNCS, vol. 9063, pp. 69–86. Springer (2015), https://doi.org/10.1007/978-3-319-17470-9_5
24. Kocher, P.C.: Leak-resistant cryptographic indexed key update (Mar 25 2003), http://www.google.com/patents/US6539092, US Patent 6,539,092
25. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M.J. (ed.) Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. LNCS, vol. 1666, pp. 388–397. Springer (1999), https://doi.org/10.1007/3-540-48405-1_25
26. Longo, J., Martin, D.P., Oswald, E., Page, D., Stam, M., Tunstall, M.: Simulatable Leakage: Analysis, Pitfalls, and New Constructions. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I. LNCS, vol. 8873, pp. 223–242. Springer (2014), https://doi.org/10.1007/978-3-662-45611-8_12
27. Luo, P., Fei, Y., Zhang, L., Ding, A.A.: Differential Fault Analysis of SHA-3 Under Relaxed Fault Models. J. Hardware and Systems Security 1(2), 156–172 (2017), https://doi.org/10.1007/s41635-017-0011-4

28. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks. Springer (2007)
29. Martin, D.P., Oswald, E., Stam, M., Wójcik, M.: A Leakage Resilient MAC. In: Groth, J. (ed.) Cryptography and Coding - 15th IMA International Conference, IMACC 2015, Oxford, UK, December 15-17, 2015. Proceedings. LNCS, vol. 9496, pp. 295–310. Springer (2015), https://doi.org/10.1007/978-3-319-27239-9_18
30. Medwed, M., Standaert, F.X., Joux, A.: Towards Super-Exponential Side-Channel Security with Efficient Leakage-Resilient PRFs. In: Prouff and Schaumont [37], pp. 193–212, https://doi.org/10.1007/978-3-642-33027-8_12
31. Medwed, M., Standaert, F.X., Nikov, V., Feldhofer, M.: Unknown-Input Attacks in the Parallel Setting: Improving the Security of the CHES 2012 Leakage-Resilient PRF. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I. LNCS, vol. 10031, pp. 602–623 (2016), https://doi.org/10.1007/978-3-662-53887-6_22
32. Mennink, B., Neves, S.: Optimal PRFs from Blockcipher Designs. IACR Trans. Symmetric Cryptol. 2017(3), 228–252 (2017), https://doi.org/10.13154/tosc.v2017.i3.228-252
33. National Institute of Standards and Technology: FIPS PUB 202: SHA-3 Standard: Permutation-based hash and extendable-output functions. Federal Information Processing Standards Publication 202, U.S. Department of Commerce (8 2015)
34. OFlynn, C.: ChipWhisperer-Lite (CW1173) Basic Board. http://store.newae.com/chipwhisperer-lite-cw1173-basic-board/, [Online; accessed 11-July-2019]
35. Pereira, O., Standaert, F.X., Vivek, S.: Leakage-Resilient Authentication and Encryption from Symmetric Cryptographic Primitives. In: Ray, I., Li, N., Kruegel, C. (eds.) Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015. pp. 96–108. ACM (2015), https://doi.org/10.1145/2810103.2813626
36. Pietrzak, K.: A Leakage-Resilient Mode of Operation. In: Joux, A. (ed.) Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings. LNCS, vol. 5479, pp. 462–482. Springer (2009), https://doi.org/10.1007/978-3-642-01001-9_27
37. Prouff, E., Schaumont, P. (eds.): Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings, LNCS, vol. 7428. Springer (2012), https://doi.org/10.1007/978-3-642-33027-8
38. Selçuk, A.A.: On Probability of Success in Linear and Differential Cryptanalysis. J. Cryptology 21(1), 131–147 (2008), https://doi.org/10.1007/s00145-007-9013-7
39. Standaert, F.X., Pereira, O., Yu, Y.: Leakage-Resilient Symmetric Cryptography under Empirically Verifiable Assumptions. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I. LNCS, vol. 8042, pp. 335–352. Springer (2013), https://doi.org/10.1007/978-3-642-40041-4_19
40. Standaert, F.X., Pereira, O., Yu, Y., Quisquater, J.J., Yung, M., Oswald, E.: Leakage Resilient Cryptography in Practice. Cryptology ePrint Archive, Report 2009/341 (2009), https://eprint.iacr.org/2009/341
41. Standaert, F.X., Pereira, O., Yu, Y., Quisquater, J.J., Yung, M., Oswald, E.: Leakage Resilient Cryptography in Practice. In: Sadeghi, A.R., Naccache, D. (eds.)

Towards Hardware-Intrinsic Security - Foundations and Practice, pp. 99–134. Information Security and Cryptography, Springer (2010), https://doi.org/10.1007/978-3-642-14452-3_5
42. Taha, M.M.I., Schaumont, P.: Side-Channel Analysis of MAC-Keccak. In: 2013 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2013, Austin, TX, USA, June 2-3, 2013. pp. 125–130 (2013), https://doi.org/10.1109/HST.2013.6581577
43. Taha, M.M.I., Schaumont, P.: Side-channel countermeasure for SHA-3 at almost-zero area overhead. In: 2014 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2014, Arlington, VA, USA, May 6-7, 2014. pp. 93–96. IEEE Computer Society (2014), https://doi.org/10.1109/HST.2014.6855576
44. Unterstein, F., Heyszl, J., De Santis, F., Specht, R.: Dissecting Leakage Resilient PRFs with Multivariate Localized EM Attacks - A Practical Security Evaluation on FPGA. In: Guilley, S. (ed.) Constructive Side-Channel Analysis and Secure Design - 8th International Workshop, COSADE 2017, Paris, France, April 13-14, 2017, Revised Selected Papers. LNCS, vol. 10348, pp. 34–49. Springer (2017), https://doi.org/10.1007/978-3-319-64647-3_3
45. Unterstein, F., Heyszl, J., De Santis, F., Specht, R., Sigl, G.: High-Resolution EM Attacks Against Leakage-Resilient PRFs Explained - And an Improved Construction. In: Smart, N.P. (ed.) Topics in Cryptology - CT-RSA 2018 - The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings. LNCS, vol. 10808, pp. 413–434. Springer (2018), https://doi.org/10.1007/978-3-319-76953-0_22
46. Yu, Y., Standaert, F.X.: Practical Leakage-Resilient Pseudorandom Objects with Minimum Public Randomness. In: Dawson, E. (ed.) Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, San Francisco,CA, USA, February 25-March 1, 2013. Proceedings. LNCS, vol. 7779, pp. 223–238. Springer (2013), https://doi.org/10.1007/978-3-642-36095-4_15
47. Yu, Y., Standaert, F.X., Pereira, O., Yung, M.: Practical leakage-resilient pseudorandom generators. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010. pp. 141–151. ACM (2010), https://doi.org/10.1145/1866307.1866324