Christoph Erwin Dobraunig

On the Security and Design of Authenticated Encryption

DOCTORAL THESIS

to achieve the university degree of

Doktor der technischen Wissenschaften

submitted to

Graz University of Technology

Supervisor

Florian Mendel

Assessors

Prof. Christian Rechberger
Prof. Vincent Rijmen

Institute for Applied Information Processing and Communications (IAIK)

Graz, November 2017

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present doctoral thesis.

Date

Signature

Preface

First of all, live is meaningless without having friends and family. Hence, I want to thank my friends, my family, and my girlfriend Anna for their mere existence and the fun, love, patience, and sorrow provided by them. So far, I can consider myself to be one of the luckiest persons on earth, who has never had any health issues and in general has experienced life in easy mode.

While my interest in cryptography already started back in my school days (mainly because the concept of asymmetric cryptography felt too unrealistic), I have to thank Mario Lamberger for giving excellent courses on IT-security, which have woken my interest in this area again. Although the concept of cryptanalysis — to make contributions to the progress of science by pure thinking, just needing pen&paper — seemed very appealing to me, my fear of failure was overwhelming. Luckily, I met Martin Schläffer, who took that fear away and guided me through my first steps, until he had to leave the department during the first months of my PhD.

My PhD started with Maria Eichlseder, Florian Mendel, and Martin Schläffer inviting me to join the design team of the CAESAR candidate Ascon, which was an incredible honour for a rookie like me. Their spirit of working without the slightest sign of enviousness, putting the result in the first place has served as a role model for me and I have always tried to follow this spirit.

Most of the time during my PhD, I had the opportunity to work with two geniuses, Maria and Florian. I have to thank Maria for being a wonderful colleague, patiently bearing my humor and always open to discuss ideas. Moreover, I am very thankful for the guidance provided by Florian. From the start of my PhD on, he has become the mentor I needed. He never hesitated to share his ideas, or to discuss mine. Without his continuous support and motivation I would not have been able to finish this thesis. Moreover, I want to thank Vincent Rijmen for agreeing to be my external assessor.

Furthermore, I want to thank Stefan Mangard and Christian Rechberger for employing me, leaving me freedom to do research in areas of my interest and always having an open door to discuss issues, or to provide help with their expertise. Moreover, I want to thank all my colleagues and the small board for providing such a great working atmosphere. At last, I have to thank all my co-authors: Andrey Bogdanov, Christoph Ehrenhöfer, Maria Eichlseder,

Hannes Groß, Markus Hofinger, Daniel Kales, François Koeune, Thomas Korak, Martin M. Lauridsen, Eik List, Victor Lomné, Stefan Mangard, Florian Mendel, Alexander Oprisnik, Thomas Plos, Martin Schläffer, François-Xavier Standaert, Elmar Tischhauser, Thomas Unterluggauer, Erich Wenger, Christoph Wiesmeier, and Johannes Wiesmeier. Without them, none of the work would have been possible.

Abstract

In contrast to ordinary encryption schemes that only ensure the confidentiality of data, the purpose of authenticated encryption is to ensure both, confidentiality and authenticity. In this work, we look at authenticated encryption from the two different perspectives of algorithmic security and implementation security (e.g., side-channel and fault attacks). This work focuses on symmetric authenticated encryption, meaning that all involved parties share the same key material.

The contributions regarding cryptanalysis are tightly linked with CAESAR, an international competition aiming to identify good authenticated encryption schemes which are secure for the next decades. In the first round of CAESAR, 57 candidates entered this competition. Since the selection process of CAESAR is largely based on publicly available inputs, one task of the cryptographic community to aid the progress of CAESAR is to identify insecure candidates and get insight in the security of the others. To contribute in this direction, we have developed an automatic search tool for linear characteristics that is applicable to a wide range of candidates. Furthermore, we provide dedicated analyses for the CAESAR candidates Ascon, ICEPOLE, and KIASU.

While it is the minimum requirement to withstand cryptanalytic attacks, the trend to connect more and more devices to the Internet generates the need for implementations of authenticated encryption schemes that withstand side-channel and fault attacks. The need for low-cost countermeasures against side-channel attacks has led us in the direction of re-keying. We present new fresh re-keying schemes, allowing the protection of a simple blockcipher call, and evaluate the security of re-keying schemes like Keymill. Moreover, we have designed Isap, a sponge-based authenticated encryption scheme, where we put the focus on side-channel protection during the whole design process.

With respect to fault attacks, the nonce defined in most schemes seems — at the first glimpse — to be a built-in countermeasure against fault attacks, since it usually prevents an attacker from collecting pairs of faulty and correct ciphertexts. This is a condition needed to execute a wide range of fault attacks like differential fault analysis. To counter this impression, we provide practical fault attacks on AES-based authenticated encryption schemes, which are applicable to several CAESAR candidates.

Table of Contents

Part I

Background

1 Introduction

This thesis is centered around authenticated encryption. More precisely, we focus on nonce-based symmetric authenticated encryption schemes with associated data [121]. In the context of this thesis, we consider an authenticated encryption algorithm as a function that takes a quadruple of secret key K, nonce N, associated data A and plaintext P as input and generates a ciphertext C and a tag T as output:

$$\mathcal{E}(K, N, A, P) = (C, T)$$

We assume that the secret key K is pre-shared between communication parties, whereas the nonce N, associated data A, ciphertext C and the tag T can be exchanged between communicating parties via insecure/public channels. This means that they can typically be observed and manipulated by third parties. The goal of authenticated encryption is to ensure the confidentiality of the plaintext P and additionally the authenticity of the plaintext P and associated data A under such conditions. In order to achieve this, nonce-based authenticated encryption schemes typically require a nonce N (number used only once) as additional input, while the tag T is additional information used to verify the authenticity.

Informally speaking, confidentiality in the context of authenticated encryption means that it should be infeasible for an entity not knowing the secret key K to extract any information about the plaintext P (except its length) from observed data, i.e., the nonce N, associated data A, ciphertext C, or tag T. The idea behind keeping the plaintext P and associated data A authentic is

to ensure that an intentional (by an entity not knowing the secret key), or unintentional modification of plaintext P and associated data A can be detected. As a consequence, the corresponding decryption algorithm processing (N, A, C, T) returns the plaintext P only if the tag verifies, otherwise invalid ($\perp$) is returned:

$$\mathcal{D}(K, N, A, C, T) \in \{P, \perp\}$$

Motivation. Authenticated encryption fulfills a crucial need, because for most practical applications there is not much value in just ensuring that a message is kept confidential. For instance, consider an ordinary bank transaction. Here, confidentiality on how much money to whom is transmitted is clearly desirable. However, the authenticity that 4 € should be transmitted and not 4 000 000 € can also be considered as crucial. Despite this need, research has focused for a long time on schemes just fulfilling one property separately, providing encryption schemes for confidentiality and message authentication codes (MAC) to ensure authenticity. Thus, in order to achieve authenticated encryption, ordinary encryption schemes and MACs have been combined in a so-called generic composition [13, 75].

While first dedicated authenticated encryption schemes were already presented by Jutla [64] around 2000 and many more followed, the question from a practitioner's viewpoint is to choose amongst them. For symmetric cryptography, cryptographic competitions have shown to be a valuable instrument in identifying good and useful cryptographic primitives, as demonstrated by the AES competition [106], the eSTREAM Project [46], or the SHA-3 competition [108].

In 2014, the competition for authenticated encryption: security, applicability, and robustness (CAESAR) [34] started. The goal of this competition is to identify a portfolio of authenticated encryption schemes that are suitable for widespread adoption and offer an advantage over AES-GCM [34]. A total of 57 authenticated encryption schemes entered the first round of the competition. Amongst them was Ascon:

- Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. "Ascon". Submission to the CAESAR competition: `http://competitions.cr.yp.to`. 2014

The task of the cryptographic community is to participate in the selection process. While this is typically done by providing cryptanalysis of the candidates, over the past years, the resistance of schemes and their implementations against so-called implementation attacks has become more and more important. In contrast to cryptanalysis, implementation attacks do not only consider the algorithmic description of a scheme alone, but consider it together with its implementation as a whole. Most implementation attacks require an attacker to have access to the device in order to observe side-channels like the power consumption [72], or to be able to induce faults to disturb the computation [33].

Contributions in this thesis. This thesis gives a broad view on authenticated encryption by considering it under the light of cryptanalysis, side-channel attacks, and fault attacks. The provided cryptanalytic results are tightly coupled with CAESAR. We present an automatic search tool for linear characteristics [c7] in order to facilitate the analysis of the candidates. To demonstrate its power, this tool is applied to the CAESAR candidates Ascon, ICEPOLE, Keyak, Minalpher, and Prøst. Our contribution to the cryptanalysis of CAESAR candidates is complemented with dedicated analyses for Ascon [c13], ICEPOLE [c10], and KIASU [c12].

In the context of implementation attacks, we present Isap [j2], a sponge-based authenticated encryption scheme, where we focus on side-channel protection during the design process. Amongst others, Isap contains concepts inherited from re-keying-based countermeasures, knowledge we have gathered during the design of fresh re-keying schemes [c14] and the analysis of Keymill [c3]. The last contribution contained in this thesis presents fault attacks on authenticated encryption schemes [c2]. In particular, it is demonstrated that such attacks are easy to execute in practice for a wide range of schemes, and the need for dedicated countermeasures against fault attacks for implementations that are used in hostile environments is pointed out.

Other contributions. Besides the contributions that can be found in this thesis, the author of this thesis has contributed to various other publications. This includes participation in the analysis of hash functions like Kupyna [c9] or SHA-512/224 and SHA-512/256 [c6], (tweakable) blockciphers like LowMC [c11], MANTIS [j1], or KIASU-BC [c15], permutations like Simpira [c5], and further authenticated encryption schemes like Calico [c1] or Prøst-OTR [c8]. Moreover, the author contributed to the development of a search tool for differential characteristics [c16], attacks on fresh re-keying schemes [c4], and made some small contributions in the area of hardware designs [c17, j3, c18]. A full list of over 20 publications can be found in Part II.

About this thesis. This thesis is a cumulative thesis consisting of two parts. The first part of this thesis serves as a high-level overview on the field of authenticated encryption, cryptanalysis, side-channel attacks, and fault attacks to put the publications of Part II in context. Hence, Part I does not contain any new scientific contributions or any new insights, and is only based on previously published work (where the author of this thesis has given his best to correctly give all references). For sake of clarity and to be easier to digest, the explanations given in Part I are highly condensed and simplified, giving probably too much of a black-and-white view about the topic.

Part II contains the scientific contributions of this thesis by appending the papers and detailing the author's contribution to each paper. Regarding the contribution, we want to point out the following statement of the American Mathematical

Society, which we think also applies to cryptography: *"In most areas of mathematics, joint research is a sharing of ideas and skills that cannot be attributed to the individuals separately. The roles of researchers are seldom differentiated (in the way they are in laboratory sciences, for example). Determining which person contributed which ideas is often meaningless because the ideas grow from complex discussions among all partners"* [3]. As a consequence, the lists of authors of all appended papers are in alphabetical order, which stands in contrast to other areas that always identify a most important person that is put in first place.

The outline of Part I is as follows. First, we show the basic principles of the design of authenticated encryption schemes and its building blocks in Chapter 2. After that, we discuss the purpose of cryptanalysis and introduce important techniques in Chapter 3. Chapter 4 gives a short introduction to side-channel attacks and potential countermeasures with a focus on re-keying. Then, we introduce the concept of fault attacks and discuss their application to authenticated encryption in Chapter 5.

2

Designing Authenticated Encryption Schemes

Keeping data confidential and authentic is one of the main goals of symmetric cryptography. One way to achieve both (confidentiality and authenticity) is to combine two schemes that just achieve one (either confidentiality or authenticity) in a so-called "generic composition", where an encryption scheme is combined with a message authentication code (MAC). Such constructions are widely used in practice, for example in TLS [42], SSH [145], or IPSec [67]. However, using two schemes for authenticated encryption implies usually two runs over the data, resulting in a potential implementation overhead. So it is natural to ask: can we do better?

A partial answer to this question might be given by CAESAR [34], a competition that aims to identify a portfolio of authenticated encryption schemes. In total, 57 authenticated encryption schemes have been submitted to the first round of CAESAR, too many to cover them all in this chapter. Therefore, we restrict the focus of this chapter on (tweakable) blockcipher-based and sponge-based designs, which are covered in Part II of this thesis.

As the name suggests, the underlying "building blocks" of (tweakable) blockcipher-based authenticated encryption schemes are (tweakable) blockciphers, whereas in the case of sponge-based designs, those are typically unkeyed permutations. The design of these "building blocks" is a delicate task that significantly influences the security and implementation characteristics of the resulting authenticated

encryption scheme. Although there are numerous ways for designing these blocks, several "higher level" design principles, or structures, have been demonstrated to be useful. We briefly introduce two of them in this chapter, substitution permutation networks (SPNs) and the Feistel structure.

The remainder of this chapter is structured as follows. First, we give a brief overview on the combination of encryption schemes and MACs to construct authenticated encryption schemes in Section 2.1. This is followed by a review of dedicated designs: (tweakable) blockcipher-based designs in Section 2.2 and sponge-based designs in Section 2.3. Finally, we have a brief look at design principles of building blocks, like blockciphers or unkeyed permutations, in Section 2.4.

2.1 Generic Composition

In the context of authenticated encryption, the term "generic composition" usually refers to constructions that combine an encryption scheme (ENC) with a message authentication code (MAC) in order to provide authenticated encryption. Naturally, there exist several ways how an encryption scheme can be combined with a MAC, which do not necessarily follow the interface of nonce-based authenticated encryption as introduced in Chapter 1. As a consequence, different compositions have been used in practice. For instance, TLS [42] can use a MAC-then-encrypt approach, SSH [145] an encrypt-and-MAC approach, and IPSec [67] an encrypt-then-MAC approach (if they do not use a dedicated authenticated encryption scheme), which has already been pointed out by Krawczyk [75]:

- **MAC-then-encrypt:** $T = \mathsf{MAC}_{K_a}(P)$, $C = \mathsf{ENC}_{K_b}(P\|T)$. C is exchanged.
- **Encrypt-and-MAC:** $T = \mathsf{MAC}_{K_a}(P)$, $C = \mathsf{ENC}_{K_b}(P)$. C and T are exchanged.
- **Encrypt-then-MAC:** $C = \mathsf{ENC}_{K_b}(P)$, $T = \mathsf{MAC}_{K_a}(C)$. C and T are exchanged.

These constructions have been analyzed independently by Krawczyk [75] and Bellare and Namprempre [13]. Both analyses seem to have a clear favorite, encrypt-then-MAC. For instance, Krawczyk states: "*We show that any secure channels protocol designed to work with any combination of secure encryption (against chosen plaintext attacks) and secure MAC must use the encrypt-then-authenticate method*" [75]. But does this mean that the other two methods should not be used at all?

Recently, Namprempre, Rogaway, and Shrimpton [104] give a more extensive analysis of the generic composition. They investigate how nonce-based authenticated encryption with associated data can be realized using various building

blocks like encryption schemes that require an initial value IV to be random, or encryption schemes that just require a nonce N to be unique. Some of the resulting schemes that have been shown to be secure by them can be considered to follow MAC-then-encrypt and encrypt-and-MAC principles. Furthermore, Namprempre, Rogaway, and Shrimpton [104] also demonstrate that encrypt-then-MAC constructions can fail if, for example, the nonce is not authenticated. Therefore, they point out that extreme caution has to be taken when interpreting results for the generic composition from the literature in an attempt to instantiate concrete schemes.

2.2 (Tweakable) Blockcipher-based Designs

As we have seen in the previous section, it is possible to design authenticated encryption schemes by combining an encryption scheme with a MAC. Besides the danger of insecure instantiations of generic compositions, the fact that two dedicated schemes are used requires the data to be processed twice (two-pass scheme). However, it is also possible to design dedicated authenticated encryption schemes which provide authenticity and confidentiality with just one pass over the data, potentially leading to schemes with a lower overhead for implementation and higher performance.

Among the first dedicated authenticated encryption schemes are the blockcipher-based schemes by Jutla [64]. Jutla observes that an encryption of a plaintext concatenated with a checksum of it using the CBC mode yields a dedicated authenticated encryption scheme providing confidentiality and authenticity if the output is masked with a random sequence. In a further step, Jutla also uses masks to remove the chaining between the processing of the single blocks, leading to IAPM shown in Figure 2.1.

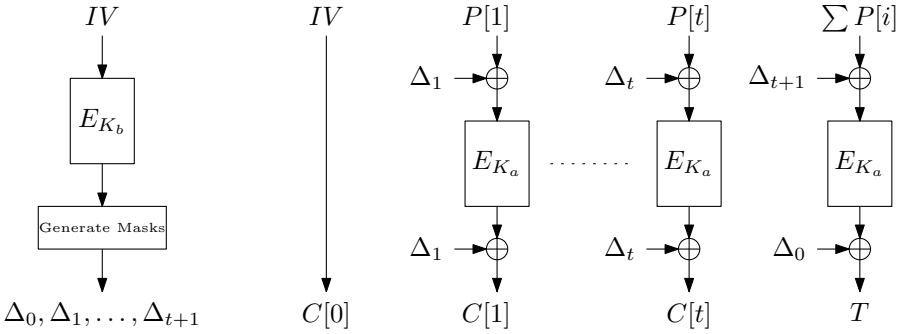


Figure 2.1: Integrity aware parallelizable mode (IAPM) as shown in [65].

There exist many variants of IAPM and the variant shown in Figure 2.1 is described by Jutla in [65]. This variant of IAPM requires two secret keys K_a and

K_b. The secret key K_b, together with a random IV is used for the generation of the secret masks Δ_i. These masks are needed for the encryption of the plaintext P and the generation of the tag T. The plaintext P is split into t n-bit blocks $P[i]$, which are encrypted separately using a blockcipher with the key K_a and secret masks that are xored to the input and output of the blockcipher. The tag T is generated by encrypting the xor sum of all plaintext blocks using the masks Δ_{t+1} and Δ_0.

Building upon IAPM, Rogaway, Bellare, Black, and Krovetz [122] propose the offset codebook mode (called OCB1 in the following [76]). OCB1 provides several improvements over IAPM. For instance, OCB1 allows the encryption of arbitrary-length plaintexts, which do not have to be a multiple of the block size, while ensuring that the resulting ciphertext length (not considering the tag) matches the plaintext length. Schemes like IAPM and OCB1 provide confidentiality and authenticity for data, however, as noted by Rogaway [121], those schemes (in their initial versions) do not address the problem of associated data. Associated data is data that shall be kept authentic, but where confidentiality is not needed, or is even undesired. For example, consider the concatenation of some header information (associated data A) with some message (plaintext P). Clearly, for transmission via public networks, the authenticity of A and P and the confidentiality of P are desirable properties, while the confidentiality of A might be counterproductive for the transmission. Thus, Rogaway introduces OCB2, which provides authenticated encryption with associated data (AEAD) [121].

In [82], Liskov, Rivest, and Wagner introduce the concept of tweakable blockciphers. In contrast to blockciphers, tweakable blockciphers provide an additional

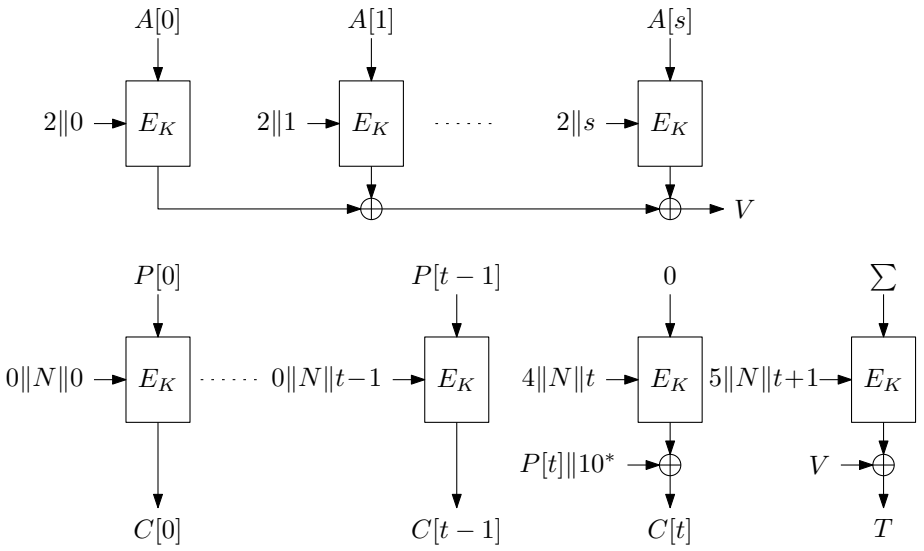


Figure 2.2: ΘCB3-based mode of Deoxys v1.41 (figure based on [62]).

public input called the tweak. As stated by Liskov, Rivest, and Wagner, "*each fixed setting of the tweak gives rise to a different, apparently independent, family of standard blockcipher encryption operators*" [82]. This property can be used to design quite elegant schemes, for instance, the authenticated encryption scheme TAE [82], which essentially follows the ideas of OCB1, but uses a tweakable blockcipher instead of a masked blockcipher.

Acknowledging the idea of tweakable blockciphers, Krovetz and Rogaway [76] introduce the tweakable blockcipher-based authenticated encryption scheme ΘCB3 that underlies OCB3. ΘCB3 serves as basis of modes used by the CAESAR candidates Kiasu [60] and Deoxys [62]. The ΘCB3-based mode of Deoxys is sketched in Figure 2.2.

2.3 Sponge-based Designs

Besides basing authenticated encryption schemes on (tweakable) blockciphers, another prominent way of designing authenticated encryption schemes is to utilize unkeyed permutations. A significant number of permutation-based authenticated encryption schemes (for instance Ascon, ICEPOLE, Keyak, or Ketje) follow the concept of the sponge construction [17, 19], or rather the concept of the duplex construction [19, 22]. In this section, we will briefly describe the sponge and duplex construction in Section 2.3.1 and Section 2.3.2, while discussing variants and modifications of them in Section 2.3.3.

2.3.1 Sponge Construction

Amongst others, the winner of the SHA-3 competition Keccak [16, 109] follows the sponge construction [17, 19], proposed by Bertoni, Daemen, Peeters, and Van Assche. Keccak and most other sponge based schemes build upon a fixed-width b-bit permutation p. As shown in Figure 2.3, a sponge splits its b-bit internal state into an outer part of r bits and an inner part of c bits, where only the outer part is modified by the input I, or used as output O. The size of the outer part is called the rate r and the size of the inner part is called the capacity c.

An input I that has to be processed by a sponge is padded and split into r-bit blocks. Those r-bit blocks are absorbed by xoring each block to the outer part of the state, followed by an application of the permutation p. To produce the output O, the sponge is then squeezed. Here, the permutation p is iteratively applied to the whole state, where r bits of the outer part can be used as output after each application of p. In this way, a sponge is able to provide outputs of arbitrary length.

Bertoni, Daemen, Peeters, and Van Assche [18] have been able to prove that the sponge construction is indifferentiable from a random oracle if it is used with

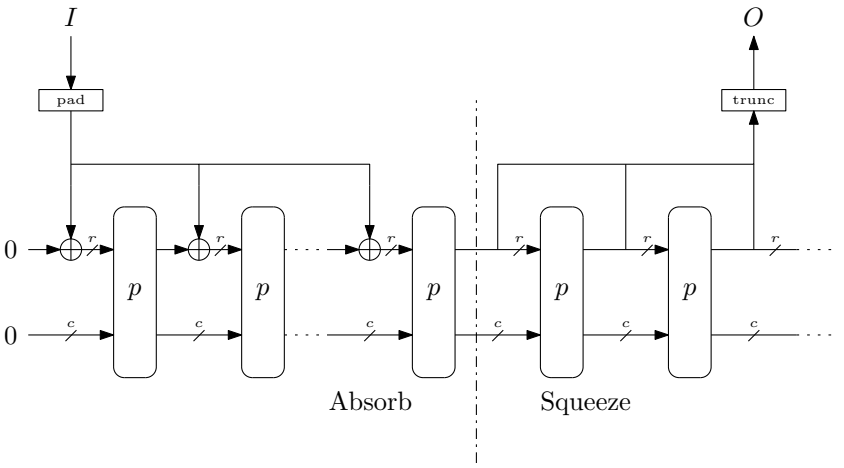


Figure 2.3: Sponge construction (figure based on [19]).

a random permutation. They show that the advantage of an attacker trying to differentiate such a sponge from a random oracle depends on the capacity c. Informally, a random oracle is a theoretical construction, which maps each new input of arbitrary length to a random infinite output string, providing always the same output for the same input. Having a construction that essentially behaves like a random oracle does not only allow hashing, but can also be used to instantiate message authentication codes (MAC), or streamciphers as observed in [17]. For instance, a sponge-based MAC and a sponge-based streamcipher are used in the authenticated encryption scheme ISAP [j2] in a generic composition. However, it is also possible to design authenticated encryption schemes that only need a single pass over the data by utilizing the duplex construction.

2.3.2 Duplex Construction

Bertoni, Daemen, Peeters, and Van Assche [22] introduce the duplex construction shown in Figure 2.4. Like sponge constructions, the duplex construction builds upon fixed-width permutations (or transformations). However, in contrast to the sponge construction, the duplex construction allows to absorb one r-bit input block $I[i]$, while producing one r-bit output block $O[i]$ separated by one permutation call p. An output block $O[i]$ depends on all previously absorbed input blocks $I[0], I[1], \ldots, I[i]$.

With the help of the duplex construction, single-pass authenticated encryption schemes can be realized. However, while a lot of authenticated encryption schemes have their roots in the duplex or sponge construction, designers usually take some liberty to tweak (and possibly improve upon) the original duplex or sponge construction. In the following, we will have a look at developments in this area.

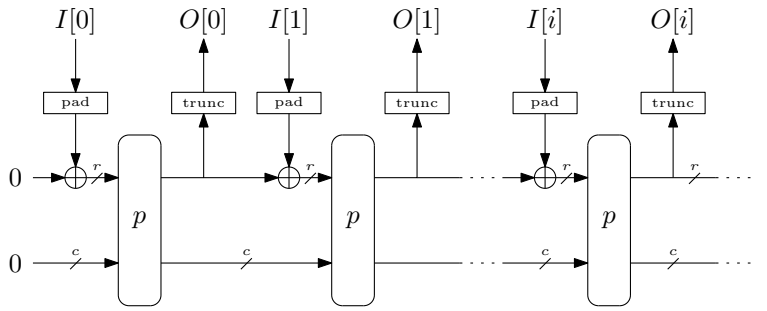


Figure 2.4: Duplex construction (figure based on [19]).

2.3.3 Variants

The sponge construction and its most prominent instance Keccak [16] allow a broad range of use cases, ranging from unkeyed constructions, like hash functions, to keyed constructions, such as MACs or streamciphers. However, it is an interesting question to ask which modifications of the original sponge and duplex construction can be made for constructions that limit the potential use cases and, for instance, always use a secret key and potentially a unique nonce. Various lines of work deal with this question and propose designs that, for example, reduce the capacity, allow the modification of the inner part of the state, and explore the use of permutations with a different cryptographic "strength" (typically a different number of rounds). One of the first proposals in this direction comes from Bertoni, Daemen, Peeters, and Van Assche [23], who propose the constructions DonkeySponge for, e.g., MACs and MonkeyDuplex for, e.g., authenticated encryption.

As already mentioned, several CAESAR candidates build upon the sponge construction, or more precisely, the MonkeyDuplex construction. For instance, they also use permutations with a different number of rounds for different phases of the scheme. Typically, those phases can be split into initialization, data processing, and finalization, as sketched in Figure 2.5.

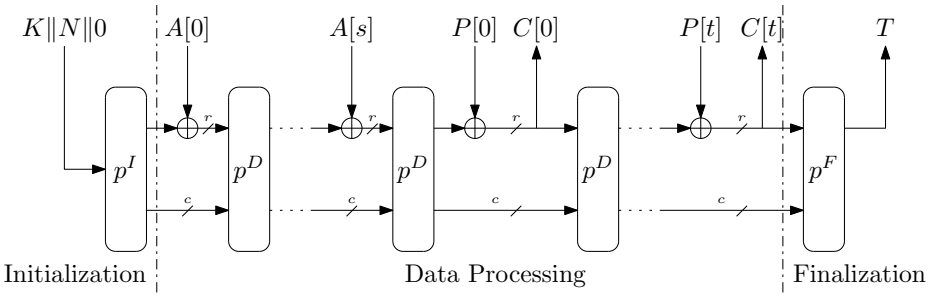


Figure 2.5: Sketch of MonkeyDuplex-based authenticated encryption scheme.

During the initialization, the key K is processed together with the nonce N. The permutation p^I used during this phase is a "strong" permutation, so that states after the initialization can be considered to be independent for different nonces. This fact usually limits the available attack paths and might allow a designer to reduce the number of rounds of the permutation p^D that is used during the processing of the associated data A, plaintext P, and generation of the ciphertext C. Typically, the number of rounds of the permutation p^F used for the finalization is increased compared to p^D. As a result of this approach, parameters like the rate r or the number of rounds for the permutation used in such designs have a tight interaction with the security of the scheme and thus, have to be based on a thorough cryptanalysis of the whole resulting authenticated encryption scheme.

Besides using permutations of different "strength", other tweaks introduced include the keying of initialization and finalization as done by, e.g., Ascon [i2], or parallel processing of data as done by, e.g., NORX [7]. By making such modifications compared to the scrutinized original sponge and duplex constructions, designers take the risk of introducing flaws and allowing, for instance, generic attacks on their sponge-based constructions that are independent of the choice of the concrete permutations. Thus, Jovanovic, Luykx, and Mennink [63] take a closer look at the security of several sponge-based authenticated encryption schemes (independent of the used permutations) focusing on NORX, but also argue that their given proof generalizes to several other CAESAR candidates like, e.g., Ascon, ICEPOLE, or Keyak. They observe that the use of a unique nonce and secret key can allow to reduce the size of the capacity compared to the results regarding unkeyed sponge-based constructions, also enhancing the insights already developed for keyed sponges in [21].

Further work in the area of sponge-based authenticated encryption deals with the ways of absorbing associated data. As already observed for DonkeySponge [23], which has been shown to be sound in [53], it is possible to construct a sponge-based MAC where the whole permutation width b can be used to absorb the data. Hence, for sponge-based authenticated encryption schemes, the question arises if the inner part of the state can be modified and used to absorb associated data. A positive answer is given in the work of Sasaki and Yasuda [127] or Mennink, Reyhanitabar, and Vizár [99].

2.4 Designing Building Blocks

So far, we have dealt with rather high-level descriptions of authenticated encryption schemes, not tackling the problem of designing their building blocks, which are in our case unkeyed permutations and (tweakable) blockciphers (which are permutations for a fixed key and tweak). So typically, the design of a permutation is similar to the design of a blockcipher without a key schedule as already mentioned by Bertoni, Daemen, Peeters, and Van Assche [22]. For sake of

simplicity, we just focus on the design of the (underlying) permutation and ignore the design of a key schedule. We will discuss two prominent design approaches — substitution permutation networks (SPNs) and Feistel structures — and describe their high-level working principle in this section.

Already an article dating back to 1973 by Feistel [48] describes the concept of substitution permutation networks (SPN), where its roots have been credited to earlier work of Shannon [129]. Figure 2.6 sketches the concept of an SPN, which consists of the iterative application of small non-linear substitution boxes (S-boxes), a bit permutation (or more general any linear function L), and a round key addition K (or constant addition in the case of unkeyed permutations). Following the principles of Shannon [129], the non-linear S-boxes are responsible for providing confusion, while the linear layer provides diffusion between them.

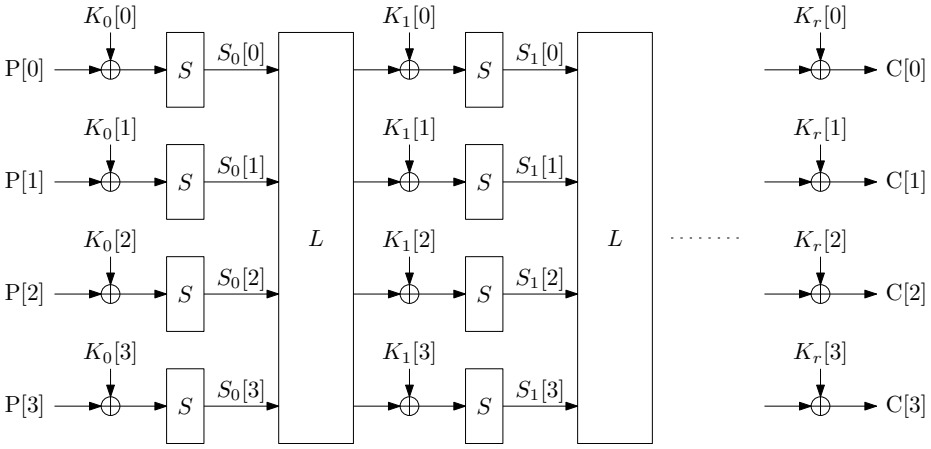


Figure 2.6: Sketch of an SPN-based blockcipher having r rounds.

In contrast to an SPN, where each round has to be a permutation and thus also each S-box, the so-called Feistel structure (Figure 2.7) as used for instance by DES [105] allows to design a permutation from (possibly keyed) functions F_i. However, usually, the used round functions follow similar design principles as SPNs, meaning that they also consist of layers of small S-boxes and linear functions. Nevertheless, an advantage of the Feistel structure is the fact that it is invertible without the need to implement the inverse of the single round functions F_i. Furthermore, by relying on generalized Feistel structures, permutations of different width can be realized using always similar round functions F_i as done, e.g., by Simpira [57].

Based on these concepts, numerous different ciphers and permutations have been designed and the choice of the used S-boxes, linear layers, and the resulting number of rounds r can be seen as an evolutionary process that is driven by new insights regarding security and possible applications, but also depends on the context of the use of the building block (e.g., in which high level construction

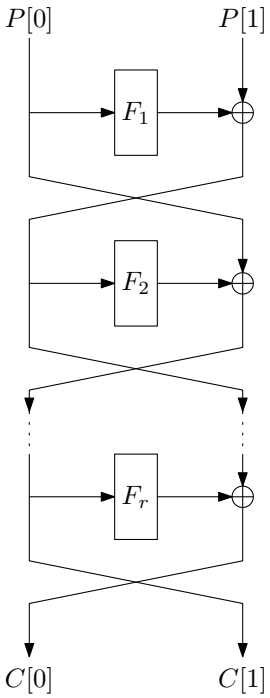


Figure 2.7: Sketch of a Feistel-based blockcipher having r rounds.

the building block is used). With respect to security, the introduction of timing attacks on table-based implementations of AES [14, 114] and power analysis attacks [72] are reasons to develop building blocks and designs which allow for constant-time implementations and are easy to mask like the permutations of the CAESAR candidates Ascon [i2] or Keyak [24]. Furthermore, new cryptanalytic insights like the discovery of differential [27] and linear cryptanalysis [88] usually lead to novel designs like Shark [120], Square [37], and Rijndael [38] that aim to resist these attacks by carefully choosing the S-boxes and incorporating maximum distance separable error correcting codes in their respective linear layers. In the next chapter, we will see various cryptanalytic techniques that we have used in Part II of this thesis to evaluate the security of various authenticated encryption schemes.

3 Cryptanalysis

This chapter deals with the analysis of cryptographic primitives, focusing on authenticated encryption schemes. In contrast to side-channel (Chapter 4) and fault attacks (Chapter 5), cryptanalysis assesses the security of a cryptographic primitive when it is treated like a black-box, as illustrated in Figure 3.1. In such a scenario, it is usually assumed that an attacker has knowledge about the cryptographic algorithm used (except the secret key) and is able to observe the data that is exchanged. In the case of authenticated encryption, this might be the nonce N, associated data A, ciphertext C, and tag T. Furthermore, an attacker might have knowledge about confidential inputs like the plaintext, or is even able to enforce the encryption of chosen inputs. However, an attacker is assumed to neither have information about intermediate results of the internal computation, nor the ability to influence the internal computation.

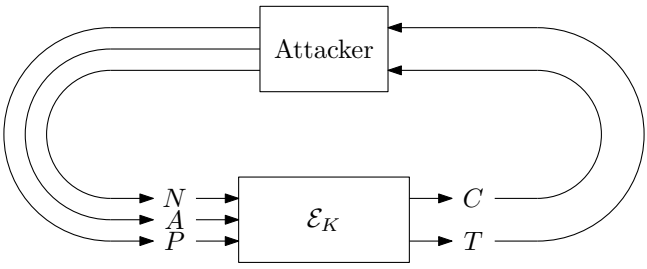


Figure 3.1: Basic concept of cryptanalytic attacks.

The aim for a real-world attacker is typically to break an authenticated encryption scheme, having probably the ultimate goal of recovering the secret key, which results in a total loss of confidentiality and authenticity. In contrast to a real-world attacker, whose interest is only the knowledge whether a scheme can be broken or not, the purpose of cryptanalysis is to get as much insight in the security of a cryptographic primitive as possible. This is typically done by evaluating different design choices and parameters that might influence the security of a cryptographic scheme, like the number of rounds for the building blocks or the rate r of designs following a MonkeyDuplex-based [23] design approach, but can also include the evaluation of the building blocks apart from their use in authenticated encryption schemes. Thus, cryptanalysis has typically a tight interaction with the design of primitives by evaluating the effects of design choices and changes in the security parameters (e.g., number of rounds) on the security of the scheme.

Over the years, various techniques have been proven to be useful in the process of analyzing primitives. Differential cryptanalysis, introduced by Biham and Shamir [27], and linear cryptanalysis, introduced by Matsui [88], are probably the most popular ones. Typically, differential cryptanalysis exploits knowledge that pairs of inputs having a certain difference lead to certain differences at the output (or intermediate values) of a cryptographic primitive, while linear cryptanalysis exploits good linear approximations of non-linear equations. Since finding and evaluating the probability of such differentials or linear approximations for larger structures like several rounds of a permutation or blockcipher is usually infeasible, differentials and linear approximations are typically evaluated for small structures like an S-box and "chained" together resulting in a differential or linear characteristic. Due to the popularity of differential and linear cryptanalysis, ciphers are expected to provide good arguments why they are secure against these techniques, which is typically achieved by a combination of proofs on upper bounds of the probability/bias of characteristics and heuristic searches for characteristics suitable for attacks.

However, quite naturally, it is usually not sufficient for a cipher to withstand one set of attacks, rather all have to be considered. For instance, in recent years, it has become more and more popular to design ciphers which use S-boxes with a low algebraic degree (e.g., Ascon [i2], LowMC [2], Ketje [25], Keyak [24], etc.). As a consequence, attacks directly targeting the algebraic description of the cipher become an interesting research field, which can be seen in the recent development of, e.g., cube-like [43] and conditional cube attacks [58]. Furthermore, it has become quite popular to base authenticated encryption schemes on tweakable blockciphers [60, 62, 76, 82]. Therefore, design strategies for dedicated tweakable blockciphers are explored, as, for instance, in the TWEAKEY framework [61]. However, compared to classic blockciphers, the insertion of the tweak results in additional freedom that can potentially be exploited by an attacker. Thus, dedicated analysis to investigate potential negative effects of the tweak on the security is needed.

Our contributions in the area of cryptanalysis (Part II of this thesis) are centered

around CAESAR. To aid the analysis of CEASER candidates, we have developed an automatic search tool [c7] dealing with the task of finding good linear characteristics suitable for various purposes. Automatic search tools have also been used in the analyses of the CAESAR candidates ICEPOLE and Ascon [c10, c13]. Additionally, we demonstrate with the forgery attack on round-reduced variants of ICEPOLE that in addition to good search tools, a cryptographer still has to find a good point to attack the scheme [c10]. Furthermore, we complement the analysis of Ascon with cube-like [43] attacks that are able to attack 6 (out of 12) rounds of the initialization [c13]. We also analyze the effect of tweaks in dedicated tweakable blockiphers. To be more precise, we analyze KIASU-BC, a dedicated tweakable blockcipher that underlies the CAESAR candidate KIASU [60] and builds upon AES-128 [107]. We show that the additional freedom introduced by the tweak allows to perform square attacks for one more round on KIASU-BC compared to AES-128, leading to an attack on 7 (out of 10) rounds of KIASU-BC.

We start this chapter with a discussion on security claims and goals of cryptanalysis (Section 3.1). The following Sections 3.2, 3.3, and 3.4 provide an introduction to differential and linear cryptanalysis, cube and cube-like attacks, and integral attacks, respectively.

3.1 On Security Claims and Goals

An important aspect for cryptographic primitives are the requirements that they should fulfill and in turn the security claim of the designer. Likely, different designers have different ideas what, for instance, the security requirements for an authenticated encryption scheme are. For example, let us consider the following requirements for an authenticated encryption scheme regarding the complexity of key recovery (breaking confidentiality and authenticity), tag forgery (breaking authenticity), and plaintext recovery (breaking confidentiality), which are based on the security requirements given by Bertoni, Daemen, Peeters, and Van Assche [19, Section 4.1.2]:

- **Key Recovery:** Assuming a key length of k bits, there should not be an attack significantly better than a brute force search for the key, where the probability of success after Y tries is $Y \cdot 2^{-k}$.
- **Tag Forgery:** The success probability for a forgery of a chosen quadruple (N, A, C, T) should not be significantly higher than 2^{-t}, where t is the bit-length of the tag. To be a forgery, the quadruple (N, A, C, T) has to be new, e.g., not produced by a legitimate holder of the secret key.
- **Plaintext Recovery:** Recovering information (beside its length) about the plaintext P out of quadruples (N, A, C, T) should be as hard as recovering the secret key, assuming that the nonce used for every encryption is unique.

Clearly, this interpretation of security requirements for authenticated encryption schemes is just one possible out of many and it results in a security strength that is fully determined by the size of the used keys and tags. However, not every construction is capable of fulfilling such requirements. Therefore, CAESAR [34] explicitly requires for each candidate to make a security claim and quantify the intended number of bits of security regarding its confidentiality and authenticity. According to NIST, the security strength (or "bits of security") is: "*A number associated with the amount of work (that is, the number of operations) that is required to break a cryptographic system*" [9]. Usually, this number is taken to the logarithm base 2 and the point of reference is left free to interpretation, but typically, one call to a primitive or building block is considered to be one operation.

Besides evaluating the security claim, the main task of cryptanalysis is to get as much insight in the security of a cryptographic scheme as possible. This process involves the analysis of weakened versions of the scheme, e.g., by reducing the number of rounds of underlying building blocks, or, as proposed for KETJE [25], to change the rate. Usually, a decrease in the number of rounds leads to a point where the round-reduced scheme does not hold up against attacks anymore. The difference between this point to the original proposed number of rounds is typically referred to as the security margin (see e.g., [39]). However, gathering insight in the security of a scheme does not stop at evaluating round-reduced versions, but also includes evaluating design decisions (e.g., the choice of the S-box), assessing the security of isolated building blocks, etc. Moreover, it is also important that all relevant design decisions are documented by the designers of a scheme. This allows cryptanalysts, e.g., to reconstruct the design process to a certain extend and to evaluate the decisions made in order to gather knowledge that can be used in future designs.

Such an analysis happens under various assumptions on the capabilities of an attacker, e.g., if the attacker just knows the transmitted data (N, A, C, T), or additionally knows the corresponding plaintext P, or is even able to manipulate inputs that get encrypted, cf. [98, Section 1.13.1]. However, in the case of nonce-based authenticated encryption, the manipulation of the public nonce might be limited. As mentioned in the CAESAR call for submissions [34], no assumptions on how the nonce is chosen should be made, except for the fact that it is unique for every encryption. Furthermore, all security can be lost if nonces are repeated. Thus, some attacks consider an attacker to be able to influence the nonce, or assume that the protocol uses a specific implementation of the nonce, while forbidding the attacker to use the same nonce twice. Nevertheless, it might also be interesting to evaluate the effects of such a nonce-reuse, which typically ranges from some loss of confidentiality up to key recovery. Note that some authenticated encryption schemes (e.g., Deoxys [62], AES-COPA [6]) also make security claims in such nonce-misuse scenarios.

An attacker additionally faces a restriction for the decryption. As we have seen in Chapter 1, the plaintext is just released to the user and thus, in a worse case,

to the attacker if the quadruple (N, A, C, T) is authentic. So, an attacker is unable to observe the effect of, e.g., a chosen ciphertext on the plaintext without creating a forgery in the first place. However, similar to the nonce-reuse, it is also of interest to evaluate the effect of a release of unverified plaintext on an authenticated encryption scheme [5].

3.2 Differential and Linear Cryptanalysis

Differential cryptanalysis by Biham and Shamir [27] and linear cryptanalysis by Matsui [88] are cryptanalytic techniques that have been introduced by showing attacks on DES [105]. These techniques can be seen as the root for a plethora of subsequently introduced techniques and concepts, amongst others zero-correlation [32], truncated differentials [68], boomerang attacks [142], or differential-linear attacks [79]. While arguably one of the most popular use-cases for differential and linear cryptanalysis is the analysis of blockciphers and hence, they are usually introduced on the basis of this example, those techniques are not limited to this single cryptographic primitive. To better fit to the context in Part II of this thesis, we will describe differential and linear cryptanalysis with the help of sponge-based authenticated encryption schemes [19] in the next section. This use of differential and linear cryptanalysis is not an invention of the author of this thesis and can probably be traced back to the results for hash functions given in the original paper of Biham and Shamir [27] in the case of differential cryptanalysis. Moreover, note that the explanations in Section 3.2.1 are highly simplified. However, from the point of view of the author of this thesis, they are a suitable and easy way to understand the basic concepts of differential and linear cryptanalysis. After the introduction to differential and linear cryptanalysis, we give an overview on strategies and tools for searching differential and linear characteristics to set our contributions in this area in context.

3.2.1 A Short Introduction to Differential Cryptanalysis and Linear Cryptanalysis

As described by Biham and Shamir [27], differential cryptanalysis is a technique that analyzes the influence that differences of input values have on the difference of the output (or some intermediate value) for a cryptographic primitive. Now, let us consider the unkeyed permutation p shown in Figure 3.2. For this permutation, we assume that we have knowledge that pairs of inputs (I_1, I_2) having difference $\Delta_I = I_1 \oplus I_2$ lead to output pairs having difference $\Delta_O = O_1 \oplus O_2$ with probability P_{Δ_I,Δ_O}. In short, we assume that we know a differential $\Delta_I \longrightarrow \Delta_O$ for the permutation p that holds with a certain probability P_{Δ_I,Δ_O}. Since we deal with an unkeyed permutation in this example, P_{Δ_I,Δ_O} depicts the relation of all plaintext pairs with difference Δ_I that lead to the output difference Δ_O to all possible plaintext pairs with difference Δ_I.

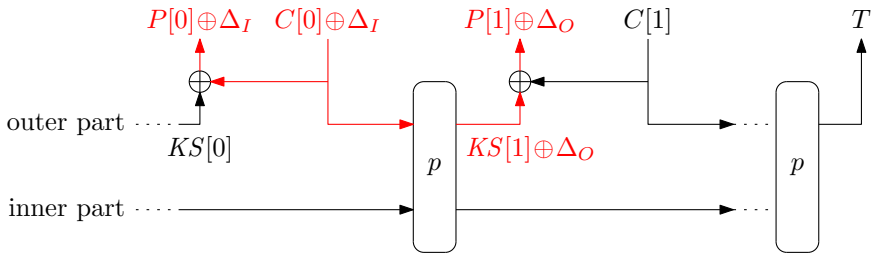


Figure 3.2: Concept of a forgery for a sponge-based authenticated encryption scheme.

We can use this differential [78] to create forgeries from known valid messages with probability P_{Δ_I,Δ_O} if Δ_I and Δ_O are free of differences for the inner part of the state. As shown in Figure 3.2, creating a second message (N_2, A_2, C_2, T_2) out of a known valid message (N_1, A_1, C_1, T_1) by just changing one ciphertext block to $C_2[0] = C_1[0] \oplus \Delta_I$ results in a valid forgery with probability P_{Δ_I,Δ_O}, where two plaintext blocks have differences compared to the original message, but the rest remains the same. A similar principle targeting the finalization is used in the analysis of round-reduced variants of the CAESAR candidate ICEPOLE-128 [101]:

- Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. "Forgery Attacks on Round-Reduced ICEPOLE-128". In: *Selected Areas in Cryptography – SAC 2015*. Ed. by Orr Dunkelman and Liam Keliher. Vol. 9566. LNCS. Springer, 2016, pp. 479–492. URL: `https://doi.org/10.1007/978-3-319-31301-6_27`

While differential cryptanalysis deals with the exploitation of differences between pairs of processed data, linear cryptanalysis [88] exploits good linear approximations of input and output bits (or bits of intermediate values) of cryptographic primitives. Since we only deal with unkeyed permutations in our example, we do not consider potential bits of (round) keys and effects of them. A linear approximation holds with a certain probability, where its quality is captured by the actual value of the deviation of the probability from 1/2. This deviation from 1/2 is called the bias ϵ. Such linear approximations are usually depicted as the inner product of vectors between linear masks α_I or α_O and the input x or output y:

$$\alpha_I \cdot x = \alpha_O \cdot y$$

Bits which are involved in the linear approximation ('1' entries in masks) are called active bits, whereas the other bits ('0' entries in masks) are called inactive.

Let us assume that we know a good linear approximation for the permutation p shown in Figure 3.2, which is characterized by the input and output masks α_I and α_O. If those masks do not have active bits involving the inner part of the

state, the linear approximation just involves bits of a ciphertext block, e.g., $C[0]$ and the keystream, e.g, $KS[1]$. This allows to distinguish the keystream used to encrypt $P[1]$ from a uniformly distributed random sequence.

Cryptanalysts typically face the problem of finding good and useful differentials and linear approximations covering larger structures (e.g., several rounds of a blockcipher or permutation). In fact, even evaluating the probability for such differentials and linear approximations is usually infeasible. A concept that is often used instead are differential and linear characteristics (also known as path, or trail) that have already been proposed in the original work of Biham and Shamir [27] and Matsui [88]. Please note that differential and linear characteristics are not only a substitute for differentials and linear approximations covering larger structures, but are a more versatile tool.

On a high level, the concept of differential and linear characteristics can be seen as a divide and conquer approach, where smaller parts of a cipher (e.g., a single round) are analyzed, typically leading to a fixation of intermediate differences or intermediate masks. Those intermediate differences or masks chained together form a differential or linear characteristic. For a differential characteristic, differences at the output of one round are the input differences for the next one and thus, have to match in order to form a valid characteristic. This is also the case for the intermediate mask in a linear characteristic. Non-matching masks would result in linear approximations involving bits of some intermediate variable, which is usually not wanted. A prominent strategy to estimate the probability of a differential characteristic is to multiply the probability of the differentials forming the differential characteristic [27], while in the case of linear characteristics the Piling-up lemma is used [88]. In both cases, the fulfillments of the individual differentials or smaller linear approximations are implicitly treated as independent events. In the next section, we give a brief overview on the automatic search for differential or linear characteristics.

3.2.2 Searching for Characteristics

While the manual search for characteristics can lead to good and useful results, searching manually is usually quite time consuming and prone to errors. Hence, several methods and tools that aid in the search for characteristics have been proposed over the past years. On a high level, the automatic search for characteristics can be categorized into approaches that build tools and programs from scratch and approaches that build upon existing MILP or SAT solvers. Furthermore, the aim of the resulting search tools varies. For example, there exist tools that aim at finding or bounding the probability of the best existing characteristic, whereas other tools are more heuristic in their nature and just search for good characteristics that are useful in attacks. If a tight bound on the probability can be given and the corresponding characteristic can be found highly depends on the analyzed primitive. Often, tight bounds can only be derived for a

small number of rounds, making the heuristic search for characteristics that cover more rounds necessary to get additional insight in the security of the primitive.

Already in 1994, Matsui [89] introduced a dedicated method that is capable of finding the best differential and linear characteristics for DES. Based on Matsui's insights, Biryukov and Nikolić [29] presented a tool searching for related-key differential characteristics that provides good results for byte-aligned ciphers. Biryukov, Velichkov, and Le Corre [30] deal with the automatic search for best characteristics for ARX-based primitives. Tools dedicated to specific primitives that typically exploit structural properties have been used in the case of KECCAK by Mella, Daemen, and Van Assche [93] and Daemen and Van Assche [40] in order to provide bounds on the probability. The seminal work of Wang and Yu [143] on the search for collisions for hash functions has lead to numerous tools capable of searching for differential characteristics. Those tools aided in the process of getting insight in the security of various hash functions like MD4 [50, 128], MD5 [131], SHA-1 [41], SHA-2 [c6, 47, 95, 97], SM3 [96], or Skein [80].

Besides designing search tools for characteristics from scratch, it has become more and more popular to base the search for differential and linear characteristics on MILP and SAT solvers. One of the first works utilizing mixed integer linear programming (MILP) to prove bounds on the minimum number of active S-boxes in a differential or linear characteristic has been described by Mouha, Wang, Gu, and Preneel [103]. This method has been extended to also prove bounds in the related-key setting for SPN-structures in [133]. Recently, MILP solvers and SAT solvers are also used in the search for differential and linear characteristics for ARX-based primitives [51, 83, 102]. As can be seen from the numerous results [51, 74, 83, 134, 135] for the lightweight primitives Simon and Speck, MILP- or SAT-based approaches excel whenever ciphers have a rather simple round function or small state size (or are structured). To complement MILP- or SAT-based tools and due to the lack of publicly available dedicated heuristic search tools for linear characteristics that are capable of dealing with weakly aligned [20] primitives having a large state size like ASCON, ICEPOLE, and KEYAK, we have developed the following publicly available tool:

- Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. "Heuristic Tool for Linear Cryptanalysis with Applications to CAESAR Candidates". In: *Advances in Cryptology – ASIACRYPT 2015*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9453. LNCS. Springer, 2015, pp. 490–509. URL: https://doi.org/10.1007/978-3-662-48800-3_20

3.3 Cube and Cube-like Attacks

Cube attacks introduced by Dinur and Shamir [44] exploit the fact that the single output bits of most cryptographic algorithms can be expressed as polynomial functions of the input bits. Typically, such a polynomial is depicted in its

algebraic normal form (ANF), being the sum of terms, where each term is a product of the input bits (or a constant). The input bits of a cryptographic primitive can be split into public bits V_i that can be influenced (e.g., plaintext, nonce), public bits C_i that are constant (e.g., constant initial value), and secret bits K_i that are typically constant (e.g., the key). The goal of a cube attack is now to find the values of the secret bits. In a cube attack, some of the public input bits V_i are now selected as cube variables $CV_i = V_i$, while others are set to a constant value. If the product of the cube variables $\prod(CV_j)$ is now factored out of all terms where it appears, we can express one output bit O_i as:

$$O_i = \prod(CV_j) \cdot p_S + q$$

Here, p_S is called the superpoly and q is a polynomial where in each term at least one of the cube variables CV_i is missing. In a cube attack, we now calculate the sum of O_i for each possible assignment of our cube variables CV_i. The resulting cube sum is then just the evaluation of the superpoly p_S, because the superpoly influences the output bit O_i just one time when all CV_i are 1, whereas all terms appearing in q are summed an even number of times since at least one cube variable per term is missing. Dinur and Shamir [44] demonstrate that a careful selection of the cube variables leads to situations where the superpoly p_S is just a linear function of the secret bits K_i. If the cube sum is known, such linear functions can be used to reveal the secret key. It is worth noting that the underlying working principles of cube attacks are similar to, e.g., higher order differential cryptanalysis introduced by Lai [77] or an algebraic IV differential attack (AIDA) by Vielhaber [141].

Later, Dinur, Morawiecki, Pieprzyk, Srebrny, and Straus [43] published cube-like attacks, where the superpoly does not have to be a linear function of the secret bits k_i; in fact, the presented attacks also work if it is a non-linear function. In the presented cube-like attacks, the cube variables are chosen so that it is known which secret bits k_i appear in the superpoly. With this knowledge, a fingerprint for each possible assignment of the involved k_i can be created with the help of the cube sum in a pre-processing phase, which in turn is used in an attack to recover parts of the secret key. We apply this strategy to analyze round-reduced variants of Ascon:

- Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. "Cryptanalysis of Ascon". In: *Topics in Cryptology – CT-RSA 2015*. Ed. by Kaisa Nyberg. Vol. 9048. LNCS. Springer, 2015, pp. 371–387. URL: `https://doi.org/10.1007/978-3-319-16715-2_20`

Recently, Li, Dong, and Wang [81] present conditional cube attacks on round-reduced variants of Ascon that improve upon the cube-like attacks presented in [c13]. Conditional cube attacks have been introduced by Huang, Wang, Xu, Wang, and Zhao [58]. These attacks exploit knowledge about the appearance of

the product of all cube variables $\prod(CV_j)$ in the terms of the output functions. If this product does not appear, the superpoly is empty and the cube sum is guaranteed to be zero, whereas if it appears, the cube sum of a bit can also be 1. By carefully selecting cube variables and analyzing their distribution and propagation within the first few rounds of a targeted cipher, necessary conditions (certain values that input bits have to take) for $\prod(CV_j)$ not appearing in the terms can be identified. If secret bits are needed to fulfill these conditions, cube sums of an attacked cipher give information about those secret bits.

3.4 Integral Attacks

The square attack has been developed for the blockcipher Square [37] and exploits its byte-aligned structure. The concept of the square attack is also applicable to AES [39], which has later been improved in [49] and generalized as integral attacks by Knudsen and Wagner [69]. In Part II of our thesis, we only rely on concepts introduced in the attack on Square [37] and extensions of it on AES-128 [49]. Hence, we decided to stick with the name square attack for this attack. In this section, we explain the working principles of such a square or integral attack based on the work of Daemen and Rijmen [39] on AES-128 first. Then, we discuss the relation to our work on KIASU-BC, give recent attacks on KIASU-BC and finally, discuss the division property as generalization of the properties exploited in square or integral attacks, which might also be considered in future work on KIASU-BC.

AES-128 [107] is a blockcipher that operates on a state arranged as a 4×4 square of bytes. Its round function consists of the iterative application of SubBytes, ShiftRows, MixColumns, and AddRoundKey. In a square attack, the encryption of a set of 256 plaintexts is now traced. When looking at a single byte position in this set, the byte is called active `A` if the values of the set at its position iterate over all possible 256 values, constant `C` if the values of the set at its position always have the same value and balanced `B` if the xor sum over the set at this position is zero.

The properties of a set can be influenced by the round functions. For instance, AddRoundKey does not change the properties of a set, while ShiftRows just changes the position of the bytes. However, a closer look has to be taken on SubBytes and MixColumns.

SubBytes is the byte-wise application of a bijective S-box to the state. Thus, if the input of the S-box iterates over all possible values, also the output iterates over all possible values. Hence, an active byte `A` stays active. Furthermore, the same input always results in the same output. Therefore, a constant byte `C` stays constant. However, since the S-box is a non-linear transformation, it is not guaranteed that an input set which sums to zero leads to an output that sums to zero. So, the balanced property vanishes in general.

MixColumns is a linear transformation applied to each column of the state separately. Thus, we can conclude that a column of constant bytes at the input leads to constant bytes at the output. Each output byte of MixColums is calculated as a linear function of the 4 input bytes, where each coefficient of the function is invertible. Thus, having one byte active while the other three bytes are constant results in an all-active column. Moreover, if all bytes at the input are active and so each byte sums to zero, also the output sums to zero, which leads to the knowledge that the bytes at the output are at least balanced.

This knowledge can be used to construct a 3-round distinguisher for AES-128 (Figure 3.3), where one active and 15 constant bytes at the input lead to an all-balanced state after 3 rounds.

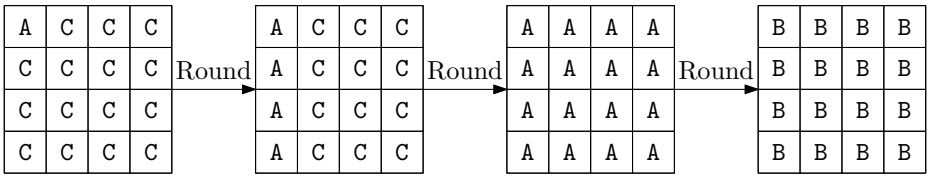


Figure 3.3: Square distinguisher.

For KIASU-BC, we show that the additional freedom introduced by the tweak allows to extend this 3-round distinguisher by one more round. Thus, it is possible to attack 7 (out of 10) rounds of KIASU-BC. Furthermore, a variant of this attack is also applicable to a round-reduced variant of the authenticated encryption scheme KIASU≠:

- Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. "Square Attack on 7-Round Kiasu-BC". In: *Applied Cryptography and Network Security, ACNS 2016*. Ed. by Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider. Vol. 9696. LNCS. Springer, 2016, pp. 500–517. URL: https://doi.org/10.1007/978-3-319-39555-5_27

Subsequent work shows that the freedom introduced by the tweak can also be exploited to improve other attack techniques, so that also one more round can be gained compared to the same technique applied to AES-128. Tolba, Abdelkhalek, and Youssef [139] show meet-in-the-middle attacks targeting round-reduced versions of KIASU-BC with 8 rounds, impossible differential attacks are demonstrated to work for 8 rounds of KIASU-BC independently by Dobraunig and List [c15] and by Abdelkhalek, Tolba, and Youssef [1], and Boomerang attacks are shown for 8 rounds by Dobraunig and List [c15].

Recently, a generalization of the properties used to construct integral distinguishers has been proposed under the name division property by Todo [138]. The division property [138] characterizes multisets of n-bit binary values x_j with the help of the bit product function $\pi_u(x_j) = \prod_i (x_j[i]^{u[i]})$. This function processes

the single bits $x_j[i]$ and $u[i]$ of the n-bit values x_j and u. The outcome of this function is the binary AND of the value $x_j[i]$ for positions where $u[i] = 1$. A multiset of values x_j has the division property $\mathcal{D}_k^n$ if it is fulfilled that $\bigoplus_j \pi_u(x_j) = 0$ for all u with a Hamming weight smaller than k. The division property allows to capture the properties used in integral attacks. For instance, the property that a set contains all values can be captured by $\mathcal{D}_n^n$, or the property that the values in a multiset sum to 0 can be captured with $\mathcal{D}_2^n$.

As demonstrated by Todo [138], the division property provides a nice framework to construct integral distinguishers for ciphers, especially if S-boxes (or functions in general) are non-bijective or have a low algebraic degree d. Todo shows that an input multiset with division property $\mathcal{D}_k^n$ leads to a multiset at the output of an S-box having division property $\mathcal{D}_r^n$ with $r = \lceil k/d \rceil$. Furthermore, a multiset preserves the division property $\mathcal{D}_n^n$ for bijective S-boxes.

The division property has been used in various attacks, most prominently in the attack on full MISTY1 [137]. Besides this result, we see a constant development of new tools and methods that aid in the search for integral characteristics using the division property, amongst others the work of Zhang and Rijmen [146] and Sun, Wang, and Wang [132]. Often, the found integral distinguishers improve upon previous results, which construct integral distinguisher just relying on the concepts introduced in square or integral attacks that we use in our analysis of KIASU-BC. Hence, we consider the application of the division property on KIASU-BC as an interesting research topic, especially when considering potential exploitations of the interaction of the tweak and the AES round function.

4

Side-Channel Attacks

In this chapter, we deal with side-channel attacks. In contrast to cryptographic attacks discussed in Chapter 3, here, we assume that an attacker has also limited information about the internal processing of cryptographic algorithms via side-channels (see Figure 4.1). Side-channel attacks became popular in the 1990's when attacks exploiting the timing information [71] and, especially, power analysis attacks like simple power analysis (SPA) and differential power analysis (DPA) were introduced [72]. In particular for power analysis attacks, it turned out that when this side-channel is not taken into consideration during the design and implementation of a cryptographic algorithm, an attacker is almost always able to extract the secret from a device that executes the algorithm if the attacker is able to observe the device's power consumption.

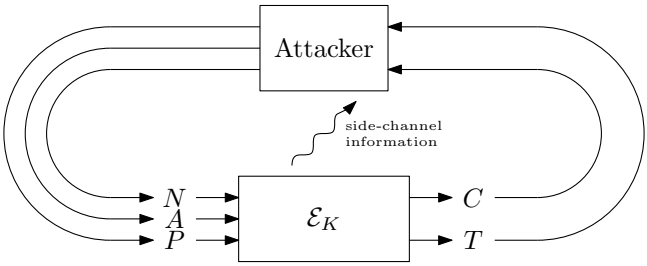


Figure 4.1: Concept of side-channel attacks.

From this point on, a rich research field emerged, proposing a plethora of different

countermeasures and attacks. While the exploitation of timing side-channels of symmetric cryptographic primitives [14, 114, 140] has become less important due to the introduction of fast implementations that have a data-independent timing behaviour [66], power analysis attacks — especially DPA — are still a threat to deployed implementations of cryptographic algorithms. Two recent examples of DPA targeting deployed devices are attacks targeting FPGAs [100] and smart lamps [123]. In the first attack, Moradi and Schneider recover the secret key that is used to decrypt bitfiles for FPGAs [100], while in the second attack, a sophisticated DPA is used to extract the global AES-CCM key that is used to encrypt and authenticate firmware updates [123].

Although side-channel attacks target the implementation of an algorithm rather than the algorithm itself, taking countermeasures against side-channel attacks into account during the design process leads to constructions that are "easier" to protect. In this context, the term "easier" has a wide range of meaning that typically depends on what a designer wants to achieve. For instance, modern authenticated encryption schemes like Ascon [i2] or Keyak [24] allow fast constant-time implementations, so that no data-dependent look-ups are necessary to achieve fast implementations in software. Another trend in authenticated encryption is to use building blocks (e.g., blockciphers, permutations, etc.) which are "easier" to mask, as indicated in the design documents of, e.g., Ascon [i2], PRIMATEs [4], and SCREAM [56]. Here, the term "easier" typically refers to overhead in terms of either space (e.g., additional area to store shares for hardware implementations) or time (e.g., additional time needed to process shares).

In this thesis, we explore further options to harden authenticated encryption schemes against side-channel attacks. The outcome of this research is Isap [j2]. Isap incorporates principles of fresh re-keying [92] to provide protection against DPA. A crucial part that leads to Isap are the insights gathered during our research on fresh re-keying schemes in [c14] and [c3].

The purpose of this chapter is to make the reader familiar with the high-level concepts of side-channel attacks and countermeasure against them in general, complemented by a more detailed look on fresh re-keying. For a more detailed description of power analysis attacks and masking, we refer to the work of Mangard, Oswald, and Popp [85] and more recent publications like [119]. Furthermore, we refer the reader to the thesis of Neve [110] that covers the content presented in Section 4.1 about cache attacks in more detail.

This chapter starts with an overview on side-channel attacks exploiting timing information in Section 4.1, followed by power analysis attacks in Section 4.2. After that, we give a short introduction to masking, which is a prominent countermeasure against power analysis attacks in Section 4.3. Finally, we discuss re-keying-based countermeasures in Section 4.4.

4.1 Timing Attacks

In the seminal work of Kocher [71] on timing attacks, he demonstrates that variations in the timing behaviour of cryptographic primitives caused by the processed data can be used to extract secret information. Concretely, Kocher attacks a simple modular exponentiator. Besides this attack, Kocher's paper contains the following remark that turned out to have a significant impact on the design of modern symmetric encryption schemes: *"RAM cache hits can produce timing characteristics in implementations of Blowfish[...], SEAL[...], DES, and other ciphers if tables in memory are not used identically in every encryption"* [71]. Kocher refers here to potential problems of cipher implementations using tables, which are caused by the memory hierarchy. In short, there exists a measurable time difference if the data of accessed table entries is located in the cache (cache hit), or if it has to be loaded from main memory (cache miss).

Among the first practical attacks that make use of access dependent timing behaviour is an attack on DES by Tsunoo, Saito, Suzaki, Shigeri, and Miyauchi [140]. Their attack relies on the following observation. Let us assume that the attacked cipher is implemented in software and uses tables to implement the S-boxes. Because of the fact that tables are used, the time needed for a single encryption depends on the number of cache hits, or respectively on the number of cache misses. A short encryption time indicates more cache hits, whereas a longer encryption time indicates fewer cache hits. If no S-box data is present in the cache at the start of the encryption, more cache hits indicate that the inputs of the used S-boxes are more likely to be similar, whereas less cache hits indicate that they are more likely to be different. A convenient way to see how this observation translates to an attack is to have a look at the toycipher example (Figure 4.2) used in [140], which only has two S-boxes.

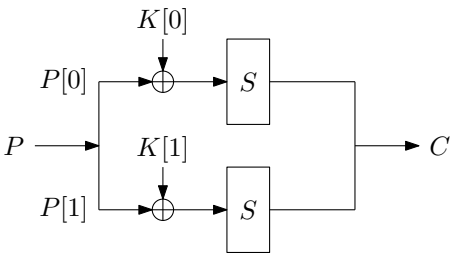


Figure 4.2: Toycipher with two S-boxes used in [140].

If we now encrypt several different plaintexts P and measure the time needed for encryption, we can observe differences in the encryption time. These differences in the encryption time are likely caused by differences in the number of cache hits. In the concrete example, we only observe a cache hit if $P[0] \oplus K[0] = P[1] \oplus K[1]$. Thus, we can deduce from plaintexts with a short encryption time the difference of $K[0] \oplus K[1] = P[0] \oplus P[1]$. For plaintexts characterized by a long encryption

time, it can be assumed that the inputs of the two S-boxes are different and thus $K[0] \oplus K[1] \neq P[0] \oplus P[1]$. Furthermore, it is demonstrated in [140] that this concept translates to more complex constructions by attacking a specific implementation of DES executed on a Pentium III.

Later on, table-based implementations of AES have been attacked independently by Bernstein [14] and Osvik, Shamir, and Tromer [114]. Back then, the fastest way to implement AES in software was making use of four 1024-byte tables (nowadays the fastest way to use AES in software is usually to make use of AES instructions, or use fast bitsliced implementations [66] that provide protection against timing attacks). Bernstein's attack targets a server that uses a table-based AES implementation and processes data received from clients. Assuming that the processing time of the server is somehow related to the actual input that is processed by the tables in the first round of AES, an attacker could learn those timings from a server using a known key, and utilize this information on an identical setup to retrieve information about the secret key. In an attack of Osvik, Shamir, and Tromer [114], the attacker is assumed to be able to run an unprivileged process in parallel to the encryption on the same processor. By observing the cache access patterns of the encrypting process, information about the used secret key can be deduced.

As observed by Bernstein [14] and Osvik, Shamir, and Tromer [114], one countermeasure against this type of attacks is to use (or design) ciphers that facilitate fast constant-time implementations. This phrase simply means that only operations which have an execution time that is independent of the values of their operators on most platforms should be used, like bitwise Boolean functions, modular addition, or shifts and rotations by a constant value. In particular, data dependent table look-ups or data dependent rotations should be avoided. Examples for primitives that allow fast implementations only relying on bitwise Boolean functions and rotations by constant values are among others the permutations used in Ascon [i2] or Keccak [16].

4.2 Power Analysis Attacks

Power analysis attacks are known to a broader audience since the late 1990s when Kocher, Jaffe, and Jun [72] introduced simple power analysis (SPA) and differential power analysis (DPA). In principle, it turned out that an attacker can extract the secret of a device if the attacker is able to observe side-channels like the power consumption [72], or electro magnetic radiation [118] during the execution of a cryptographic algorithm. The reason for power analysis attacks to work is the fact that the power consumption of a device executing a cryptographic algorithm usually depends on the data that is processed [85]. Since the processed data is usually a mixture of known and secret components, the power consumption leaks information of the secret. In this section, we give a brief overview on the working principles of simple power analysis (SPA) and differential power analysis

(DPA). The presented content can also be found in the work of Mangard, Oswald, and Popp [85] to which we refer for a more detailed description of the various attacks.

4.2.1 Simple Power Analysis

Kocher, Jaffe, and Jun [72] refer to SPA as an attack technique that involves the direct interpretation of the measurements collected during the execution of a cryptographic algorithm. As shown in their paper, it can be possible to deduce the different stages of cryptographic algorithms for unprotected implementations down to the revelation of the actual sequence of instructions that are executed. They give various examples where an SPA can be used in such cases. For instance, they describe attacks targeting the key schedule of DES if the key schedule is implemented in the following way.

For the DES key schedule, the 28-bit halves of the key are rotated to the left in each round as part of the key schedule to produce the single round keys. Let us assume that a rotation to the left by one bit is implemented by a left shift and a conditional check if the bit that is shifted out is 1 or not. Depending on this decision, different code is executed. If an attacker can distinguish this code with the help of measurements, single bits of the secret key can be recovered.

Another SPA that targets the key schedule of a blockcipher has been demonstrated by Mangard [84]. This attack focuses on the key schedule of AES and — in contrast to the attack above — does not exploit conditional branching. Instead, the fact that certain devices leak the Hamming weight of intermediates is utilized. Mangard [84] shows that the leakage of Hamming weight information of single round key bytes can be rather efficiently combined via the key schedule to significantly reduce the number of key candidates for a brute force search.

The two presented SPAs are just a small fraction of all possible variants and published attacks. Nevertheless, they give insight into the principles and assumptions behind an SPA. Typically, an SPA requires detailed knowledge about the implementation of the attacked algorithm and possibly about the hardware of the attacked device. Next, we will have a look at a more powerful class of attacks presented by Kocher, Jaffe, and Jun [72], differential power analysis (DPA).

4.2.2 Differential Power Analysis

In contrast to simple power analysis (SPA), differential power analysis (DPA) exploits differences in the power consumption caused by different inputs to the cryptographic algorithm at a fixed moment of time [85]. For that, an attacker usually makes predictions about the power consumption of intermediate variables under a certain key guess and evaluates these predictions against real measurements. The advantage of a DPA lies in the ability that the key can be

guessed in fractions of a size that is typically tied to the size of the used S-box for the cipher. For such attacks to work, an attacker has to be able to observe a device processing many different inputs under the same key.

In their seminal work on power analysis, Kocher, Jaffe, and Jun [72] introduce DPA by attacking DES. In this attack, they make predictions about a single bit at the output of an S-box in the last round and expect to see a difference in the power consumption whether this bit is 0 or 1. To calculate this bit from known ciphertexts, 6 bits of the last round key have to be guessed. In an attack, many power traces for different plaintexts are recorded. For each guess of 6 bits of the round key, those traces are separated into two groups, where the targeted bit of the S-box output is calculated to be 0 or 1. For the correct key guess, the respective 0 and 1 bits are assigned to the correct groups, while this is not necessarily true for other guesses. Hence, if the difference of the means of the power traces of both groups is calculated, the resulting differential trace should have spikes in regions where the power consumption depends on the value of the targeted bit for a correct key guess.

From this point on, many variants and extensions of differential power analysis have been developed, too many to be covered in this work. However, Mangard, Oswald, and Standaert [86] provide a nice description of the underlying working principle of a class of DPAs to which they refer to as standard DPA. Next, we will restate this description taking an SPN-based blockcipher (sketched in Figure 2.6) as example. As mentioned by Mangard, Oswald, and Standaert [86], a standard DPA is characterized by executing the following three points:

- **Prediction of intermediate values:** In the first step, an attacker calculates intermediate values for a certain key guess. For the example in Figure 2.6, an attacker could target the output of one S-box of the first round. Thus, the attacker calculates the value $S_0[0]$ under the guess of the partial key $K_0[0]$ for every different plaintext P for which power measurements have been conducted.
- **Prediction of power consumption:** In this second step, the power consumption that leads to or is caused by the predicted intermediate values of $S_0[0]$ is modeled. Often, simple models like the Hamming weight of the predicted intermediate value $S_0[0]$ or the Hamming distance are used.
- **Evaluation of prediction against measurements:** In this step, the predicted power consumption is evaluated against the measured one for each plaintext P and partial key guess $K_0[0]$. For this evaluation, various statistical tests can be used. The partial key $K_0[0]$ that produces the best result in this evaluation is most likely the correct one.

In the following sections, we will have a look at two classes of countermeasures against side-channel attacks, namely countermeasures based on masking and rekeying. The countermeasures based on masking of Section 4.3 attempt to achieve

independence of power consumption from the value of intermediate variables by splitting the single intermediate variables into s shares. Countermeasures based on re-keying (Section 4.4) on the other hand aim to limit the usage of the secret key. Hence, the number of measurements per secret key is limited in the most extreme case to one measurement, which makes side-channel attacks harder to execute and usually precludes techniques that require measurements of a device processing different inputs under the same key like DPA.

4.3 Masking

The idea of masking [35, 55] is to make the power consumption of the device processing a cryptographic algorithm independent of the values of the intermediate variables [85]. Already in the work of Chari, Jutla, Rao, and Rohatgi [35], it is proposed to split the intermediate variables v of a cryptographic algorithm into s shares v_i so that the shares v_i recombine to v with respect to a certain operator $\circ$ and to execute the algorithm by just using these shares v_i without reconstructing the original value v:

$$v = v_1 \circ v_2 \circ v_3 \circ \ldots \circ v_{s-1} \circ v_s$$

The operation that is used to split and recombine v into the shares v_i depends on the operations used in the algorithm that has to be protected. For instance, Boolean masking, where the xor operation is used to share the variables, has turned out to be quite suitable for many ciphers. Chari, Jutla, Rao, and Rohatgi [35] require that the single shares are uniformly distributed and that any subset of $s-1$ shares v_i is statistically independent of v. If we do not consider the processing of the shares, this requirement ensures that an attacker who learns at most $s-1$ shares of v_i via side-channels cannot reconstruct v.

While most masking schemes agree upon the uniformity requirement of the initial sharing (for Boolean masking) and the fact that functions f that are linear with respect to the sharing should be processed per share $f(v) = f(v_1 \oplus v_2 \oplus \ldots \oplus v_s) = f(v_1) \oplus f(v_2) \oplus \ldots \oplus f(v_s)$, there exists a myriad of masking schemes proposing ways of, e.g., handling the non-linear parts, or the injection of fresh randomness having different assumptions about the underlying hardware and using different models to prove the masking schemes secure. One of the first and still important schemes that comes with a proof is the scheme of Ishai, Sahai, and Wagner [59]. In the case of Boolean masking, they show a general concept how to transform a cryptographic algorithm into a masked scheme that can withstand an attacker probing a limited amount of bits (or wires) per clock cycle. As later observed by Mangard, Popp, and Gammel [87], Ishai, Sahai, and Wagner [59] assume that the information on such wires can just change once per clock cycle, which is not necessarily the case for hardware implementations due to glitches. A method that works on a more abstract level and provides protection in the presence of glitches are threshold implementations [111–113].

The idea of threshold implementations [111–113] is to split an input v into shares v_i and compute a function $z = f(v)$ decomposed into functions f_i in a way that a first-order DPA on an implementation cannot be performed. For withstanding first-order DPA, a masked implementation of f following the ideas of threshold implementations has to fulfill the two properties correctness and non-completeness. For being correct, the output shares z_i should recombine to the correct value z. Non-completeness means that the computation of an output share z_i using function f_i should be independent of at least one input share v_i. Hence, an attacker who is just able to observe the execution of one function f_i cannot learn information about v assuming v has been shared uniformly.

To sum up, this section contains a basic and probably oversimplified view on the working principles of masking. For more details about masking we refer to [85, 119]. Clearly, the challenge of masking is to provide implementations of cryptographic algorithms that protect against DPA attacks, while minimizing the overhead induced by the masking. This overhead depends amongst other things on the number of shares used, additional computational overhead for processing the shares, and the required number of random bits. However, the resulting overhead also depends on the cryptographic algorithm that has to be masked, e.g., on the choice of the S-box. Designers of new schemes tend to take this fact into account as indicated in the design documents of, e.g., Ascon [i2], PRIMATEs [4], and SCREAM [56].

4.4 Re-keying

While the idea of masking is to make the power consumption of a device independent of the data it processes, the idea of re-keying based countermeasures is to limit the number of measurements an attacker is allowed to make per secret key. While not as prominent as masking, the idea of re-keying is mentioned by Kocher [70] and in the book "*Power Analysis Attacks*" by Mangard, Oswald, and Popp [85].

In this section, we will focus on a countermeasure called fresh re-keying [92] and extensions of it. First, we give an introduction to the ideas and concepts of fresh re-keying. Then, we discuss the fact that some instantiations of fresh re-keying, or re-keying in general, are susceptible to time-memory trade-off attacks. After that, we put the contributions included in Part II in this area in context and detail why fresh re-keying might lead to interesting constructions in the field of authenticated encryption.

4.4.1 The Concept of Fresh Re-keying

Fresh re-keying has been proposed in [92] as low-cost countermeasure against differential power analysis (DPA) and differential fault attacks (DFA), where

this section focuses on its side-channel characteristics. To provide protection against DPA, fresh re-keying deploys a "separation of duties" principle, where a cryptographic function (in the simplest case just one blockcipher call E) is protected (re-keyed) by a function g that is easy to protect against side-channel attacks (Figure 4.3).

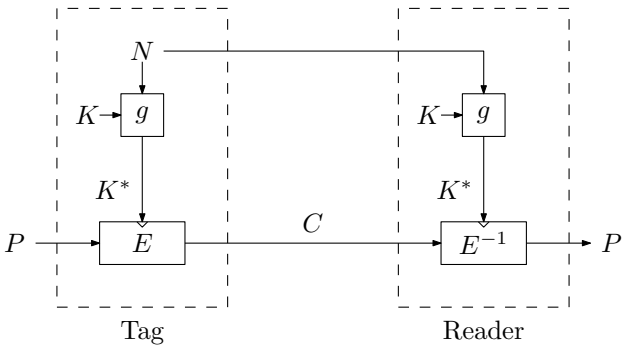


Figure 4.3: Fresh re-keying as proposed in [92] (figure based on [c3]).

Figure 4.3 depicts a communication scenario, where a low-cost RFID tag, requiring cheap protection against DPA, communicates with an RFID reader, where more costly countermeasures like masking of the blockcipher can be deployed. Every time the tag wants to encrypt a plaintext P, it generates a fresh nonce N to derive a session key K^* from a static master key K. This session key K^* is then used to encrypt the plaintext P to get the ciphertext C. The ciphertext C is sent together with the nonce N to the reader, where it can be decrypted again.

First, we focus just on the blockcipher call on the side of the tag. Here, using a fresh session key K^* ideally limits the exploitable measurements to one per session key. Thus, assuming that one measurement per key is not sufficient to perform a DPA, an attacker can just perform SPA and hence, the blockcipher has "just" to be protected against this class of attacks. Since the session key is derived from the public on tag generated nonce N and the static master key K, the problem of protection against DPA is shifted to g. Therefore, g requires protection against both, SPA and DPA. If we take a look on the reader, we see that the reader has no means to influence the nonce N and usually has to decrypt any combination of received nonce N and ciphertext C. So, it cannot be assumed for the reader that a session key is not used more than once in combination with different ciphertext. Thus, also the blockcipher requires protection against DPA.

The question that we have not answered yet is which function to use for g. Ideally, we want to use functions which behave like a pseudo-random function to generate the session keys K^*, for instance constructions based on GGM trees [54] as proposed in [130]. However, the usage of such a function would not result in a "cheap" protection of a single blockcipher call. Thus, it is argued in [92] that also "cryptographically weak" functions can serve as re-keying functions g, while

providing protection against DPA for the blockcipher. In [92], six properties that such a function g should fulfill are listed, which we restate next:

1. **Good Diffusion**: The re-keying function should provide good diffusion, so that one bit of the session key K^* depends on several bits of the master key K.
2. **Synchronization Free**: The session key K^* should only be derived from a static master key K and a nonce N without the need of other secret internals, which probably have to be synchronized between parties.
3. **No Additional Key Material**: The size of the session key K^* should match the size of the master key K in bits.
4. **Small Implementation Overhead**: The overall construction should be cheaper than directly protecting the blockcipher by using, e.g., masking.
5. **Easy Protection Against Side-Channel Attacks**: The re-keying function should facilitate easy protection against SPA and DPA by, e.g., masking.
6. **Regularity**: The re-keying function should provide a regular structure in order to be easier to implement.

It is proposed by Medwed, Standaert, Großschädl, and Regazzoni [92] to use a polynomial multiplication as re-keying function g. They argue that such a multiplication allows the efficient implementation of countermeasures like shuffling and masking against SPA and DPA. Subsequent work [91] proposes re-keying functions that are able to process multiple nonces, so that multiple parties can contribute to the generation of the session key K^* and thus, can be protected against DPA.

4.4.2 Time-Memory Trade-off Attacks on Re-keying

In this section, we shortly summarize time-memory trade-off attacks on fresh re-keying that are capable of recovering the secret master key [c4]. Please note that the underlying principle of the attack is neither new nor limited to fresh re-keying and has already been applied in scenarios where a blockcipher is re-keyed as already demonstrated by Biham [26], or — in a more general sense — is used in many attacks (or is part of attacks) where the key is mixed with a known variable value in an unfortunate way via a function that is easy to invert [c1, 94].

To demonstrate the attack, let us assume a simple challenge-response protocol using a blockcipher with a matching key size k and block size n, where the reader sends a challenge P to the tag in Figure 4.3 and the tag replies with the pair (N, C), where $C = E_{K^*}(P)$ and $K^* = g_K(N)$. Now, an attacker can execute the following key recovery attack by impersonating the reader:

- **Offline Phase:** In the offline phase, the attacker creates a list having t entries of pairs (C_i, K_i^*), where C_i is generated by encrypting always a constant challenge P with changing K_i^*.

- **Online Phase:** In the online phase, the attacker queries the RFID tag t' times with the constant challenge P used to generate the list during the offline phase. The tag replies with a pair (N_i, C_i). If a ciphertext appears in the list, the attacker can recover the session key K_i^* and the corresponding nonce N_i. Depending on the used re-keying function g, an attacker can use N_i and K_i^* to recover the master key K.

The attack complexity depends on the size of the list t, which in turn determines the expected number of online queries t' needed before we find a matching entry in the list. The trade-off that needs the least amount of $2 \cdot 2^{k/2}$ total blockcipher calls uses a list of size $2^{k/2}$ followed by $2^{k/2}$ online queries. Finding ways of preventing this attack on re-keying schemes is the goal of the following publication, which is included in Part II:

- Christoph Dobraunig, François Koeune, Stefan Mangard, Florian Mendel, and François-Xavier Standaert. "Towards Fresh and Hybrid Re-Keying Schemes with Beyond Birthday Security". In: *Smart Card Research and Advanced Applications, CARDIS 2015*. Ed. by Naofumi Homma and Marcel Medwed. Vol. 9514. LNCS. Springer, 2016, pp. 225–241. URL: https://doi.org/10.1007/978-3-319-31271-2_14

4.4.3 Side-channel Aspects of Re-keying

Choosing a suitable re-keying function, which together with the re-keyed cryptographic primitive gives a construction that makes the life of a side-channel attacker as hard as possible, is not a trivial task. While the straightforward application of known standard side-channel attacks is usually precluded, other attacks might still be feasible. In the case of the multiplication used in [92], several side-channel attacks have been shown [11, 12, 115].

Those attacks build upon the insight that the polynomial multiplication used in [92] can be rewritten as a system of linear equations in GF(2). An attacker who is able to get noisy information about the single bits of the session key K^* by, e.g., observing its use in the blockcipher E, faces a similar problem as in the learning parity with noise (LPN) problem [116]. For a detailed description of the attacks we refer to the original publications [11, 12, 115].

Alternative proposals include re-keying functions which base their security on hard problems like learning parity with leakage (LPL) or learning with errors (LWE) [45], or Keymill [136], a construction based on NLFSRs. For this construction, it is claimed that it is secure against side-channel attacks without needing

additional circuits. In particular, it is claimed that in a DPA, the key hypothesis has to be done for the whole key, not allowing a divide and conquer approach. However, in the following publication shown in Part II, we show a side-channel attack by recovering information equivalent to the secret key. Concretely, we recover differences of neighboring bits of the used shift-registers:

- Christoph Dobraunig, Maria Eichlseder, Thomas Korak, and Florian Mendel. "Side-Channel Analysis of Keymill". In: *Constructive Side-Channel Analysis and Secure Design, COSADE 2017*. Ed. by Sylvain Guilley. Vol. 10348. LNCS. Springer, 2017, pp. 138–152. URL: `https://doi.org/10.1007/978-3-319-64647-3_9`

4.4.4 Re-keying and Authenticated Encryption

Recently and independently from our work, several authenticated encryption schemes have been proposed that address the need for protection against side-channel attacks. For instance, Berti, Koeune, Pereira, Peters, and Standaert [15] and Barwell, Martin, Oswald, and Stam [10] propose leakage resilient authenticated encryption schemes. Berti, Koeune, Pereira, Peters, and Standaert [15] present a construction that achieves leakage resilience of the authenticated encryption, but leaves the option to attack the authenticated decryption out of scope. With Isap [j2], we wanted to focus also on authenticated decryption and propose a sponge-based authenticated encryption scheme that withstands DPA for both, encryption and decryption, by incorporating concepts from (fresh) re-keying. Additionally, Isap reduces the attack surface against SPA by increasing the capacity of keyed parts compared to the minimum required from a cryptographic perspective.

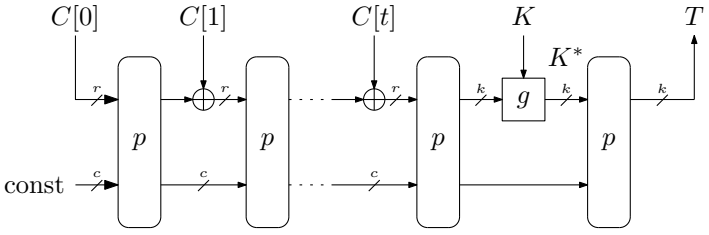


Figure 4.4: Sketch of the MAC used in Isap [j2].

Isap is an encrypt-then-MAC construction that uses a sponge-based suffix MAC (sketched in Figure 4.4) and a sponge-based streamcipher. For encryption, DPA is prevented by the requirement that the nonce has to be unique and thus, keys for encryption and MAC change. However, the decryption should at least process every input, leading to the threat that the nonce could be kept constant while the ciphertext is changed, enabling DPA. For Isap, a DPA on the decryption part is prevented by executing the MAC first. In the case of the used sponge-based

suffix MAC, a change of the ciphertext (or any other processed data) always results in a change of the key K^*, thus also precluding DPA here:

- Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. "ISAP – Towards Side-Channel Secure Authenticated Encryption". In: *IACR Transactions on Symmetric Cryptology* 2017.1 (2017), pp. 80–105. URL: `http://tosc.iacr.org/index.php/ToSC/article/view/585`

After the publication of Isap, we discovered that Kocher, Jaffe, Jun, and Rohatgi describe an authenticated encryption scheme in a paper called "Introduction to differential power analysis" [73]. Here, they sketch an encrypt-then-MAC scheme, where the MAC is the combination of a key-tree [70] with a hash function. To be precise, the ciphertext is hashed and the result is processed by the key-tree. This results in a MAC that withstands DPA. In addition, they describe the idea to execute the MAC before decrypting the ciphertext to prevent any DPA attacks on the upcoming decryption. So contrary to what is claimed in Isap [j2], the credits for the first authenticated encryption scheme that prevents DPA for encryption and decryption belong to Kocher, Jaffe, Jun, and Rohatgi [73].

Nevertheless, we think that the sponge-based authenticated encryption scheme is a valuable contribution in itself. For instance, the MAC of Figure 4.4 demonstrates that just changing the function to absorb the key can result in a construction which is much harder to attack by DPA compared to using a plain xor. A very interesting topic for future research is to evaluate the side-channel and cryptographic security of the MAC shown in Figure 4.4 for different re-keying functions g that are more lightweight compared to the one used in Isap. Potential candidates are polynomial multiplications as proposed in [92], or the functions proposed in [45].

5 Fault Attacks

Like side-channel attacks discussed in Chapter 4, fault attacks belong to the category of implementation attacks. However, in contrast to side-channel attacks, an attacker does not exploit side-channels to retrieve information about the internal computations, or values during the execution of a cryptographic algorithm. Instead, the attacker disturbs the internal computation or values as sketched in Figure 5.1.

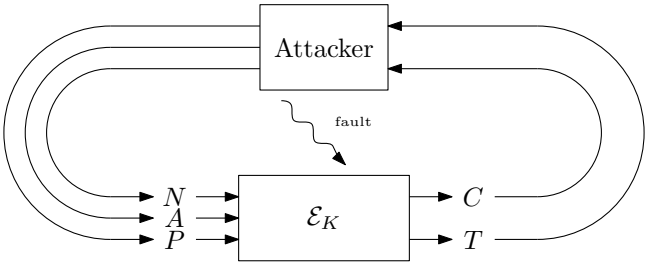


Figure 5.1: Concept of fault attacks.

The problem of faults induced by physical means in electronic circuits has already been known for decades. As mentioned in [8], already May and Woods [90] or Ziegler and Lanford [147] describe sources and effects of faults in the late 1970s. However, to the best of our knowledge, the first work pointing out that faults can be used to break cryptographic algorithms is the work of Boneh, DeMillo, and Lipton [33] published in 1996. They already mention the option that an attacker

possessing a device may induce faults and show fault attacks on implementations of various asymmetric schemes, such as RSA. Shortly after the publication of Boneh, DeMillo, and Lipton [33], Biham and Shamir [28] demonstrated the vulnerability of symmetric schemes by presenting several attacks on blockciphers, including differential fault analysis (DFA).

Differential fault analysis is closely related to differential cryptanalysis and usually works by exploiting pairs of correct and faulted computation results, e.g., pairs of correct and faulty ciphertexts. So, simply speaking, DFA requires an attacker to be able to trigger an encryption of the exact same input twice, where one execution is faulted. This condition is often not fulfilled for authenticated encryption schemes, since usually, each new encryption uses a new nonce. Additionally, the condition that unverified plaintext is not released often implicitly detects single faults. Thus, recent fault attacks on authenticated encryption usually require nonce reuse [126], release of unverified plaintext [125], or multiple faults per invocation [124].

To counteract the potential resulting impression that authenticated encryption schemes are secure against fault attacks, we demonstrate fault attacks on a wide range of AES-based primitives in [c2] (Part II of this thesis). In contrast to DFA, our attacks are based on statistical fault attacks (SFA) on AES [52]. Hence, our attacks target the underlying (tweakable) blockciphers and assume that an attacker is able to change the distribution of an intermediate value from uniform to some non-uniform distribution, while the input of the blockcipher is different and potentially unknown for each invocation. Thus, the requirement of encrypting the same input twice is circumvented, allowing the attack to also be applicable in a nonce-respecting scenario.

The remainder of this chapter is as follows. First, we introduce the working principles of differential fault analysis (DFA) in Section 5.1 and describe statistical fault attacks (SFA) in Section 5.2, which are the basis of our attacks. Section 5.3 gives a short summary on fault attacks besides DFA and SFA. Finally, we give a short overview on recent fault attacks on authenticated encryption schemes in Section 5.4.

5.1 Differential Fault Analysis

Differential Fault Analysis (DFA) has been introduced by Biham and Shamir [28] in 1997. The working mechanisms of DFA are closely related to differential cryptanalysis [27] and require typically the knowledge of correct and faulty data. Usually, the ability of an attacker to induce a fault towards the last rounds of a cipher can be exploited to inject a difference (compared to the correct intermediate value) in the intermediate state, which basically allows to apply the concepts of differential cryptanalysis to just a few rounds. As an example, Biham and Shamir [28] give an attack on DES, which we summarize in the following.

Let us assume a round-based hardware implementation of DES where an attacker can flip a single bit in a register, but has no influence on the position or the round where the fault occurs. Assuming that just one bit-flip happens per encryption, then — due to the Feistel structure — a bit-flip that happens before the last round can be easily identified. Such a fault is characterized by just a single difference in the half of the ciphertext that formed the input to the last round, while the other half of the state shows only differences due to the single bit entering the function. As argued by Biham and Shamir [28], knowing the input and output differences of one S-box reduces the set of possible keys to four 6-bit keys on average.

However, DFA is not limited to Feistel constructions and thus, can also be applied to SPN structures like AES. For instance, Piret and Quisquater [117] show how to attack AES using just 2 faulty ciphertexts. On a high level, the attack can be seen as truncated differential attack [68] on 2 rounds of AES. For simplicity, let us start with a simpler version to explain, which assumes that an attacker can inject a difference in one byte just before the last MixColumns operation of AES. Thus, as a result, the ciphertext has differences in 4 bytes. An attacker can now guess the 4 last round key bytes at the position of the differences and invert the last round followed by an inverse MixColumns operation. Key guesses that lead to a state having more then one byte difference can be discarded. As shown by Piret and Quisquater [117], two faulty cipheretxts are usually sufficient to determine the 4 key bytes uniquely. Placing the difference right before the second to last application of MixColumns leads to a single one-byte difference per column right before the last MixColumns, allowing to recover 4 times 4 bytes of the last round key in parallel.

5.2 Statistical Fault Attacks

In contrast to differential fault analysis (DFA), statistical fault attacks (SFA) [52] do not exploit the fact that a fault can be used to inject differences in a pair of intermediate states. Instead, it is assumed that faults can be used to influence the distribution of intermediate values across many encryptions. Thus, the requirements of encrypting two times the same inputs are replaced with just processing many different unknown inputs. The probably simplest version of this attack on AES is a stuck-at-zero fault right before the last application of the round key, which immediately reveals the key. Fuhr, Jaulmes, Lomné, and Thillard [52] consider the following three fault models on byte level assuming v is the targeted byte:

1. $v \leftarrow v$ AND 0
2. $v \leftarrow v$ AND 0 with probability 0.5
   $v \leftarrow v$ AND e with probability 0.5
   e picked randomly from uniform distribution

3. $v \leftarrow v$ AND e
   e picked randomly from uniform distribution

These three fault models are used to show attacks on AES targeting the 6^{th} to 9^{th} round. Since our attacks on nonce-based authenticated encryption schemes [c2] are based on statistical fault attacks [52] targeting the 8^{th} round, we will have a closer look at this attack.

In their 8^{th}-round attack, Fuhr, Jaulmes, Lomné, and Thillard [52] assume that an attacker can fault one byte right after the key addition of the 8^{th} round, which also marks the beginning of the 9^{th}, and change the distribution of this byte. Since the S-box of AES acts as a permutation on byte level and ShiftRows just influences the position, we end up with a situation where 1 byte right before the last MixColumns application is not uniformly distributed. The idea of the attack is to exploit this distribution under a 4-byte key guess of the last round key, assuming that a wrong key guess will result in a byte distribution which is closer to uniform compared to a right key guess. To evaluate the distribution, Fuhr, Jaulmes, Lomné, and Thillard [52] use the squared euclidean imbalance (SEI). They show that 6 faults are needed in the case of fault model 1, 14 faults for fault model 2, and 80 faults for fault model 3 to recover 4 bytes of the last round key with a probability of 99%.

5.3 Other Fault Attack Techniques

Clearly, DFA and SFA are not the only way in which cryptographic primitives can be attacked via faults. In this section, we want to provide a short summary of other prominent fault attacks apart from DFA and SFA.

The underlying principle of collision fault attacks [31] is somewhat similar to DFA. However, in collision fault attacks, related inputs are used to induce a difference, which is then tried to be cancelled after a few rounds by using a fault. If the induced differences can be cancelled successfully, the knowledge about the difference caused by the fault in combination with knowledge about the input difference gives insight on the used secret.

Another interesting category of fault attacks are attacks that just exploit the knowledge whether a fault has an effect on the computation or not. One prominent example of this category is the safe error attack [144]. In their attack, Yen and Joye [144] are able to retrieve information on the secret by observing if faulted intermediate values are used or not. To perform an ineffective fault attack [36], an attacker needs knowledge on the effect of a fault, e.g., that a fault can be used to set one byte to zero. Such a behaviour can be used to "probe" intermediate values of a cryptographic primitive, which can then in turn be exploited in attacks.

5.4 Fault Attacks on Authenticated Encryption

In this section, we discuss our work on fault attacks on authenticated encryption schemes [c2] in relation with recent attacks targeting the authenticated encryption scheme APE [125, 126] and PAEQ [124]. The fault attacks on APE require the cipher to be used in a so-called misuse scenario. Concretely, the first attack [126] on APE requires multiple encryptions using the same nonce, while the second attack [125] works only if unverified plaintext is released. The attack on PAEQ [124] is parallel independent work to the statistical fault attacks on nonce-based authenticated encryption schemes [c2] and attacks the encryption without making any assumptions on how the nonce is chosen. Simplified, PAEQ generates keystream to encrypt plaintext by using a permutation in a counter mode. Hence, a fault on the counter can be used to receive two permutation implementations having the same inputs, while a second fault can be used to get a pair of fault-free and faulty outputs, which can then be used to mount a DFA. So, are nonce-reuse, release of unverified plaintext, or the use of a counter mode necessary conditions for a fault attack and are other modes used in a nonce-respecting setting secure?

For understanding why — at the first glimpse — one might have the impression that authenticated encryption schemes withstand fault attacks, let us take a look at the encryption of COPA [6]. The value V in Figure 5.2 depends in an usually (for an attacker) unpredictable way on the nonce N and thus changes in a nonce-respecting use for every new invocation of the encryption. Therefore, also ciphertext and tag change even if the plaintext is kept constant. This leads to the fact that fault attacks like DFA that rely on that ability to collect faulty

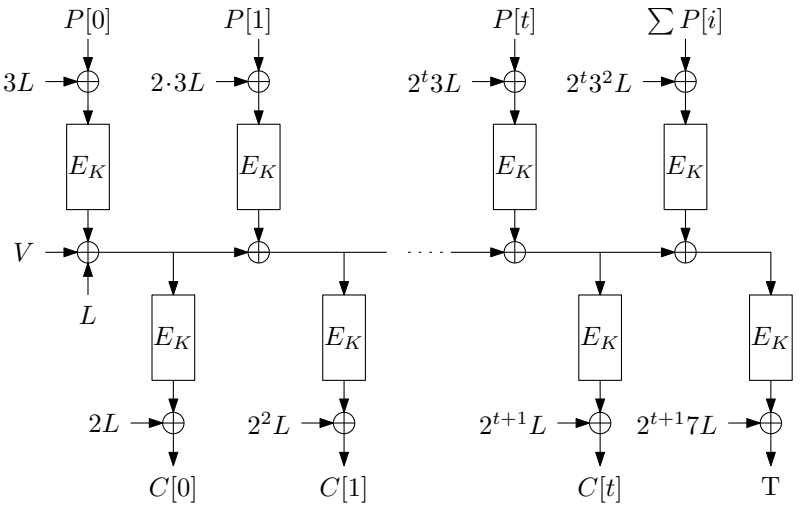


Figure 5.2: Encryption of AES-COPA [6]. ($L = E_K(0)$)

and fault-free ciphertext pairs cannot be applied in a straightforward manner.

This changes if we move away from fault models which assume that just bits can be flipped, or that a fault changes the value of intermediates randomly according to a uniform distribution. The assumption that faults can be used to change the distribution of uniformly distributed values to some non-uniform distribution as in SFA [52] allows for attacks on authenticated encryption without the need of having pairs of faulty and fault-free ciphertexts. In the following work, we demonstrate the applicability of statistical fault attacks on a wide range of authenticated encryption schemes that are based on AES:

- Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Victor Lomné, and Florian Mendel. "Statistical Fault Attacks on Nonce-Based Authenticated Encryption Schemes". In: *Advances in Cryptology – ASIACRYPT 2016*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. LNCS. 2016, pp. 369–395. URL: `https://doi.org/10.1007/978-3-662-53887-6_14`

Bibliography

[1] Ahmed Abdelkhalek, Mohamed Tolba, and Amr M. Youssef. "Impossible Differential Cryptanalysis of 8-round Kiasu-BC". Personal communication.

[2] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. "Ciphers for MPC and FHE". In: *Advances in Cryptology – EUROCRYPT 2015*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. LNCS. Springer, 2015, pp. 430–454. URL: https://doi.org/10.1007/978-3-662-46800-5_17.

[3] American Mathematical Society. "The Culture of Research and Scholarship in Mathematics: Joint Research and Its Publication". http://www.ams.org/profession/leaders/culture/CultureStatement04.pdf. 2004.

[4] Elena Andreeva, Begül Bilgin, Andrey Bogdanov, Atul Luykx, Florian Mendel, Bart Mennink, Nicky Mouha, Qingju Wang, and Kan Yasuda. "PRIMATEs v1.02". Submission to the CAESAR competition: http://competitions.cr.yp.to. 2014.

[5] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. "How to Securely Release Unverified Plaintext in Authenticated Encryption". In: *Advances in Cryptology – ASIACRYPT 2014*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. LNCS. Springer, 2014, pp. 105–125. URL: https://doi.org/10.1007/978-3-662-45611-8_6.

[6] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Elmar Tischhauser, and Kan Yasuda. "AES-COPA v.2". Submission to the CAESAR competition: http://competitions.cr.yp.to.

[7] Jean-Philippe Aumasson, Philipp Jovanovic, and Samuel Neves. "NORX V1". Submission to the CAESAR competition: http://competitions.cr.yp.to. 2014.

[8] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. "The Sorcerer's Apprentice Guide to Fault Attacks". In: *Proceedings of the IEEE* 94.2 (2006), pp. 370–382. URL: https://doi.org/10.1109/JPROC.2005.862424.

[9] Elaine B. Barker. "NIST SP-800-57 Pt1 Rev 4: Recommendation for Key Management, Part 1: General". Jan. 2016. URL: https://doi.org/10.6028/NIST.SP.800-57pt1r4.

[10] Guy Barwell, Daniel P. Martin, Elisabeth Oswald, and Martijn Stam. "Authenticated Encryption in the Face of Protocol and Side Channel Leakage". Cryptology ePrint Archive, Report 2017/68. http://eprint.iacr.org/2017/068. 2017.

[11] Sonia Belaïd, Jean-Sébastien Coron, Pierre-Alain Fouque, Benoît Gérard, Jean-Gabriel Kammerer, and Emmanuel Prouff. "Improved Side-Channel Analysis of Finite-Field Multiplication". In: *Cryptographic Hardware and Embedded Systems – CHES 2015*. Ed. by Tim Güneysu and Helena Handschuh. Vol. 9293. LNCS. Springer, 2015, pp. 395–415. URL: https://doi.org/10.1007/978-3-662-48324-4_20.

[12] Sonia Belaïd, Pierre-Alain Fouque, and Benoît Gérard. "Side-Channel Analysis of Multiplications in GF(2^{128}) - Application to AES-GCM". In: *Advances in Cryptology – ASIACRYPT 2014*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8874. LNCS. Springer, 2014, pp. 306–325. URL: https://doi.org/10.1007/978-3-662-45608-8_17.

[13] Mihir Bellare and Chanathip Namprempre. "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm". In: *Advances in Cryptology – ASIACRYPT 2000*. Ed. by Tatsuaki Okamoto. Vol. 1976. LNCS. Springer, 2000, pp. 531–545. URL: https://doi.org/10.1007/3-540-44448-3_41.

[14] Daniel J. Bernstein. "Cache-timing attacks on AES". https://cr.yp.to/papers.html#cachetiming. 2004.

[15] Francesco Berti, François Koeune, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. "Leakage-Resilient and Misuse-Resistant Authenticated Encryption". Cryptology ePrint Archive, Report 2016/996. http://eprint.iacr.org/2016/996. 2016.

[16] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. "The Keccak SHA-3 submission (Version 3.0)". http://keccak.noekeon.org/Keccak-submission-3.pdf. 2011.

[17] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Sponge functions". Ecrypt Hash Workshop 2007. May 2007.

[18] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "On the Indifferentiability of the Sponge Construction". In: *Advances in Cryptology – EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. LNCS. Springer, 2008, pp. 181–197. URL: https://doi.org/10.1007/978-3-540-78967-3_11.

[19] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Cryptographic sponge functions (Version 0.1)". http://sponge.noekeon.org. Jan. 2011.

[20] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "On alignment in Keccak". http://www.ecrypt.eu.org/hash2011/proceedings/hash2011_15.pdf. May 2011.

[21] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "On the security of the keyed sponge construction". Symmetric Key Encryption Workshop (SKEW). Feb. 2011.

[22] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications". In: *Selected Areas in Cryptography, SAC 2011*. Ed. by Ali Miri and Serge Vaudenay. Vol. 7118. LNCS. Springer, 2012, pp. 320–337. URL: https://doi.org/10.1007/978-3-642-28496-0_19.

[23] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Permutation-based encryption, authentication and authenticated encryption". Directions in Authenticated Ciphers. July 2012.

[24] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. "Keyak". Submission to the CAESAR competition: http://competitions.cr.yp.to. 2014.

[25] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. "Ketje v2". Submission to the CAESAR competition: http://competitions.cr.yp.to. 2016.

[26] Eli Biham. "How to decrypt or even substitute DES-encrypted messages in 2^{28} steps". In: *Inf. Process. Lett.* 84.3 (2002), pp. 117–124. URL: https://doi.org/10.1016/S0020-0190(02)00269-7.

[27] Eli Biham and Adi Shamir. "Differential Cryptanalysis of DES-like Cryptosystems". In: *Advances in Cryptology – CRYPTO '90*. Ed. by Alfred Menezes and Scott A. Vanstone. Vol. 537. LNCS. Springer, 1991, pp. 2–21. URL: https://doi.org/10.1007/3-540-38424-3_1.

[28] Eli Biham and Adi Shamir. "Differential Fault Analysis of Secret Key Cryptosystems". In: *Advances in Cryptology – CRYPTO '97*. Ed. by Burton S. Kaliski Jr. Vol. 1294. LNCS. Springer, 1997, pp. 513–525. URL: https://doi.org/10.1007/BFb0052259.

[29] Alex Biryukov and Ivica Nikolić. "Automatic Search for Related-Key Differential Characteristics in Byte-Oriented Block Ciphers: Application to AES, Camellia, Khazad and Others". In: *Advances in Cryptology – EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. LNCS. Springer, 2010, pp. 322–344. URL: https://doi.org/10.1007/978-3-642-13190-5_17.

[30] Alex Biryukov, Vesselin Velichkov, and Yann Le Corre. "Automatic Search for the Best Trails in ARX: Application to Block Cipher Speck". In: *Fast Software Encryption, FSE 2016*. Ed. by Thomas Peyrin. Vol. 9783. LNCS. Springer, 2016, pp. 289–310. URL: https://doi.org/10.1007/978-3-662-52993-5_15.

[31] Johannes Blömer and Volker Krummel. "Fault Based Collision Attacks on AES". In: *Fault Diagnosis and Tolerance in Cryptography, FDTC 2006*. Ed. by Luca Breveglieri, Israel Koren, David Naccache, and Jean-Pierre Seifert. Vol. 4236. LNCS. Springer, 2006, pp. 106–120. URL: https://doi.org/10.1007/11889700_11.

[32] Andrey Bogdanov and Vincent Rijmen. “Linear hulls with correlation zero and linear cryptanalysis of block ciphers”. In: *Des. Codes Cryptography* 70.3 (2014), pp. 369–383. URL: https://doi.org/10.1007/s10623-012-9697-z.

[33] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. “On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract)”. In: *Advances in Cryptology – EUROCRYPT ’97*. Ed. by Walter Fumy. Vol. 1233. LNCS. Springer, 1997, pp. 37–51. URL: https://doi.org/10.1007/3-540-69053-0_4.

[34] CAESAR committee. “CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness”. 2014. URL: http://competitions.cr.yp.to/caesar.html.

[35] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. “Towards Sound Approaches to Counteract Power-Analysis Attacks”. In: *Advances in Cryptology – CRYPTO ’99*. Ed. by Michael Wiener. Vol. 1666. LNCS. Springer, 1999, pp. 398–412. URL: https://doi.org/10.1007/3-540-48405-1_26.

[36] Christophe Clavier. “Secret External Encodings Do Not Prevent Transient Fault Analysis”. In: *Cryptographic Hardware and Embedded Systems – CHES 2007*. Ed. by Pascal Paillier and Ingrid Verbauwhede. Vol. 4727. LNCS. Springer, 2007, pp. 181–194. URL: https://doi.org/10.1007/978-3-540-74735-2_13.

[37] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. “The Block Cipher Square”. In: *Fast Software Encryption, FSE ’97*. Ed. by Eli Biham. Vol. 1267. LNCS. Springer, 1997, pp. 149–165. URL: https://doi.org/10.1007/BFb0052343.

[38] Joan Daemen and Vincent Rijmen. “The Block Cipher Rijndael”. In: *Smart Card Research and Applications, CARDIS ’98*. Ed. by Jean-Jacques Quisquater and Bruce Schneier. Vol. 1820. LNCS. Springer, 2000, pp. 277–284. URL: https://doi.org/10.1007/10721064_26.

[39] Joan Daemen and Vincent Rijmen. “The Design of Rijndael: AES – The Advanced Encryption Standard”. Information Security and Cryptography. Springer, 2002. ISBN: 3-540-42580-2.

[40] Joan Daemen and Gilles Van Assche. “Differential Propagation Analysis of Keccak”. In: *Fast Software Encryption, FSE 2012*. Ed. by Anne Canteaut. Vol. 7549. LNCS. Springer, 2012, pp. 422–441. URL: https://doi.org/10.1007/978-3-642-34047-5_24.

[41] Christophe De Cannière and Christian Rechberger. “Finding SHA-1 Characteristics: General Results and Applications”. In: *Advances in Cryptology – ASIACRYPT 2006*. Ed. by Xuejia Lai and Kefei Chen. Vol. 4284. LNCS. Springer, 2006, pp. 1–20. URL: https://doi.org/10.1007/11935230_1.

[42] T. Dierks and E. Rescorla. “The Transport Layer Security (TLS) Protocol Version 1.2”. RFC 5246. RFC. 2008.

[43] Itai Dinur, Pawel Morawiecki, Josef Pieprzyk, Marian Srebrny, and Michal Straus. "Cube Attacks and Cube-Attack-Like Cryptanalysis on the Round-Reduced Keccak Sponge Function". In: *Advances in Cryptology – EUROCRYPT 2015*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. LNCS. Springer, 2015, pp. 733–761. URL: https://doi.org/10.1007/978-3-662-46800-5_28.

[44] Itai Dinur and Adi Shamir. "Cube Attacks on Tweakable Black Box Polynomials". In: *Advances in Cryptology – EUROCRYPT 2009*. Ed. by Antoine Joux. Vol. 5479. LNCS. Springer, 2009, pp. 278–299. URL: https://doi.org/10.1007/978-3-642-01001-9_16.

[45] 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*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9815. LNCS. Springer, 2016, pp. 272–301. URL: https://doi.org/10.1007/978-3-662-53008-5_10.

[46] ECRYPT. "The eSTREAM Project". 2004. URL: http://www.ecrypt.eu.org/stream/project.html.

[47] Maria Eichlseder, Florian Mendel, and Martin Schläffer. "Branching Heuristics in Differential Collision Search with Applications to SHA-512". In: *Fast Software Encryption, FSE 2014*. Ed. by Carlos Cid and Christian Rechberger. Vol. 8540. LNCS. Springer, 2015, pp. 473–488. URL: https://doi.org/10.1007/978-3-662-46706-0_24.

[48] Horst Feistel. "Cryptography and Computer Privacy". In: *Scientific American* 228 (May 1973), pp. 15–23.

[49] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, and Doug Whiting. "Improved Cryptanalysis of Rijndael". In: *Fast Software Encryption, FSE 2000*. Ed. by Bruce Schneier. Vol. 1978. LNCS. Springer, 2001, pp. 213–230. URL: https://doi.org/10.1007/3-540-44706-7_15.

[50] Pierre-Alain Fouque, Gaëtan Leurent, and Phong Q. Nguyen. "Automatic Search of Differential Path in MD4". Cryptology ePrint Archive, Report 2007/206. http://eprint.iacr.org/2007/206. 2007.

[51] Kai Fu, Meiqin Wang, Yinghua Guo, Siwei Sun, and Lei Hu. "MILP-Based Automatic Search Algorithms for Differential and Linear Trails for Speck". In: *Fast Software Encryption, FSE 2016*. Ed. by Thomas Peyrin. Vol. 9783. LNCS. Springer, 2016, pp. 268–288. URL: https://doi.org/10.1007/978-3-662-52993-5_14.

[52] Thomas Fuhr, Eliane Jaulmes, Victor Lomné, and Adrian Thillard. "Fault Attacks on AES with Faulty Ciphertexts Only". In: *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*. Ed. by Wieland Fischer and Jörn-Marc Schmidt. IEEE Computer Society, 2013, pp. 108–118. URL: https://doi.org/10.1109/FDTC.2013.18.

[53] Peter Gazi, Krzysztof Pietrzak, and Stefano Tessaro. “The Exact PRF Security of Truncation: Tight Bounds for Keyed Sponges and Truncated CBC”. In: *Advances in Cryptology – CRYPTO 2015*. Ed. by Rosario Gennaro and Matthew Robshaw. Vol. 9215. LNCS. Springer, 2015, pp. 368–387. URL: https://doi.org/10.1007/978-3-662-47989-6_18.

[54] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. “How to construct random functions”. In: *J. ACM* 33.4 (1986), pp. 792–807. URL: http://doi.acm.org/10.1145/6490.6503.

[55] Louis Goubin and Jacques Patarin. “DES and Differential Power Analysis (The "Duplication" Method)”. In: *Cryptographic Hardware and Embedded Systems, CHES'99*. Ed. by Çetin Kaya Koç and Christof Paar. Vol. 1717. LNCS. Springer, 1999, pp. 158–172. URL: https://doi.org/10.1007/3-540-48059-5_15.

[56] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, Kerem Varici, Anthony Journault, François Durvaux, Lubos Gaspar, and Stéphanie Kerckhoff. “SCREAM (Version 3)”. Submission to the CAESAR competition: http://competitions.cr.yp.to. 2015.

[57] Shay Gueron and Nicky Mouha. “Simpira v2: A Family of Efficient Permutations Using the AES Round Function”. In: *Advances in Cryptology – ASIACRYPT 2016*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. LNCS. 2016, pp. 95–125. URL: https://doi.org/10.1007/978-3-662-53887-6_4.

[58] Senyang Huang, Xiaoyun Wang, Guangwu Xu, Meiqin Wang, and Jingyuan Zhao. “Conditional Cube Attack on Reduced-Round Keccak Sponge Function”. In: *Advances in Cryptology – EUROCRYPT 2017*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10211. LNCS. 2017, pp. 259–288. URL: https://doi.org/10.1007/978-3-319-56614-6_9.

[59] Yuval Ishai, Amit Sahai, and David Wagner. “Private Circuits: Securing Hardware against Probing Attacks”. In: *Advances in Cryptology – CRYPTO 2003*. Ed. by Dan Boneh. Vol. 2729. LNCS. Springer, 2003, pp. 463–481. URL: https://doi.org/10.1007/978-3-540-45146-4_27.

[60] Jérémy Jean, Ivica Nikolić, and Thomas Peyrin. “KIASU”. Submission to the CAESAR competition: http://competitions.cr.yp.to. 2014.

[61] Jérémy Jean, Ivica Nikolić, and Thomas Peyrin. “Tweaks and Keys for Block Ciphers: The TWEAKEY Framework”. In: *Advances in Cryptology – ASIACRYPT 2014*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8874. LNCS. Springer, 2014, pp. 274–288. URL: https://doi.org/10.1007/978-3-662-45608-8_15.

[62] Jérémy Jean, Ivica Nikolić, Thomas Peyrin, and Yannick Seurin. “Deoxys v1.41”. Submission to the CAESAR competition: http://competitions.cr.yp.to. 2016.

[63] Philipp Jovanovic, Atul Luykx, and Bart Mennink. "Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes". In: *Advances in Cryptology – ASIACRYPT 2014*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. LNCS. Springer, 2014, pp. 85–104. URL: https://doi.org/10.1007/978-3-662-45611-8_5.

[64] Charanjit S. Jutla. "Encryption Modes with Almost Free Message Integrity". In: *Advances in Cryptology – EUROCRYPT 2001*. Ed. by Birgit Pfitzmann. Vol. 2045. LNCS. Springer, 2001, pp. 529–544. URL: https://doi.org/10.1007/3-540-44987-6_32.

[65] Charanjit S. Jutla. "Encryption Modes with Almost Free Message Integrity". In: *J. Cryptology* 21.4 (2008), pp. 547–578. URL: https://doi.org/10.1007/s00145-008-9024-z.

[66] Emilia Käsper and Peter Schwabe. "Faster and Timing-Attack Resistant AES-GCM". In: *Cryptographic Hardware and Embedded Systems – CHES 2009*. Ed. by Christophe Clavier and Kris Gaj. Vol. 5747. LNCS. Springer, 2009, pp. 1–17. URL: https://doi.org/10.1007/978-3-642-04138-9_1.

[67] S. Kent. "IP Encapsulating Security Payload (ESP)". RFC 4303. RFC. 2005.

[68] Lars R. Knudsen. "Truncated and Higher Order Differentials". In: *Fast Software Encryption, FSE 1994*. Ed. by Bart Preneel. Vol. 1008. LNCS. Springer, 1995, pp. 196–211. URL: https://doi.org/10.1007/3-540-60590-8_16.

[69] Lars R. Knudsen and David A. Wagner. "Integral Cryptanalysis". In: *Fast Software Encryption, FSE 2002*. Ed. by Joan Daemen and Vincent Rijmen. Vol. 2365. LNCS. Springer, 2002, pp. 112–127. URL: https://doi.org/10.1007/3-540-45661-9_9.

[70] Paul Kocher. "Design and Validation Strategies for Obtaining Assurance in Countermeasures to Power Analysis and Related Attacks". NIST Physical Security Testing Workshop (http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-3/physec/papers/physecpaper09.pdf). Dec. 2005.

[71] Paul C. Kocher. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". In: *Advances in Cryptology – CRYPTO '96*. Ed. by Neal Koblitz. Vol. 1109. LNCS. Springer, 1996, pp. 104–113. URL: https://doi.org/10.1007/3-540-68697-5_9.

[72] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. "Differential Power Analysis". In: *Advances in Cryptology – CRYPTO '99*. Ed. by Michael Wiener. Vol. 1666. LNCS. Springer, 1999, pp. 388–397. URL: https://doi.org/10.1007/3-540-48405-1_25.

[73] Paul C. Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. "Introduction to differential power analysis". In: *J. Cryptographic Engineering* 1.1 (2011), pp. 5–27. URL: https://doi.org/10.1007/s13389-011-0006-y.

[74] Stefan Kölbl, Gregor Leander, and Tyge Tiessen. "Observations on the SIMON Block Cipher Family". In: *Advances in Cryptology – CRYPTO 2015*. Ed. by Rosario Gennaro and Matthew Robshaw. Vol. 9215. LNCS. Springer, 2015, pp. 161–185. URL: https://doi.org/10.1007/978-3-662-47989-6_8.

[75] Hugo Krawczyk. "The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?)" In: *Advances in Cryptology – CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. LNCS. Springer, 2001, pp. 310–331. URL: https://doi.org/10.1007/3-540-44647-8_19.

[76] Ted Krovetz and Phillip Rogaway. "The Software Performance of Authenticated-Encryption Modes". In: *Fast Software Encryption, FSE 2011*. Ed. by Antoine Joux. Vol. 6733. LNCS. Springer, 2011, pp. 306–327. URL: https://doi.org/10.1007/978-3-642-21702-9_18.

[77] Xuejia Lai. "Higher Order Derivatives and Differential Cryptanalysis". In: *Communications and Cryptography: Two Sides of One Tapestry*. Ed. by Richard E. Blahut, Daniel J. Costello Jr., Ueli Maurer, and Thomas Mittelholzer. Vol. 276. International Series in Engineering and Computer Science. Kluwer Academic Publishers, 1994, pp. 227–233.

[78] Xuejia Lai, James L. Massey, and Sean Murphy. "Markov Ciphers and Differential Cryptanalysis". In: *Advances in Cryptology – EUROCRYPT '91*. Ed. by Donald W. Davies. Vol. 547. LNCS. Springer, 1991, pp. 17–38. URL: https://doi.org/10.1007/3-540-46416-6_2.

[79] Susan K. Langford and Martin E. Hellman. "Differential-Linear Cryptanalysis". In: *Advances in Cryptology – CRYPTO '94*. Ed. by Yvo Desmedt. Vol. 839. LNCS. Springer, 1994, pp. 17–25. URL: https://doi.org/10.1007/3-540-48658-5_3.

[80] Gaëtan Leurent. "Construction of Differential Characteristics in ARX Designs Application to Skein". In: *Advances in Cryptology – CRYPTO 2013*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. LNCS. Springer, 2013, pp. 241–258. URL: https://doi.org/10.1007/978-3-642-40041-4_14.

[81] Zheng Li, Xiaoyang Dong, and Xiaoyun Wang. "Conditional Cube Attack on Round-Reduced ASCON". In: *IACR Transactions on Symmetric Cryptology* 2017.1 (2017), pp. 175–202. URL: http://tosc.iacr.org/index.php/ToSC/article/view/590.

[82] Moses Liskov, Ronald L. Rivest, and David Wagner. "Tweakable Block Ciphers". In: *Advances in Cryptology – CRYPTO 2002*. Ed. by Moti Yung. Vol. 2442. LNCS. Springer, 2002, pp. 31–46. URL: https://doi.org/10.1007/3-540-45708-9_3.

[83] Yunwen Liu, Qingju Wang, and Vincent Rijmen. "Automatic Search of Linear Trails in ARX with Applications to SPECK and Chaskey". In: *Applied Cryptography and Network Security, ACNS 2016*. Ed. by Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider. Vol. 9696. LNCS.

Springer, 2016, pp. 485–499. URL: https://doi.org/10.1007/978-3-319-39555-5_26.

[84] Stefan Mangard. "A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion". In: *Information Security and Cryptology – ICISC 2002*. Ed. by Pil Joong Lee and Chae Hoon Lim. Vol. 2587. LNCS. Springer, 2003, pp. 343–358. URL: https://doi.org/10.1007/3-540-36552-4_24.

[85] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. "Power Analysis Attacks. Revealing the Secrets of Smart Cards". Springer, 2007. ISBN: 978-0-387-30857-9.

[86] Stefan Mangard, Elisabeth Oswald, and François-Xavier Standaert. "One for all - all for one: unifying standard differential power analysis attacks". In: *IET Information Security* 5.2 (2011), pp. 100–110. URL: https://doi.org/10.1049/iet-ifs.2010.0096.

[87] Stefan Mangard, Thomas Popp, and Berndt M. Gammel. "Side-Channel Leakage of Masked CMOS Gates". In: *Topics in Cryptology – CT-RSA 2005*. Ed. by Alfred Menezes. Vol. 3376. LNCS. Springer, 2005, pp. 351–365. URL: https://doi.org/10.1007/978-3-540-30574-3_24.

[88] Mitsuru Matsui. "Linear Cryptanalysis Method for DES Cipher". In: *Advances in Cryptology – EUROCRYPT '93*. Ed. by Tor Helleseth. Vol. 765. LNCS. Springer, 1994, pp. 386–397. URL: https://doi.org/10.1007/3-540-48285-7_33.

[89] Mitsuru Matsui. "On Correlation Between the Order of S-boxes and the Strength of DES". In: *Advances in Cryptology – EUROCRYPT '94*. Ed. by Alfredo De Santis. Vol. 950. LNCS. Springer, 1995, pp. 366–375. URL: https://doi.org/10.1007/BFb0053451.

[90] Timothy C. May and Murray H. Woods. "A New Physical Mechanism for Soft Errors in Dynamic Memories". In: *16th International Reliability Physics Symposium*. Apr. 1978, pp. 33–40.

[91] 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: *Smart Card Research and Advanced Applications, CARDIS 2011*. Ed. by Emmanuel Prouff. Vol. 7079. LNCS. Springer, 2011, pp. 115–132. URL: https://doi.org/10.1007/978-3-642-27257-8_8.

[92] 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*. Ed. by Daniel J. Bernstein and Tanja Lange. Vol. 6055. LNCS. Springer, 2010, pp. 279–296. URL: https://doi.org/10.1007/978-3-642-12678-9_17.

[93] Silvia Mella, Joan Daemen, and Gilles Van Assche. "New techniques for trail bounds and application to differential trails in Keccak". In: *IACR Transactions on Symmetric Cryptology* 2017.1 (2017), pp. 329–357. URL: http://tosc.iacr.org/index.php/ToSC/article/view/597.

[94] Florian Mendel, Bart Mennink, Vincent Rijmen, and Elmar Tischhauser. "A Simple Key-Recovery Attack on McOE-X". In: *Cryptology and Network Security, CANS 2012*. Ed. by Josef Pieprzyk, Ahmad-Reza Sadeghi, and Mark Manulis. Vol. 7712. Springer, 2012, pp. 23–31. URL: https://doi.org/10.1007/978-3-642-35404-5_3.

[95] Florian Mendel, Tomislav Nad, and Martin Schläffer. "Finding SHA-2 Characteristics: Searching through a Minefield of Contradictions". In: *Advances in Cryptology – ASIACRYPT 2011*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. LNCS. Springer, 2011, pp. 288–307. URL: https://doi.org/10.1007/978-3-642-25385-0_16.

[96] Florian Mendel, Tomislav Nad, and Martin Schläffer. "Finding Collisions for Round-Reduced SM3". In: *Topics in Cryptology – CT-RSA 2013*. Ed. by Ed Dawson. Vol. 7779. LNCS. Springer, 2013, pp. 174–188. URL: https://doi.org/10.1007/978-3-642-36095-4_12.

[97] Florian Mendel, Tomislav Nad, and Martin Schläffer. "Improving Local Collisions: New Attacks on Reduced SHA-256". In: *Advances in Cryptology – EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. LNCS. Springer, 2013, pp. 262–278. URL: https://doi.org/10.1007/978-3-642-38348-9_16.

[98] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. "Handbook of Applied Cryptography". CRC Press, 1996. ISBN: 0-8493-8523-7.

[99] Bart Mennink, Reza Reyhanitabar, and Damian Vizár. "Security of Full-State Keyed Sponge and Duplex: Applications to Authenticated Encryption". In: *Advances in Cryptology – ASIACRYPT 2015*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9453. LNCS. Springer, 2015, pp. 465–489. URL: https://doi.org/10.1007/978-3-662-48800-3_19.

[100] Amir Moradi and Tobias Schneider. "Improved Side-Channel Analysis Attacks on Xilinx Bitstream Encryption of 5, 6, and 7 Series". In: *Constructive Side-Channel Analysis and Secure Design, COSADE 2016*. Ed. by François-Xavier Standaert and Elisabeth Oswald. Vol. 9689. LNCS. Springer, 2016, pp. 71–87. URL: https://doi.org/10.1007/978-3-319-43283-0_5.

[101] Pawel Morawiecki, Kris Gaj, Ekawat Homsirikamol, Krystian Matusiewicz, Josef Pieprzyk, Marcin Rogawski, Marian Srebrny, and Marcin Wójcik. "ICEPOLE v1". Submission to the CAESAR competition: http://competitions.cr.yp.to. 2014.

[102] Nicky Mouha and Bart Preneel. "A Proof that the ARX Cipher Salsa20 is Secure against Differential Cryptanalysis". Cryptology ePrint Archive, Report 2013/328. http://eprint.iacr.org/2013/328. 2013.

[103] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. "Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming". In: *Information Security and Cryptology, Inscrypt 2011*. Ed. by Chuankun Wu, Moti Yung, and Dongdai Lin. Vol. 7537. LNCS. Springer, 2012, pp. 57–76. URL: https://doi.org/10.1007/978-3-642-34704-7_5.

[104] Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. "Reconsidering Generic Composition". In: *Advances in Cryptology – EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. LNCS. Springer, 2014, pp. 257–274. URL: https://doi.org/10.1007/978-3-642-55220-5_15.

[105] National Bureau of Standards. "FIPS PUB 46: Data Encryption Standard". Jan. 1977.

[106] National Institute of Standards and Technology. "ANNOUNCING REQUEST FOR CANDIDATE ALGORITHM NOMINATIONS FOR THE ADVANCED ENCRYPTION STANDARD (AES)". http://csrc.nist.gov/archive/aes/pre-round1/aes_9709.htm. Sept. 1997.

[107] National Institute of Standards and Technology. "FIPS PUB 197: Specification for the ADVANCED ENCRYPTION STANDARD (AES)". Federal Information Processing Standards Publication 197. Nov. 2001. URL: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf.

[108] National Institute of Standards and Technology. "Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family". http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf. Nov. 2007.

[109] 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. Aug. 2015. URL: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf.

[110] Michael Neve. "Cache-based Vulnerabilities and SPAM analysis". PhD thesis. Université catholique de Louvain, June 2006.

[111] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. "Threshold Implementations Against Side-Channel Attacks and Glitches". In: *Information and Communications Security, ICICS 2006*. Ed. by Peng Ning, Sihan Qing, and Ninghui Li. Vol. 4307. LNCS. Springer, 2006, pp. 529–545. URL: https://doi.org/10.1007/11935308_38.

[112] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. "Secure Hardware Implementation of Non-linear Functions in the Presence of Glitches". In: *Information Security and Cryptology – ICISC 2008*. Ed. by Pil Joong Lee and Jung Hee Cheon. Vol. 5461. LNCS. Springer, 2009, pp. 218–234. URL: https://doi.org/10.1007/978-3-642-00730-9_14.

[113] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. “Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches”. In: *J. Cryptology* 24.2 (2011), pp. 292–321. URL: https://doi.org/10.1007/s00145-010-9085-7.

[114] Dag Arne Osvik, Adi Shamir, and Eran Tromer. “Cache Attacks and Countermeasures: The Case of AES”. In: *Topics in Cryptology – CT-RSA 2006*. Ed. by David Pointcheval. Vol. 3860. LNCS. Springer, 2006, pp. 1–20. URL: https://doi.org/10.1007/11605805_1.

[115] Peter Pessl and Stefan Mangard. “Enhancing Side-Channel Analysis of Binary-Field Multiplication with Bit Reliability”. In: *Topics in Cryptology – CT-RSA 2016*. Ed. by Kazue Sako. Vol. 9610. LNCS. Springer, 2016, pp. 255–270. URL: https://doi.org/10.1007/978-3-319-29485-8_15.

[116] Krzysztof Pietrzak. “Cryptography from Learning Parity with Noise”. In: *SOFSEM 2012: Theory and Practice of Computer Science*. Ed. by Mária Bieliková, Gerhard Friedrich, Georg Gottlob, Stefan Katzenbeisser, and György Turán. Vol. 7147. LNCS. Springer, 2012, pp. 99–114. URL: https://doi.org/10.1007/978-3-642-27660-6_9.

[117] Gilles Piret and Jean-Jacques Quisquater. “A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD”. In: *Cryptographic Hardware and Embedded Systems – CHES 2003*. Ed. by Colin D. Walter, Çetin Kaya Koç, and Christof Paar. Vol. 2779. LNCS. Springer, 2003, pp. 77–88. URL: https://doi.org/10.1007/978-3-540-45238-6_7.

[118] Jean-Jacques Quisquater and David Samyde. “ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards”. In: *Smart Card Programming and Security, E-smart 2001*. Ed. by Isabelle Attali and Thomas P. Jensen. Vol. 2140. LNCS. Springer, 2001, pp. 200–210. URL: https://doi.org/10.1007/3-540-45418-7_17.

[119] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. “Consolidating Masking Schemes”. In: *Advances in Cryptology – CRYPTO 2015*. Ed. by Rosario Gennaro and Matthew Robshaw. Vol. 9215. LNCS. Springer, 2015, pp. 764–783. URL: https://doi.org/10.1007/978-3-662-47989-6_37.

[120] Vincent Rijmen, Joan Daemen, Bart Preneel, Antoon Bosselaers, and Erik De Win. “The Cipher SHARK”. In: *Fast Software Encryption, FSE 1996*. Ed. by Dieter Gollmann. Vol. 1039. LNCS. Springer, 1996, pp. 99–111. URL: https://doi.org/10.1007/3-540-60865-6_47.

[121] Phillip Rogaway. “Authenticated-encryption with associated-data”. In: *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002*. Ed. by Vijayalakshmi Atluri. ACM, 2002, pp. 98–107. URL: http://doi.acm.org/10.1145/586110.586125.

[122] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. "OCB: a block-cipher mode of operation for efficient authenticated encryption". In: *Proceedings of the 8th ACM Conference on Computer and Communications Security, CCS 2001*. Ed. by Michael K. Reiter and Pierangela Samarati. ACM, 2001, pp. 196–205. URL: http://doi.acm.org/10.1145/501983.502011.

[123] Eyal Ronen, Colin O'Flynn, Adi Shamir, and Achi-Or Weingarten. "IoT Goes Nuclear: Creating a ZigBee Chain Reaction". In: *2017 IEEE Symposium on Security and Privacy, SP 2017*. IEEE Computer Society, 2017, pp. 195–212. URL: https://doi.org/10.1109/SP.2017.14.

[124] Dhiman Saha and Dipanwita Roy Chowdhury. "EnCounter: On Breaking the Nonce Barrier in Differential Fault Analysis with a Case-Study on PAEQ". In: *Cryptographic Hardware and Embedded Systems – CHES 2016*. Ed. by Benedikt Gierlichs and Axel Y. Poschmann. Vol. 9813. LNCS. Springer, 2016, pp. 581–601. URL: https://doi.org/10.1007/978-3-662-53140-2_28.

[125] Dhiman Saha and Dipanwita Roy Chowdhury. "Scope: On the Side Channel Vulnerability of Releasing Unverified Plaintexts". In: *Selected Areas in Cryptography – SAC 2015*. Ed. by Orr Dunkelman and Liam Keliher. Vol. 9566. LNCS. Springer, 2016, pp. 417–438. URL: https://doi.org/10.1007/978-3-319-31301-6_24.

[126] Dhiman Saha, Sukhendu Kuila, and Dipanwita Roy Chowdhury. "EscApe: Diagonal Fault Analysis of APE". In: *Progress in Cryptology – INDOCRYPT 2014*. Ed. by Willi Meier and Debdeep Mukhopadhyay. Vol. 8885. LNCS. Springer, 2014, pp. 197–216. URL: https://doi.org/10.1007/978-3-319-13039-2_12.

[127] Yu Sasaki and Kan Yasuda. "How to Incorporate Associated Data in Sponge-Based Authenticated Encryption". In: *Topics in Cryptology – CT-RSA 2015*. Ed. by Kaisa Nyberg. Vol. 9048. LNCS. Springer, 2015, pp. 353–370. URL: https://doi.org/10.1007/978-3-319-16715-2_19.

[128] Martin Schläffer and Elisabeth Oswald. "Searching for Differential Paths in MD4". In: *Fast Software Encryption, FSE 2006*. Ed. by Matthew J. B. Robshaw. Vol. 4047. LNCS. Springer, 2006, pp. 242–261. URL: https://doi.org/10.1007/11799313_16.

[129] Claude E. Shannon. "Communication Theory of Secrecy Systems". In: *Bell System Technical Journal* 28 (1949), pp. 656–715.

[130] François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. "Leakage Resilient Cryptography in Practice". Cryptology ePrint Archive, Report 2009/341. http://eprint.iacr.org/2009/341. 2009.

[131] Marc Stevens. "Fast Collision Attack on MD5". Cryptology ePrint Archive, Report 2006/104. http://eprint.iacr.org/2006/104. 2006.

[132] Ling Sun, Wei Wang, and Meiqin Wang. "Automatic Search of Bit-Based Division Property for ARX Ciphers and Word-Based Division Property". Cryptology ePrint Archive, Report 2017/860. http://eprint.iacr.org/2017/860. 2017.

[133] Siwei Sun, Lei Hu, Ling Song, Yonghong Xie, and Peng Wang. "Automatic Security Evaluation of Block Ciphers with S-bP Structures Against Related-Key Differential Attacks". In: *Information Security and Cryptology, Inscrypt 2013*. Ed. by Dongdai Lin, Shouhuai Xu, and Moti Yung. Vol. 8567. LNCS. Springer, 2014, pp. 39–51. URL: https://doi.org/10.1007/978-3-319-12087-4_3.

[134] Siwei Sun, Lei Hu, Meiqin Wang, Peng Wang, Kexin Qiao, Xiaoshuang Ma, Danping Shi, Ling Song, and Kai Fu. "Constructing Mixed-integer Programming Models whose Feasible Region is Exactly the Set of All Valid Differential Characteristics of SIMON". Cryptology ePrint Archive, Report 2015/122. http://eprint.iacr.org/2015/122. 2015.

[135] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. "Automatic Security Evaluation and (Related-key) Differential Characteristic Search: Application to SIMON, PRESENT, LBlock, DES(L) and Other Bit-Oriented Block Ciphers". In: *Advances in Cryptology – ASIACRYPT 2014*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. LNCS. Springer, 2014, pp. 158–178. URL: https://doi.org/10.1007/978-3-662-45611-8_9.

[136] Mostafa M. I. Taha, Arash Reyhani-Masoleh, and Patrick Schaumont. "Keymill: Side-Channel Resilient Key Generator". Cryptology ePrint Archive, Report 2016/710. http://eprint.iacr.org/2016/710. 2016.

[137] Yosuke Todo. "Integral Cryptanalysis on Full MISTY1". In: *Advances in Cryptology – CRYPTO 2015*. Ed. by Rosario Gennaro and Matthew Robshaw. Vol. 9215. LNCS. Springer, 2015, pp. 413–432. URL: https://doi.org/10.1007/978-3-662-47989-6_20.

[138] Yosuke Todo. "Structural Evaluation by Generalized Integral Property". In: *Advances in Cryptology – EUROCRYPT 2015*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. LNCS. Springer, 2015, pp. 287–314. URL: https://doi.org/10.1007/978-3-662-46800-5_12.

[139] Mohamed Tolba, Ahmed Abdelkhalek, and Amr M. Youssef. "A Meet in the Middle Attack on Reduced Round Kiasu-BC". In: *IEICE Transactions* 99-A.10 (2016), pp. 1888–1890. URL: http://search.ieice.org/bin/summary.php?id=e99-a_10_1888.

[140] Yukiyasu Tsunoo, Teruo Saito, Tomoyasu Suzaki, Maki Shigeri, and Hiroshi Miyauchi. "Cryptanalysis of DES Implemented on Computers with Cache". In: *Cryptographic Hardware and Embedded Systems – CHES 2003*. Ed. by Colin D. Walter, Çetin Kaya Koç, and Christof Paar. Vol. 2779. LNCS. Springer, 2003, pp. 62–76. URL: https://doi.org/10.1007/978-3-540-45238-6_6.

[141] Michael Vielhaber. “Breaking ONE.FIVIUM by AIDA an Algebraic IV Differential Attack”. Cryptology ePrint Archive, Report 2007/413. http://eprint.iacr.org/2007/413. 2007.

[142] David Wagner. “The Boomerang Attack”. In: *Fast Software Encryption, FSE ’99*. Ed. by Lars R. Knudsen. Vol. 1636. LNCS. Springer, 1999, pp. 156–170. URL: https://doi.org/10.1007/3-540-48519-8_12.

[143] Xiaoyun Wang and Hongbo Yu. “How to Break MD5 and Other Hash Functions”. In: *Advances in Cryptology – EUROCRYPT 2005*. Ed. by Ronald Cramer. Vol. 3494. LNCS. Springer, 2005, pp. 19–35. URL: https://doi.org/10.1007/11426639_2.

[144] Sung-Ming Yen and Marc Joye. “Checking Before Output May Not Be Enough Against Fault-Based Cryptanalysis”. In: *IEEE Trans. Computers* 49.9 (2000), pp. 967–970. URL: https://doi.org/10.1109/12.869328.

[145] T. Ylonen and C. Lonvick. “The Secure Shell (SSH) Transport Layer Protocol”. RFC 4253. RFC. 2006.

[146] Wenying Zhang and Vincent Rijmen. “Division Cryptanalysis of Block Ciphers with a Binary Diffusion Layer”. Cryptology ePrint Archive, Report 2017/188. http://eprint.iacr.org/2017/188. 2017.

[147] J. F. Ziegler and W. A. Lanford. “Effect of Cosmic Rays on Computer Memories”. In: *Science* 206.4420 (1979), pp. 776–788.

Part II

Publications

List of Publications

Journal Articles

[j1] Christoph Dobraunig, Maria Eichlseder, Daniel Kales, and Florian Mendel. "Practical Key-Recovery Attack on MANTIS$_5$". In: *IACR Transactions on Symmetric Cryptology* 2016.2 (2017), pp. 248–260. URL: http://tosc.iacr.org/index.php/ToSC/article/view/573.

[j2] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. "ISAP – Towards Side-Channel Secure Authenticated Encryption". In: *IACR Transactions on Symmetric Cryptology* 2017.1 (2017), pp. 80–105. URL: http://tosc.iacr.org/index.php/ToSC/article/view/585.

[j3] Hannes Groß, Erich Wenger, Christoph Dobraunig, and Christoph Ehrenhöfer. "Ascon hardware implementations and side-channel evaluation". In: *Microprocessors and Microsystems* 52 (2017), pp. 470–479. URL: https://doi.org/10.1016/j.micpro.2016.10.006.

Conference and Workshop Papers

[c1] Andrey Bogdanov, Christoph Dobraunig, Maria Eichlseder, Martin M. Lauridsen, Florian Mendel, Martin Schläffer, and Elmar Tischhauser. "Key Recovery Attacks on Recent Authenticated Ciphers". In: *Progress in Cryptology – LATINCRYPT 2014*. Ed. by Diego F. Aranha and Alfred Menezes. Vol. 8895. LNCS. Springer, 2015, pp. 274–287. URL: https://doi.org/10.1007/978-3-319-16295-9_15.

[c2] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Victor Lomné, and Florian Mendel. "Statistical Fault Attacks on Nonce-Based Authenticated Encryption Schemes". In: *Advances in Cryptology – ASIACRYPT 2016*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. LNCS. 2016, pp. 369–395. URL: https://doi.org/10.1007/978-3-662-53887-6_14.

[c3] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, and Florian Mendel. "Side-Channel Analysis of Keymill". In: *Constructive Side-Channel Analysis and Secure Design, COSADE 2017*. Ed. by Sylvain Guilley. Vol. 10348. LNCS. Springer, 2017, pp. 138–152. URL: https://doi.org/10.1007/978-3-319-64647-3_9.

[c4] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, and Florian Mendel. "On the Security of Fresh Re-keying to Counteract Side-Channel and Fault Attacks". In: *Smart Card Research and Advanced Applications, CARDIS 2014*. Ed. by Marc Joye and Amir Moradi. Vol. 8968. LNCS. Springer, 2015, pp. 233–244. URL: https://doi.org/10.1007/978-3-319-16763-3_14.

[c5] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. "Cryptanalysis of Simpira v1". In: *Selected Areas in Cryptography – SAC 2016*. To appear.

[c6] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. "Analysis of SHA-512/224 and SHA-512/256". In: *Advances in Cryptology – ASIACRYPT 2015*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9453. LNCS. Springer, 2015, pp. 612–630. URL: https://doi.org/10.1007/978-3-662-48800-3_25.

[c7] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. "Heuristic Tool for Linear Cryptanalysis with Applications to CAESAR Candidates". In: *Advances in Cryptology – ASIACRYPT 2015*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9453. LNCS. Springer, 2015, pp. 490–509. URL: https://doi.org/10.1007/978-3-662-48800-3_20.

[c8] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. "Related-Key Forgeries for Prøst-OTR". In: *Fast Software Encryption, FSE 2015*. Ed. by Gregor Leander. Vol. 9054. LNCS. Springer, 2015, pp. 282–296. URL: https://doi.org/10.1007/978-3-662-48116-5_14.

[c9] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. "Analysis of the Kupyna-256 Hash Function". In: *Fast Software Encryption, FSE 2016*. Ed. by Thomas Peyrin. Vol. 9783. LNCS. Springer, 2016, pp. 575–590. URL: https://doi.org/10.1007/978-3-662-52993-5_29.

[c10] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. "Forgery Attacks on Round-Reduced ICEPOLE-128". In: *Selected Areas in Cryptography – SAC 2015*. Ed. by Orr Dunkelman and Liam Keliher. Vol. 9566. LNCS. Springer, 2016, pp. 479–492. URL: https://doi.org/10.1007/978-3-319-31301-6_27.

[c11] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. "Higher-Order Cryptanalysis of LowMC". In: *Information Security and Cryptology – ICISC 2015*. Ed. by Soonhak Kwon and Aaram Yun. Vol. 9558. LNCS. Springer, 2016, pp. 87–101. URL: https://doi.org/10.1007/978-3-319-30840-1_6.

[c12] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. “Square Attack on 7-Round Kiasu-BC”. In: *Applied Cryptography and Network Security, ACNS 2016*. Ed. by Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider. Vol. 9696. LNCS. Springer, 2016, pp. 500–517. URL: https://doi.org/10.1007/978-3-319-39555-5_27.

[c13] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. “Cryptanalysis of Ascon”. In: *Topics in Cryptology – CT-RSA 2015*. Ed. by Kaisa Nyberg. Vol. 9048. LNCS. Springer, 2015, pp. 371–387. URL: https://doi.org/10.1007/978-3-319-16715-2_20.

[c14] Christoph Dobraunig, François Koeune, Stefan Mangard, Florian Mendel, and François-Xavier Standaert. “Towards Fresh and Hybrid Re-Keying Schemes with Beyond Birthday Security”. In: *Smart Card Research and Advanced Applications, CARDIS 2015*. Ed. by Naofumi Homma and Marcel Medwed. Vol. 9514. LNCS. Springer, 2016, pp. 225–241. URL: https://doi.org/10.1007/978-3-319-31271-2_14.

[c15] Christoph Dobraunig and Eik List. “Impossible-Differential and Boomerang Cryptanalysis of Round-Reduced Kiasu-BC”. In: *Topics in Cryptology – CT-RSA 2017*. Ed. by Helena Handschuh. Vol. 10159. LNCS. Springer, 2017, pp. 207–222. URL: https://doi.org/10.1007/978-3-319-52153-4_12.

[c16] Christoph Dobraunig, Florian Mendel, and Martin Schläffer. “Differential Cryptanalysis of SipHash”. In: *Selected Areas in Cryptography – SAC 2014*. Ed. by Antoine Joux and Amr M. Youssef. Vol. 8781. LNCS. Springer, 2014, pp. 165–182. URL: https://doi.org/10.1007/978-3-319-13051-4_10.

[c17] Hannes Groß, Erich Wenger, Christoph Dobraunig, and Christoph Ehrenhöfer. “Suit up! - Made-to-Measure Hardware Implementations of ASCON”. In: *2015 Euromicro Conference on Digital System Design, DSD 2015*. IEEE Computer Society, 2015, pp. 645–652. URL: https://doi.org/10.1109/DSD.2015.14.

[c18] Thomas Plos, Christoph Dobraunig, Markus Hofinger, Alexander Oprisnik, Christoph Wiesmeier, and Johannes Wiesmeier. “Compact Hardware Implementations of the Block Ciphers mCrypton, NOEKEON, and SEA”. In: *Progress in Cryptology – INDOCRYPT 2012*. Ed. by Steven D. Galbraith and Mridul Nandi. Vol. 7668. LNCS. Springer, 2012, pp. 358–377. URL: https://doi.org/10.1007/978-3-642-34931-7_21.

Informal and Other Publications

[i1] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. “Security Evaluation of SHA-224, SHA-512/224, and SHA-512/256”. Tech. Report CRYPTREC. 2015. URL: http://www.cryptrec.go.jp/english/estimation.html.

[i2] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. “Ascon”. Submission to the CAESAR competition: `http://competitions.cr.yp.to`. 2014.

Heuristic Tool for Linear Cryptanalysis with Applications to CAESAR Candidates

Publication Data

Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. "Heuristic Tool for Linear Cryptanalysis with Applications to CAESAR Candidates". In: *Advances in Cryptology – ASIACRYPT 2015*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9453. LNCS. Springer, 2015, pp. 490–509. URL: `https://doi.org/10.1007/978-3-662-48800-3_20`

The appended paper is an author-created extended version available at `https://eprint.iacr.org/2015/1200`. This extended version contains the found linear characteristics.

Contributions

- One of the main authors.

Heuristic Tool for Linear Cryptanalysis with Applications to CAESAR Candidates

Christoph Dobraunig, Maria Eichlseder, and Florian Mendel

Graz University of Technology, Austria
christoph.dobraunig@iaik.tugraz.at

Abstract. Differential and linear cryptanalysis are the general purpose tools to analyze various cryptographic primitives. Both techniques have in common that they rely on the existence of good differential or linear characteristics. The difficulty of finding such characteristics depends on the primitive. For instance, AES is designed to be resistant against differential and linear attacks and therefore, provides upper bounds on the probability of possible linear characteristics. On the other hand, we have primitives like SHA-1, SHA-2, and Keccak, where finding good and useful characteristics is an open problem. This becomes particularly interesting when considering, for example, competitions like CAESAR. In such competitions, many cryptographic primitives are waiting for analysis. Without suitable automatic tools, this is a virtually infeasible job. In recent years, various tools have been introduced to search for characteristics. The majority of these only deal with differential characteristics. In this work, we present a heuristic search tool which is capable of finding linear characteristics even for primitives with a relatively large state, and without a strongly aligned structure. As a proof of concept, we apply the presented tool on the underlying permutations of the first round CAESAR candidates Ascon, ICEPOLE, Keyak, Minalpher and Prøst.

Keywords: linear cryptanalysis · authenticated encryption · automated tools · guess-and-determine · CAESAR competition

1 Introduction

Research in symmetric cryptography in the last few years is mainly driven by dedicated high-profile open competitions such as NIST's AES and SHA-3 selection procedures, or ECRYPT's eSTREAM project. While these focused competitions in symmetric cryptography are generally viewed as having provided a tremendous increase in the understanding and confidence in the security of these cryptographic primitives, the impressive increase of submissions to such competitions reveals major problems related to the analytical effort for the cryptographic community. To better evaluate the security margin of the various submissions, automatic tools are needed to assist cryptanalysts with their work.

One important class of attacks is linear cryptanalysis [15, 25]. The success of these attacks relies on the existence of suitable linear characteristics. The

difficulty of finding such characteristics depends on the primitive. For example, the wide-trail design strategy [7] incorporated by AES provides lower bounds on the minimum number of active S-boxes in a linear characteristic and therefore, gives an upper bound on the highest possible bias. On the other hand, we have primitives with weak alignment [1], such as the winner of the SHA-3 competition Keccak, where finding good characteristics is an open problem, and heuristic search results are required to evaluate the security margin of the primitive. This is particularly interesting in the context of the CAESAR competition [26]. We noticed that many first round submissions focus their analysis on differential cryptanalysis, but provide only few results for linear cryptanalysis.

Our contribution. The main contribution of this paper is a dedicated automatic tool for linear cryptanalysis, which is available at github[1]. The tool performs heuristic searches for good linear characteristics in cryptographic primitives. It was designed for primitives based on substitution-permutation networks (SP networks).

The modular design of the tool allows easy extension to other cryptographic primitives. It also allows to easily develop and test new dedicated search strategies. To facilitate further improvements and analysis, the tool is publicly available and its source code is published together with this paper. Such a tool is particularly useful when designing new cryptographic primitives. It allows to easily explore the effects of, for instance, different S-boxes and linear layers on linear characteristics and reveals possible bad decisions in an early stage of the design process. Even in wide-trail designs with provable bounds, it can be useful to evaluate different choices for building blocks with respect to their long-term behaviour over a larger number of rounds, where the quality of the best characteristics can deviate significantly from the derived bounds (i.e., two algorithms with the same bounds may behave quite differently in a heuristic search, which can be a basis for the decision of choosing one design over the other).

As a proof of concept and to demonstrate the advantages of the tool, we have chosen the first round CAESAR candidates Ascon [9], ICEPOLE [19], Keyak [4], Minalpher [22] and Prøst [13] as analysis targets. Ascon, ICEPOLE, and Keyak are sponge-based authenticated encryption schemes. All three primitives use permutations that are not strongly aligned, making it hard to find good linear characteristics. We demonstrate the capability of our automated search tool by giving linear characteristics suitable for different attack scenarios. In comparison, the permutations used in Minalpher and Prøst provide more "structure" by incorporating an "AES-like" design strategy. Hence, the designers of these two primitives are able to give bounds on the minimum number of active S-boxes by using mixed-integer linear programming (MILP) for a number of rounds sufficient to thwart attacks. For Minalpher and Prøst, we show that our tool is capable of finding linear characteristics which match the provided bounds. Our results are summarized in Table 1 (Sect. 4).

[1] https://github.com/iaikkrypto/lineartrails

Related Work. While several automatic tools for differential cryptanalysis have been published in the last few years [5, 6, 8, 12, 14, 16, 20, 23], in particular for hash functions, the work on automatic tools dedicated to linear cryptanalysis is very limited. One example is a tool designed by Sun et al. [24], extending previous work of Mouha et al. [21]. They model the differential and linear behavior of a block cipher as a mixed-integer linear program (MILP) and use general-purpose MILP tools to solve the optimization problem (i.e., find the optimal characteristics for the – often simplified – model of the cipher). This approach works well for lightweight ciphers like Simon or Present, but faces problems when it comes to large-state and less structured ciphers such as Ascon, ICEPOLE, and Keyak. Hence, a dedicated search tool for linear characteristics will complement the existing tools.

Outline. This paper is divided into two main parts: the description of our new automated search tool for linear characteristics in Sect. 3, and its application to the CAESAR candidates in Sect. 4. However, first, we start with a short introduction to linear cryptanalysis and our notation in Sect. 2. Then, we deal with the propagation of linear masks in Sect. 3.2 and discuss the proposed search strategy for linear characteristics in Sect. 3.3. The application of the tool (Sect. 4) is first discussed in detail for Keyak in Sect. 4.1. Then, our results for the other ciphers are summarized and briefly discussed in Sect. 4.2 to 4.5. Finally, we conclude in Sect. 5.

2 Linear cryptanalysis

The goal of linear cryptanalysis [15, 25] is to identify good affine linear approximations for the target cipher. More specifically, we want to find linear equations between the plaintext bits, ciphertext bits and key bits that hold with probability significantly different from $\frac{1}{2}$ (bias). Then, in the actual attack phase, these equations can be used to derive information on the key bits from known plaintext-ciphertext pairs.

For linear cryptanalysis, the operation of the cipher, or building blocks of the cipher, is considered as a vectorial boolean function $f : \mathbb{F}_2^m \to \mathbb{F}_2^n$ (where the key bits might be part of $\mathbb{F}_2^m$). A (probabilistic) linear relation between input and output bits of f is then characterized by two linear masks $\alpha \in \mathbb{F}_2^m, \beta \in \mathbb{F}_2^n$. For $x \in \mathbb{F}_2^m$, $y \in \mathbb{F}_2^n$ with $y = f(x)$, the masks represent the relation

$$\alpha^t \cdot x = \beta^t \cdot y,$$

where $v^t \cdot w$ denotes the natural inner product of vectors. The quality of a linear approximation α, β is measured by the probability that the corresponding relation holds; or more precisely, by how far this probability deviates from the

average $\frac{1}{2}$. This deviation is referred to as the bias of the masks α, β:

$$\varepsilon_{\alpha,\beta} = \text{bias}_f(\alpha,\beta) = \left| \mathbb{P}\left[\alpha^t \cdot x = \beta^t \cdot y \mid y = f(x)\right] - \frac{1}{2} \right|$$
$$= \frac{1}{2^m} \cdot \left| \left| \left\{ x \in \mathbb{F}_2^m \mid \alpha^t \cdot x = \beta^t \cdot f(x) \right\} \right| - 2^{m-1} \right|.$$

If m is very small, the expression for $\varepsilon_{\alpha,\beta}$ can easily be evaluated explicitly for all masks α, β to determine the best masks. This information is summarized in the linear distribution table (LDT), where non-zero entries mark masks α, β with non-zero bias.

However, this is obviously infeasible for the complete cipher at once. To obtain an approximation of the complete cipher, it is split into smaller parts that are easier to analyze. Matsui's piling-up lemma [15] is used to combine the individual biases of multiple building blocks to derive the overall bias (under the assumption that the validity of the partial approximations is independent). If ε denotes the bias of the overall approximation of the block cipher, Matsui [15] showed that the necessary number of plaintext-ciphertext pairs to derive the bit of key information from the approximation is proportional to $\frac{1}{\varepsilon^2}$.

The difficult part is to find a network or "trail" of partial approximations that are compatible with each other and give a good overall bias. In particular, each involved approximation must have non-zero bias, otherwise the overall bias becomes zero. For this reason, we refer to non-zero entries in the individual LDTs as "valid transitions" of masks for this building block. In the the following, such a "trail" of partial linear approximations is called linear characteristic.

Several algorithms and improvements thereof have been proposed for finding characteristics with the highest overall bias, typically by a sort of branch-and-bound algorithms. For more complex, modern ciphers, such a complete search is not feasible. Two possible approaches to handle this situation are (a) to design ciphers in a way to allow to prove bounds on the best possible bias, and (b) to use heuristic search methods to find stronger biases (for reduced versions of the cipher) to make better predictions on the security margin of the complete cipher.

In the following, we will focus on the second approach, and heuristically search for good characteristics. Unlike the original, complete search algorithms, our search will not proceed in a "linear", round-by-round manner. Instead, we will take inspiration from similar searching tools for differential cryptanalysis [8], and randomize the search order. This naturally implies that we will often start building inconsistent characteristics, which will need to be fixed or discarded.

3 An automated tool for linear cryptanalysis

The proposed automated tool can be roughly split into two main parts. The first part is described in Sect. 3.2 and deals with the description of cryptographic primitives within the search tool, including the representation of linear approximations and, most importantly, their propagation. The other part of the

tool is the choice of the search algorithm to find good linear characteristics (see Sect. 3.3). Before we start with the description of the tool, we take a look at the requirements we have for the design and implementation of such a heuristic search tool.

3.1 Implementation requirements for the search tool

In order for any automatic cryptanalysis tool to be useful for general application, for example to analyze the 57 first round CAESAR submissions, there are a number of flexibility and usability requirements:

- **Easy to add new primitives.** This is one of the main goals for the creation of this tool. To fulfill this requirement, we have decided to put the focus on primitives based on SP networks, i.e., with alternating S-box and linear layers. This simplifies the design process of the tool, since we did not have to consider every possible specialty, while still having a large group of applicable primitives. The programming interface should be designed to require as little effort as possible for converting, for example, a CAESAR reference implementation to a suitable cipher definition for the tool – ideally, it should be possible to just copy the corresponding code fragments for the round transformation steps.
- **An easily adaptable, parameterized search algorithm.** The linear tool implements a heuristic guess-and-determine search algorithm. This algorithm delivers good results for various primitives. However, the success of the search is highly dependent on various different parameters, such as the configuration of the searching order and conflict-handling behavior. Therefore, it is crucial that these parameters can be adjusted easily. For this reason, our standard guess-and-determine algorithm is parameterizable via an XML-file. This XML-file specifies the search starting point and allows to configure various other parameters.
- **Easy to add other search algorithms.** The currently implemented, stack-based guess-and-determine algorithm is certainly not the only possible way to search for linear characteristics. To be open for new ideas and evaluate other algorithms, we have designed the tool in a way that the search algorithm is clearly separated from the description of the cipher and thus, can be replaced easily. This opens the door for experiments with various alternative search algorithms and will hopefully lead to new insights in this direction.
- **Portability of the code.** We do not want the tool to require a specific operating system or platform to run. Therefore, we have reduced the dependence from external libraries whenever possible, and omitted the use of platform-specific instructions.

3.2 Propagation of linear masks

Our overall search strategy is based on the "guess-and-determine" approach. We want to build a consistent linear characteristic with high bias step by step,

starting from a "mostly unknown" (undetermined) characteristic of masks, and progressively deciding which bits should be selected (activated) by the final mask. For this purpose, we repeatedly "guess" the value of small parts of the masks, and then "determine" the consequences of this guess (in particular, whether this updated partial characteristic can still be completed to a "valid" characteristic). We refer to the "determining" step as propagation of information.

Representation of partial linear masks. The tool represents the linear masks on bit-level. During the search, we work with partially-determined search masks. We represent an active bit in the linear mask with `1` and an inactive bit of a linear mask with `0`. Mask bits that are not yet determined are represented by `?`.

Propagation in SP networks. We want to find linear characteristics for SP networks. Such a network consists of iterative applications of a substitution layer (consisting of relatively small S-boxes) and an (affine) linear layer (which typically covers larger parts of the state at once). We use different techniques for the propagation of information in these two layer types. The goal of the propagation step is to investigate whether the guess allows to derive explicit values for other ("neighbouring") bits, and in particular whether this explicit information is contradictory. The constraints that allow this propagation can be derived from the linear distribution table of the involved functions, since the characteristic must not contain any mask transitions with bias 0.

Propagation in the non-linear layer. We only deal with non-linear layers which can be represented by parallel applications of S-boxes. So the propagation of the linear masks at the input and the output of the S-boxes can be treated individually, since the parallel applications are considered independent of each other (any dependencies induced by the linking linear layers are treated separately). Therefore, we can do the propagation separately per S-box.

Many state-of-the-art ciphers use relatively small S-boxes. In many recent cipher proposals, the S-boxes map 4- to 5-bit inputs to outputs of the same size. Even the largest S-boxes hardly ever exceed a size of 8 bits. Therefore, the propagation of the linear masks can be done in a brute-force manner, based on the linear distribution table (LDT) of the S-box. The LDT is an exhaustive list of all valid (biased) mask transitions from mask α to mask β. Our current "knowledge" of the values of some input and output mask bits limits the set of available transitions. Depending on the concrete values of α and β and the remaining transition options, we have one of the following outcomes of the propagation:

1. **Contradiction:** The LDT reveals that no valid, biased transitions remain that satisfy the fixed mask bits; i.e., there is no linear relationship involving the bits currently marked by α and β as `1` (and optionally the `?` bits). In other words, we have a contradiction. This means that the current, partially

determined linear characteristic is in fact invalid. This situation has to be handled by the search algorithm by, e.g., backtracking and resolving the contradiction.

2. **Updated bits:** The LDT reveals that one or more biased transitions respecting the partially determined α and β remain. In addition, all remaining transitions share the same value (`0` or `1`) for one or more of the current `?` bits. Thus, we can refine some previously undefined bits in the masks to active or inactive bits by using information from the LDT. Before taking any further guesses, this newly-won information must in turn be propagated in all connected function components.
3. **No updates:** The LDT reveals that α and β are possible, but no additional explicit bit-wise information can be won. Nothing else happens.

Propagation in the linear layer. There are two main differences between the linear and non-linear layers from the propagation perspective: On the one hand, the linear layer typically involves significantly more variables than individual S-boxes. On the other hand, propagating partial linear masks for linear functions can be achieved easily using basic linear algebra.

Assume that the function $f : \mathbb{F}_2^m \to \mathbb{F}_2^n$ is linear, i.e., $f(x) = A \cdot x = y$ for some $A \in \mathbb{F}_2^{n \times m}$. Note that we can include affine linear functions in the same model, since the affine (constant) part is irrelevant for the bias of the linear model if we do not consider the sign of the bias. Then, for a fully determined mask α, β, the bias $\varepsilon_{\alpha,\beta}$ is either 0 (wrong model) or $\frac{1}{2}$ (exact, correct model). More specifically, α, β is a valid input-output mask if

$$\begin{aligned} \forall x :\ \alpha^t \cdot x = \beta^t \cdot f(x) \quad &\Leftrightarrow \quad \forall x :\ \alpha^t \cdot x = \beta^t \cdot (A \cdot x) \\ &\Leftrightarrow \quad \forall x :\ \alpha^t \cdot x = (A^t \cdot \beta)^t \cdot x \\ &\Leftrightarrow \quad \alpha = A^t \cdot \beta. \end{aligned}$$

If α and β are only partially determined, all propagation can be performed by propagating the information in the linear system $\alpha = A^t \cdot \beta$. For this purpose, we always keep the half-solved system in reduced row echelon form for all linear layers. Whenever mask values in α or β are updated, we perform partial Gaussian elimination to retain reduced row echelon form. If in the process, other bits of α or β are determined (case 1 or 2 from above), this information is extracted from the system and instead stored in the regular representation α, β of the mask bits that is also used for S-box propagation.

Update process. Every time the propagation step leads to new, explicit information in the linear masks (i.e., mask bits that were previously undetermined are now fixed, case 2), this information has to be propagated over the connected linear or non-linear layers, which share those updated mask bits. In other words, the propagation step needs to be iterated to update the neighbouring layers. Since we require that every linear layer is only connected to non-linear layers and vice versa, we can use a very simple update process scheduling: After each

guess or update, we first perform propagation on all non-linear layers (with updated bits), then on all linear layers (with updated bits). This process is iterated until the propagation process has converged and no additional explicit information can be learned anymore, or a contradiction is detected.

3.3 Search for linear characteristics

In this section, we discuss our proposed search strategy. The search strategy guides the guessing behavior (choice of bits or bit sets to guess, and their values), as well as the backtracking behavior after detecting contradictions. We currently implement a simple stack-based search algorithm, similar to the strategy used in recent tools for differential characteristic search [16, 17]. We first give an algorithmic overview, before detailing the choices made for individual ingredients.

Basic search algorithm. We start from a mostly-undetermined characteristic A_0 as a starting point, and incrementally guess more and more of its mask bits. We refer to the current characteristic as A, and keep a history of the guesses that led from A_0 to A in the stack S. For each guess, we select a guessable item X in the current characteristic A. Depending on the search strategy, this can be a single bit, or all bits of an S-box input-output mask (unlike some tools for differential characteristic search which only consider individual bits). The choice of X is guided by the search and backtracking strategy. The characteristics stored in S are used for backtracking, where some of the most recent guesses are undone to resolve conflicts, i.e., we return to an older status stored in S. The basic search algorithm is summarized in Algorithm 1.

Algorithm 1 Guess-and-determine search algorithm

```
choose characteristic A_0 as starting point
loop
  push A_0 to empty stack S
  repeat
    get the topmost characteristic A from S
    select a guessable item (bit or S-box) X in A
    for all most preferable possible values x of X do
      guess X to x
      propagate information
      if contradiction detected then
        undo guess x and all resulting updates
      else
        push A to S and break
    if no valid assignment x was found then
      backtrack by popping characteristics from S
      mark X critical
  until exhausted or solution characteristic found
```

Choice of the starting point. The starting point is a linear characteristic, in which most mask bits are still undetermined. The appearance of the starting point depends highly on the scenario in which the linear characteristic will be used, since it can be used to define which bits of the resulting characteristic must definitely be active or inactive.

For instance, consider the search for a linear characteristic for a block cipher or a permutation. In principle, every bit of the input or the respective output mask can be active in such a scenario. So, we can use a starting point where nearly every bit of the respective input and output linear masks is free for guessing during the search. On the other hand, if we consider sponge-like modes, we have more restrictions on the characteristic. Here, the attacker can only observe or control a fraction of the state on the input and the output. Depending on the actual attack, it can be necessary that bits belonging to unknown parts of the state remain inactive, and that only observable or controllable bits are active.

Besides defining the possible use-cases of the linear characteristic, the choice of the starting point also greatly influences the expected success of the search. By fixing parts of the starting point, it is possible to reduce the search space significantly, and thus accelerate the search to quickly find results that would otherwise be out of range. However, reducing the search space also has the potential to exclude classes of good characteristics. Thus, the starting point is usually not too much restricted at the beginning of the analysis of a certain cipher. Instead, the choice of the starting point is an adaptive process based on the cryptographer's intuition and the cipher's structure, using information from previous searches.

Guessing strategy. The guessing strategy specifies which undetermined bit or S-box is picked next for guessing, and how it will be refined. In S-box-based designs, the search success can profit significantly from guessing in an S-box-oriented manner; that is, by guessing the value of all bits in an S-box input-output mask at once. We refer to this as "guessing the S-box". Even if guesses are made S-box after S-box, the propagation procedure can produce half-guessed S-boxes with some bits fixed and others undetermined. It is also possible to mix S-box-wise and bit-wise guessing.

We refer to an S-box as "guessable" if the linear input and output masks contain at least one remaining undetermined `?`-bit, and "fixed" or "not guessable" otherwise. In addition, the search configuration may limit the selection of S-boxes currently eligible for guessing, depending on the guessing progress. The most important example for this is the "critical" status that is assigned to an S-box after a failed attempt to find any valid assignment for this S-box, and assigns top priority to this S-box. Additionally, it can be useful to impose cipher-specific rules; for example, to demand that all S-boxes of the first few rounds must be fixed before we start guessing values in the following rounds.

To guide the guessing procedure, each guessable S-box is assigned a probability for being selected as the next guessing target, for example based on the criteria described above. In addition, all possible assignments for a guessable

S-box are ranked by how promising they are estimated to be for high-bias characteristics. Of course, the primary guess for potentially inactive S-boxes (i.e., only with bits `0` and `?` so far) is to keep them inactive (i.e., all `0`). If this is not possible, the S-box is marked as active. If the selected guessable S-box is already marked active, we rank all possible assignments of the linear masks according to their linear bias and the number of active bits. We pick a random optimally ranked assignment as primary guess. If the following propagation reveals that this assignment is in fact impossible, we try other assignments until no alternative is left, or we have reached a predefined threshold on the number of trials.

Backtracking. If all alternative assignments fail (or a predefined threshold of trials is reached), we need to backtrack. To resolve this conflict, we return to an earlier version of the linear characteristic as stored on the stack S. Again, we try to guess the same critical S-box that caused the conflict. If we cannot resolve the conflict here, we jump further back, until it can be resolved.

Restarts. To better randomize the search process and avoid being "stuck" with a few unhappy first guesses, it is helpful to occasionally restart the complete search. For this purpose, we define a limit of "credits" or resources for one search run. When this limit is exhausted before finding a valid, fully determined characteristic, we clear the stack S and restart from scratch with the starting point A_0. Additionally, the search is also restarted after completing a successful run, with the hope of finding new, better characteristics. If the cryptographer detects promising patterns in the preliminary results, these can serve as a basis for improved starting points for the next run.

4 Application to CAESAR candidates

In this section, we demonstrate the advantages of our tool for linear cryptanalysis by applying it to several first round CAESAR candidates: Keyak, Ascon, ICEPOLE, Prøst, and Minalpher. All the analyzed candidates are permutation-based (rather than based on block ciphers). This is, however, not a constraint of the linear tool, which works just as well for block ciphers, since the typical round-key additions do not influence the linear characteristics. Rather, it is representative of the trend that a significant portion of CAESAR candidates with new, dedicated SPN primitives are permutation-based, since most block-cipher modes employ AES.

For each candidate, we first consider linear characteristics for the (round-reduced) permutation. However, for many modes (in particular for sponges), an attacker cannot influence the complete input to the permutation, or cannot observe its complete output. For this reason, we also investigate characteristics with additional constraints, where parts of the linear masks are fixed beforehand. We define the following three types of linear characteristics:

- **Type I (permutation):** For this type of characteristics, we do not require any additional restrictions regarding the positions of active bits in the linear characteristic. Hence, a characteristic of this type might not be usable in a concrete attack on the duplex-like constructions of KEYAK, ASCON, and ICEPOLE. Nevertheless, even for modes where Type-I characteristics allow no direct attacks, they still give insights in the resistance of the cryptographic primitive against linear attacks.
- **Type II (output constrained):** Linear characteristics of this type have the restriction that all active bits at the end of the characteristic have to be "observable". For duplex-like constructions, this means that all active mask bits have to be in the outer (rate) part of the state. Such linear characteristics can be used to create key-stream distinguishers in known-plaintext scenarios for duplex-like constructions, or even to perform key-recovery attacks.
- **Type III (input and output constrained):** In addition to Type-II characteristics, also all active bits of the input have to be in the outer (rate) part of the state. This type of linear characteristic can act as a key-stream distinguisher in known-plaintext scenarios for duplex-like constructions, targeting the encryption of the plaintext. A similar type of linear relations has been used for instance by Minaud [18] to detect linear biases in the key-stream of the CAESAR candidate AEGIS.

We first discuss our approach and our findings for KEYAK in more detail, and then briefly present our results for ASCON, ICEPOLE, PRØST, and Minalpher.

4.1 Keyak

Brief description of Keyak. KEYAK is a family of authenticated encryption algorithms designed by Bertoni et al. [4] and is one of the 57 submissions to the first round of the CAESAR competition. It is based on the round-reduced KECCAK-f permutation and follows the duplex construction [2]. The designers have defined four instances of KEYAK; all instances share the same capacity $c = 252$ and use 12 rounds of the KECCAK-f permutation, but differ in their state size b and the degree of parallelism p:

- **River Keyak:** $b = 800$, $p = 1$ (serial),
- **Lake Keyak:** $b = 1600$, $p = 1$ (serial),
- **Sea Keyak:** $b = 1600$, $p = 2$ (parallel),
- **Ocean Keyak:** $b = 1600$, $p = 4$, (parallel).

The KEYAK *duplex mode.* Fig. 1 sketches the encryption of serial KEYAK without associated data: The initialization takes as input the secret key K and public nonce N, and applies the permutation f once. This ensures that one always starts with a random-looking state at the beginning of the encryption of the plaintext. Afterwards, the plaintext is processed by xoring it block-wise to the internal state, separated by invocations of the permutation f. The ciphertext blocks are extracted from the state after adding the plaintext. After all data is

processed, the finalization applies the permutation f once more and returns the tag. For more details on KEYAK, including the rules for processing associated data, we refer to the specification [4].

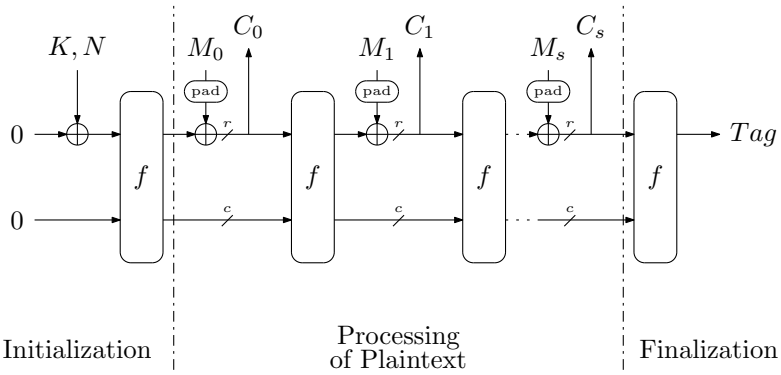


Fig. 1: Simplified sketch of LAKE KEYAK encryption (without associated data).

The KEYAK *permutation.* The KEYAK permutation is a round-reduced version of the KECCAK-f permutation, reduced to 12 rounds. It operates on the $5 \times 5 = 25$ w-bit words ("lanes") $S[x][y][*]$ of the state S, with $w = 32$ or 64. Each round applies the five steps $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$, where all steps except ι are equivalent for each round.

- Step θ adds to every bit of the state $S[x][y][z]$ the bitwise sum of the neighbouring columns $S[x-1][*][z]$ and $S[x+1][*][z-1]$. This procedure can be described by the following equation:

$$\theta: \quad S[x][y][z] \leftarrow S[x][y][z] + \sum_{y'=0}^{4} S[x-1][y'][z] + \sum_{y'=0}^{4} S[x+1][y'][z-1].$$

- Step ρ rotates the bits in every lane by a constant value,

$$\rho: \quad S[x][y][z] \leftarrow S[x][y][z + C(x,y)],$$

  where $C(x,y)$ is a constant value.
- Step π permutes the lanes using the following function:

$$\pi: \quad S[x][y][*] \leftarrow S[x'][y'][*], \qquad \text{where } \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \cdot \begin{pmatrix} x' \\ y' \end{pmatrix}.$$

- Step χ is the only non-linear step in KECCAK and operates on each row of 5 bits:

$$\chi: \quad S[x][y][z] \leftarrow S[x][y][z] \oplus ((\neg S[x+1][y][z]) \wedge S[x+2][y][z]),$$

  which can be seen as applying a 5-bit S-box in parallel to all rows.

– Step ι adds a round-dependent constant to the state. For the actual values of the constants, we refer to the design document [4].

The designers provide some results on the linear properties of this permutation online, as part of the KeccakTools package [3].

Results for Keyak. For our analysis, we focus on the primary recommendation Lake Keyak using state size $b = 1600$. Since Lake Keyak, in contrast to Ascon and ICEPOLE, uses the same permutation (with the same number of rounds) in the initialization, finalization, and plaintext-processing phase, Type-III characteristics (to target plaintext-processing) offer no remarkable advantage over Type-II characteristics (to target the initialization). For this reason, we only consider Type-I and Type-II characteristics.

Type-I characteristics (for 3 and 4 rounds of the permutation). We first consider Type-I characteristics, i.e., linear characteristics for the underlying round-reduced Keyak permutation (Keccak-f) without any additional restrictions. We performed a search for linear characteristics for 4 rounds of the 1600-bit permutation. The best linear characteristic we found is given in Table 2b. In total, this characteristic only has 33 active S-boxes, which results in a bias of 2^{-34}. The best linear characteristic for 3 rounds with 13 active S-boxes and a bias of 2^{-14} can be obtained by omitting the first round of the 4-round linear characteristic. Our results are very similar to the characteristic given in the KeccakTools package [3].

Type-II characteristics (for 3 and 4 rounds of the initialization). The previous 3 and 4-round characteristics have active bits in the inner part (last four 64-bit words) of the state after round 4. Therefore, we cannot use this characteristic in an actual attack. For an attack on the initialization of round-reduced Keyak, we have to apply additional restrictions on the linear characteristics. Since we can only observe the outer (rate) part of the state at the output of the permutation after the initialization, we apply the restriction that only this part is allowed to contain active bits. Note that the input of the first permutation call is either known or constant. Therefore, we have no problems with active bits there.

For the initialization reduced to 3 rounds, we found the characteristic in Table 3a, and for the 4-round version, the characteristic given in Table 3b. In the last round, both characteristics only have active S-boxes in the rate part of the state. Thus, considering a known-plaintext attack, we know all the output bits of these S-boxes and can invert them. This leads to the fact that the last round does not influence the bias. So we have an expected bias of 2^{-13} for the 3-round version, and 2^{-49} for the 4-round version of these characteristics. Taking the last S-box layer also into account, the bias of those characteristics would be 2^{-26} and 2^{-70}, respectively. When inverting the last S-boxes, both characteristics result in trivial key-stream distinguishers for round-reduced versions of Keyak with complexity 2^{26} and 2^{98}, respectively. Moreover, these distinguishers could also be used in a key-recovery attack on round-reduced Keyak, resulting in similar complexities.

Configuration of the search. As already mentioned, the proposed search tool is a heuristic one and thus, the quality of the results heavily depends on the applied heuristic search criteria, as well as on the definition of the starting points. For the search process that led to the Type-II characteristics for 3 and 4 rounds of KEYAK, we used a quite natural starting point: For both starting points, the only restriction is that the S-boxes of the last round which "belong" to the inner part of the state must be inactive. In addition, one S-box in the second round is marked as active (to exclude the trivial, entirely inactive solution).

We split the search into two stages. In the first stage, we only pick potentially inactive guessable S-boxes, and set them to the best possible assignment (typically a completely inactive input and output linear mask). Which S-box is picked and refined is determined by a heuristic that picks the S-boxes according to a previously configured weight distribution. These weights can be manually assigned in the search configuration file (the same file in which the starting point is defined). In case of the search for the 3-round Type-II characteristic, the weights were assigned so that S-boxes of the first and second round have a 50 times higher chance to be picked compared to an S-box of the last round. The intention behind this distribution is that the majority of the active S-boxes of the resulting linear characteristic should be located in the last round, because their output is known in an attack. Hence, these active S-boxes can be inverted and do not contribute to the bias. Our heuristic for the 4-round Type-II characteristic prefers S-boxes from rounds 2 and 3 over S-boxes from rounds 1 and 4. Additionally, round 1 is favored over the last round 4. In the second stage, after every guessable and potentially inactive S-box is already determined, we continue by guessing active and yet not fully determined S-boxes until the linear characteristic is fully determined.

As can be seen above, the choice of the starting point and search heuristic depends on the structure of the target primitive, the planned use for the linear characteristic, and on the intuition of the cryptographer. Thus, better search strategies and starting points might exist, which may lead to better linear characteristics than those given in this paper.

4.2 Ascon

Brief description of Ascon. ASCON is a family of sponge-based candidates, designed by Dobraunig et al. [9]. Compared to KEYAK, it features a significantly smaller state of 320 bits, and the linear layer is applied to each of the 5 64-bit words independently. The 5-bit S-box, on the other hand, is closely related (affine equivalent) to that of KEYAK. The primary proposal ASCON-128 has a rate of 64 bits and hence, a capacity of 256 bits.

Results for Ascon. For the linear tool, the simple design of the linear layer means that its linear model can be split into 5 separate, independent matrices. Combined with a small state size, this property greatly reduces the cost for linear algebra needed to perform the propagation compared to KEYAK.

Our findings for Ascon are summarized in Table 1. The number of active S-boxes of Type-I characteristics found with the help of this tool have already been included in work presented at CT-RSA 2015 [10]. Note that the characteristics given here are optimized for a minimum number of active S-boxes, rather than minimal bias. For Ascon-128, we additionally search for Type-II and Type-III characteristics. However, regarding Type-III characteristics, no meaningful results were obtained.

4.3 ICEPOLE

Brief description of ICEPOLE. ICEPOLE is a family of authenticated encryption schemes designed by Morawiecki et al. [19]. It consists of the three proposals ICEPOLE-128, ICEPOLE-128a, and ICEPOLE-256a, which all use the same underlying 1280-bit permutation. All variants use 12 rounds of the permutation for initialization, and 6 rounds for processing of plaintext and finalization. However, they differ in details like size of the rate, key, nonce and tag.

The 1280-bit state of ICEPOLE is stored in $5 \times 4 = 20$ 64-bit words. For the linear layer, an MDS matrix over $\mathbb{F}_{2^5}$ is first applied 64 times in parallel (to each 20-bit slice of the state). Then, each word is rotated, and the words swap positions. The S-box layer applies 5-bit S-boxes (4 parallel row-wise applications for each 20-bit slice).

ICEPOLE's designers perform no dedicated linear analysis, but compare the cipher's resistance to linear cryptanalysis to its well-studied resistance against differential cryptanalysis. They conclude that the attack complexity after 5–6 rounds should be "completely intractable" [19]. At FSE 2015, Huang et al. [11] presented 3-round linear characteristics that they use in a differential-linear attack on ICEPOLE.

Results for ICEPOLE. The Type-II and Type-III characteristics given in Table 1 are constrained with respect to a capacity of 254 bits (due to padding, 256 bits are not observable), as defined for ICEPOLE-128 and ICEPOLE-128a. In the case of ICEPOLE, we do not have an immediate output of a ciphertext block right after the 12 rounds of the initialization. Before this happens, there is the option to process a secret message number and at least an empty associated data block is processed. Hence, 6 or even another 12 additional rounds have to be passed before an output suitable for our Type-II characteristic is accessible. Thus—in the worst case—our key-stream distinguisher using Type-II characteristics works for round-reduced versions of ICEPOLE-128, where the initialization plus the following processing is reduced to 5 out of 24 rounds with a complexity of about 2^{120}.

Type-III characteristics can be used to create distinguishers that target the processing of the plaintext. Here, every version of ICEPOLE uses the 6 round version of the ICEPOLE permutation. Thus, by using the Type-III characteristic in Table 1, the key-stream produced by round-reduced variants of ICEPOLE-128,

where the permutation used in the plaintext processing is reduced to 4 (out of 6), rounds can be distinguished from a perfect randomly generated key-stream with a complexity of about 2^{88}. The bias of the 5-round Type-III characteristic is $2^{-87.08}$ and hence, the complexity of a resulting key-stream distinguisher cannot harm the 128-bit security of ICEPOLE-128. ICEPOLE-256a, on the other hand, claims a security level of 256 bits regarding the confidentiality. However, it has a higher capacity of 318 bits and therefore, the characteristics given in Table 1 cannot be used directly. Taking the higher capacity of ICEPOLE-256a into account, we get the Type-III characteristic (bias $2^{-89.49}$) shown in Table 9a, which can be used to distinguish the key-stream of a round-reduced variant of ICEPOLE-256a, where the permutation used during the encryption is reduced to 5 (out of 6 rounds). Note that ICEPOLE-256a limits the number of blocks encrypted under a single key by 2^{62}. However, this type of key-stream distinguishers exploit relations between ciphertext block C_i and the key-stream used to generate the following ciphertext block C_{i+1}. Thus, distinguishers using Type-III characteristics in this way do not rely on the fact that always the same key is used.

Table 1 contains the results with the best bias, but not necessarily the minimal number of active S-boxes we found. For example, for 6 rounds, we also found a Type-I characteristic with only 103 active S-boxes, but a bias of $2^{-133.49}$ (compared to 104 active S-boxes with bias $2^{-126.32}$ as in the table).

4.4 Minalpher

Brief description of Minalpher. Minalpher is designed by Sasaki et al. [22]. In contrast to the previous 3 candidates, Minalpher is no sponge-based design. Instead, the permutation is applied in a new tweakable block cipher construction, called tweakable Even-Mansour. For this construction, the permutation size only needs to be twice the security level, so for 128-bit security, Minalpher has the smallest of all investigated permutation sizes with only 256 bits. This small state is further divided into two halves, whose only interaction in each of the 17.5 rounds is that one half is once xored to the other half, and the two halves swap places. Besides the interaction between the halves and some nibble reordering, the linear layer features a near-MDS matrix multiplication over $\mathbb{F}_{2^4}$. The S-box size of 4 bits is also nibble-oriented.

For Minalpher's construction, only Type-I characteristics are useful. We understand our results as an analysis of the underlying permutation. However, since Minalpher claims security in nonce misuse settings and under unverified plaintext release, the Type-I characteristics could also be useful for attacks on the cipher. In particular, for a fixed nonce, the construction allows to control the entire permutation input (at least differentially, due to the Even-Mansour construction, which xors a key- and nonce-dependent value before and after the permutation) and observe the entire output (again, differentially).

The designers analyze the minimum number of active S-boxes (for differential cryptanalysis) theoretically, and prove a minimum number of 22 S-boxes for 4 rounds. For up to 7 rounds, they extend the bounds with the help of mixed integer

linear programming (MILP). The bounds obtained this way for the numbers of rounds r also covered by this work are 22 ($r = 4$), 41 ($r = 5$), and 58 ($r = 6$). The designers claim that the same bounds apply for linear cryptanalysis.

Results for Minalpher. The existing bounds serve as a kind of benchmark for our tool to check its capabilities. As shown in Table 1, we were able to match the given bounds for up to 6 rounds. For better comparability, we included our results with the minimal number of active S-boxes, but not necessarily the best bias, in the table. For example, for 6 rounds, we also found a Type-I characteristic with a smaller bias of 2^{-61}, but with 60 active S-boxes (compared to 58 active S-boxes with bias 2^{-62} in the table).

4.5 Prøst

Brief description of Prøst. PRØST, designed by Kavun et al. [13], offers both a sponge-based mode and two block-cipher-based modes, where the latter use the PRØST permutation in a single-key Even-Mansour construction. Each of the three modes offers two security levels: one based on the 256-bit PRØST-128 permutation, and one based on the 512-bit PRØST-256 permutation. The state is stored as 4×4 words of 16 or 32 bits, respectively. Both the 4-bit S-box (row-wise) and the 16-bit linear mixing function (MDS over $\mathbb{F}_{2^4}$ are applied in a bit-sliced way (using 1 bit of each word). Then, each word is rotated. The number of rounds per permutation call is $r = 16$ (PRØST-128) or $r = 18$ (PRØST-256), respectively.

We focus our analysis on PRØST-256 (formally offering 128-bit security). Like Minalpher, PRØST comes with a MILP-based proof for the minimum number of active S-boxes for differential and linear characteristics. For PRØST-256, the bounds for different round numbers are 25 ($r = 4$), 41 ($r = 5$), 105 ($r = 6$), and 169 ($r = 7$).

Results for Prøst. Again, we used the existing bounds as benchmarks for our linear tool. The tool is able to match each bound, mostly with optimal or near-optimal bias (2^{-26} for $r = 4$, 2^{-42} for $r = 5$, 2^{-107} for $r = 6$, and 2^{-175} for $r = 7$).

Table 1: Results for KEYAK, ASCON, ICEPOLE, Minalpher, and PRØST.

Cipher	Type	Rounds	Active S-boxes	Bias	Reference
KEYAK	I	3	13	2^{-14}	Table 2a
		4	33	2^{-34}	Table 2b
	II	3*	12	2^{-13}	Table 3a
		4*	43	2^{-49}	Table 3b
ASCON	I	3	13	2^{-15}	Table 4a
		4	43	2^{-50}	Table 4b
		5	67	2^{-94}	Table 4c
	II	2	6	2^{-8}	Table 5a
		3	23	2^{-30}	Table 5b
		4	61	2^{-83}	Table 5c
ICEPOLE	I	5	38	$2^{-55.08}$	Table 6a
		6	104	$2^{-126.32}$	Table 6b
	II	4	22	$2^{-30.42}$	Table 7a
		5	38	$2^{-59.49}$	Table 7b
	III	3	10	$2^{-16.66}$	Table 8a
		4	22	$2^{-43.25}$	Table 8b
		5	42	$2^{-87.08}$	Table 8c
Minalpher	I	4	22	2^{-23}	Table 10a
		5	41	2^{-42}	Table 10b
		6	58	2^{-62}	Table 10c
PRØST-256	I	4	25	2^{-26}	Table 11a
		5	41	2^{-42}	Table 11b
		6	105	2^{-107}	Table 11c
		7	169	2^{-175}	Table 11d

* Last S-box layer inverted.

5 Conclusion

We presented a dedicated tool for the automatic linear cryptanalysis of substitution-permutation networks. The goal of the tool is to identify linear characteristics for a cryptographic function, which can subsequently be used by the cryptanalyst to mount key-recovery or distinguishing attacks. The heuristic search is based on an efficient guess-and-determine approach, which has previously been proven successful for searching differential characteristics. We described how to perform efficient propagation of linear masks in linear and non-linear building blocks of a cipher.

From the cryptanalyst's perspective, the tool is simple to use, flexible, and easy to extend with regard to search strategies and target ciphers. The open-

source tool will be freely available to help analyze CAESAR candidates and other symmetric cryptographic primitives. We hope that our work will be a valuable contribution to get a better understanding of the security of these ciphers regarding linear cryptanalysis. In particular, we hope to encourage experiments with alternative, sophisticated search strategies optimized for different target ciphers.

We demonstrated the efficiency of our tool by applying it to several CAESAR candidates. The results obtained by searching for linear characteristics for the Minalpher and PRØST-256 permutation show that the presented heuristic search tool can keep pace with MILP-based approaches. However, due to the heuristic nature, we are not capable of providing bounds on the minimum number of active S-boxes.

On the other side, when looking at the results obtained for ASCON, ICEPOLE and KEYAK– all designs with weak alignment – we have been able to find new linear characteristics with a good bias that might be used in a key-recovery or distinguishing attack on round-reduced versions of the ciphers in the future. One highlight are the Type-III characteristics for round-reduced versions of ICEPOLE, which can be used to distinguish the key-stream of ICEPOLE in a nonce-respecting scenario.

Our results show that the existence of a publicly available analysis tool for linear characteristics will be of great help in the design of symmetric cryptographic primitives, in order to evaluate the resistance against linear attacks already in an early stage of the design. Thus, we think that this tool will facilitate new designs which are more balanced in their resistance against linear and differential attacks than some of today's designs.

Acknowledgments. The work has been supported in part by the Austrian Science Fund (project P26494-N15) and by the Austrian Research Promotion Agency (FFG) and the Styrian Business Promotion Agency (SFG) under grant number 836628 (SeCoS).

References

1. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On alignment in Keccak. `http://keccak.noekeon.org/KeccakAlignment.pdf`
2. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the sponge: Single-pass authenticated encryption and other applications. In: Miri, A., Vaudenay, S. (eds.) Selected Areas in Cryptography – SAC 2011. LNCS, vol. 7118, pp. 320–337. Springer (2011)
3. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: KECCAKTOOLS software. `http://keccak.noekeon.org/` (2014)
4. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: Keyak. Submission to the CAESAR competition: `http://competitions.cr.yp.to/round1/keyakv1.pdf` (2014)
5. Biryukov, A., Velichkov, V.: Automatic search for differential trails in ARX ciphers. In: Benaloh, J. (ed.) Topics in Cryptology – CT-RSA 2014. LNCS, vol. 8366, pp. 227–250. Springer (2014)

6. Brier, E., Khazaei, S., Meier, W., Peyrin, T.: Linearization framework for collision attacks: Application to CubeHash and MD6. In: Matsui, M. (ed.) Advances in Cryptology – ASIACRYPT 2009. LNCS, vol. 5912, pp. 560–577. Springer (2009)
7. Daemen, J., Rijmen, V.: AES and the wide trail design strategy. In: Knudsen, L.R. (ed.) Advances in Cryptology – EUROCRYPT 2002. LNCS, vol. 2332, pp. 108–109. Springer (2002)
8. De Cannière, C., Rechberger, C.: Finding SHA-1 characteristics: General results and applications. In: Lai, X., Chen, K. (eds.) Advances in Cryptology – ASIACRYPT 2006. LNCS, vol. 4284, pp. 1–20. Springer (2006)
9. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon. Submission to the CAESAR competition: `http://competitions.cr.yp.to/round1/asconv1.pdf` (2014)
10. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Cryptanalysis of Ascon. In: Nyberg, K. (ed.) Topics in Cryptology – CT-RSA 2015. LNCS, vol. 9048, pp. 371–387. Springer (2015)
11. Huang, T., Tjuawinata, I., Wu, H.: Differential-linear cryptanalysis of ICEPOLE. In: Leander, G. (ed.) Fast Software Encryption – FSE 2015. LNCS, vol. 9054, pp. 243–263. Springer (2015)
12. Indesteege, S., Preneel, B.: Practical collisions for EnRUPT. In: Dunkelman, O. (ed.) Fast Software Encryption – FSE 2009. LNCS, vol. 5665, pp. 246–259. Springer (2009)
13. Kavun, E.B., Lauridsen, M.M., Leander, G., Rechberger, C., Schwabe, P., Yalçın, T.: Prøst. Submission to the CAESAR competition: `http://competitions.cr.yp.to/round1/proestv11.pdf` (2014)
14. Leurent, G.: Construction of differential characteristics in ARX designs: Application to Skein. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013. LNCS, vol. 8042, pp. 241–258. Springer (2013)
15. Matsui, M.: Linear cryptoanalysis method for DES cipher. In: Helleseth, T. (ed.) Advances in Cryptology – EUROCRYPT '93. LNCS, vol. 765, pp. 386–397. Springer (1993)
16. Mendel, F., Nad, T., Schläffer, M.: Finding SHA-2 characteristics: Searching through a minefield of contradictions. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology – ASIACRYPT 2011. LNCS, vol. 7073, pp. 288–307. Springer (2011)
17. Mendel, F., Nad, T., Schläffer, M.: Improving local collisions: New attacks on reduced SHA-256. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology – EUROCRYPT 2013. LNCS, vol. 7881, pp. 262–278. Springer (2013)
18. Minaud, B.: Linear biases in AEGIS keystream. In: Joux, A., Youssef, A.M. (eds.) Selected Areas in Cryptography – SAC 2014. LNCS, vol. 8781, pp. 290–305. Springer (2014)
19. Morawiecki, P., Gaj, K., Homsirikamol, E., Matusiewicz, K., Pieprzyk, J., Rogawski, M., Srebrny, M., Wójcik, M.: ICEPOLE. Submission to the CAESAR competition: `http://competitions.cr.yp.to/round1/icepolev1.pdf` (2014)
20. Mouha, N., Preneel, B.: Towards finding optimal differential characteristics for ARX: Application to Salsa20. IACR Cryptology ePrint Archive, Report 2013/328 (2013), `http://eprint.iacr.org/2013/328`
21. Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Wu, C., Yung, M., Lin, D. (eds.) Information Security and Cryptology – Inscrypt 2011. LNCS, vol. 7537, pp. 57–76. Springer (2011)

22. Sasaki, Y., Todo, Y., Aoki, K., Naito, Y., Sugawara, T., Murakami, Y., Matsui, M., Hirose, S.: Minalpher. Submission to the CAESAR competition: http://competitions.cr.yp.to/round1/minalpherv1.pdf (2014)
23. Schläffer, M., Oswald, E.: Searching for differential paths in MD4. In: Robshaw, M.J.B. (ed.) Fast Software Encryption – FSE 2006. LNCS, vol. 4047, pp. 242–261. Springer (2006)
24. Sun, S., Hu, L., Wang, M., Wang, P., Qiao, K., Ma, X., Shi, D., Song, L.: Automatic enumeration of (related-key) differential and linear characteristics with predefined properties and its applications. IACR Cryptology ePrint Archive, Report 2014/747 (2014), http://eprint.iacr.org/2014/747
25. Tardy-Corfdir, A., Gilbert, H.: A known plaintext attack of FEAL-4 and FEAL-6. In: Feigenbaum, J. (ed.) Advances in Cryptology – CRYPTO '91. LNCS, vol. 576, pp. 172–181. Springer (1991)
26. The CAESAR committee: CAESAR: Competition for authenticated encryption: Security, applicability, and robustness (2014), http://competitions.cr.yp.to/caesar.html

A Linear characteristics

In the appendix, we include the linear characteristics for the results of Sect. 4: KEYAK (Type I in Table 2, Type II in Table 3), ICEPOLE (Type I in Table 6, Type II in Table 7, Type III in Table 8), ASCON (Type I in Table 4, Type II in Table 5), Minalpher (Type I in Table 10), and PRØST (Type I in Table 11). The Type-III linear characteristic conforming to the capacity of ICEPOLE-256a is given in Table 9.

Each table gives the linear input mask (numbered 0) and the linear mask at the output of each round (numbered 1 to r) in a hexadecimal notation (with `-` = `0`). The representation of our linear characteristics conform to the representation of the state of the corresponding reference implementations. Usually, the internal state in the reference implementations is pictured as a 1- or 2-dimensional array of words. The top left word pictured in one state of our tables corresponds to the word with the lowest index, the table word on the right bottom to the array word with the highest index. Additionally, for Type-II and Type-III characteristics, the bits with additional external constraints (i.e., uncontrollable input bits and unobservable output bits) are highlighted as `-`.

Table 2: Type-I linear characteristics for r rounds of the KEYAK permutation.

(a) $r = 3$: 13 active S-boxes, bias 2^{-14}.

Round	State				
	-----1----------	-----8----------	------8---------	----------------	----1-----------
	-----1----------	-----8----------	------8---------	----------------	----1-----------
0	-----1----------	-----8----------	------8---------	----------------	----1-----------
	----11----------	-----9----------	------8---------	----------------	----1-----------
	-----1----------	-----8----------	------8---------	----------------	----1-----------
	----------------	----------------	----------------	----------------	----------------
	----------------	----------------	----------------	----------2-----	----------------
1	----------------	----------------	----------------	----------------	----------------
	----------------	----------------	----------------	----------------	----------------
	----------------	----------------	----------------	----------2-----	----------------
	----------------	----------------	----------------	----------------	----------------
	----------------	----------------	----------------	----------------	----------------
2	----------------	----------------	----------------	----------------	----------------
	-----------2---	----------------	----------------	----------------	-----------2---
	----------------	------------1---	----------------	----------------	----------------
	----------------	-1--------------	----------------	----------------	---------8------
	----------------	-------2--------	----------------	-2--------------	----------------
3	-----------2---	----------------	----------------	----------------	----------------
	-----1----------	----------------	----------4-----	----------------	----------------
	----------------	----------------	--1-------------	--4-------------	----------------

(b) $r = 4$: 33 active S-boxes, bias 2^{-34}.

Round	State
0	-----58---4-1--- 1---8428-2----4- 4----64-1---5--- -8--421-4-8-3--1 -----11--4---88- -----48---4-1-8- 9---84a8-2----4- 4----26-1---5--- -81-421-4-8-2--1 -----11--48-288- -----49---4-1--- 1---84a822----4- 5----24-1---5--- -8--421-4-8-6--1 -----11--4--2881 -----48---4-1--- 1---84a8-2----4- 4----24-1---5--- -8--421-4-8-2--1 -----11--4--288- -----48--44-1--- 1---8-a8-2----4- 4---824-1---5--- -81-421-4-8-2--1 -----11-44--288-
1	-----1---------- -----8---------- ------8--------- ---------------- ----1----------- -----1---------- -----8---------- ------8--------- ---------------- ----1----------- -----1---------- -----8---------- ------8--------- ---------------- ----1----------- ----11---------- -----9---------- ------8--------- ---------------- ----1----------- -----1---------- -----8---------- ------8--------- ---------------- ----1-----------
2	---------------- ---------------- ---------------- ----------2----- ---------------- ---------------- ---------------- ---------------- ---------------- ---------------- ---------------- ---------------- ---------------- ---------------- ---------------- ---------------- ---------------- ---------------- ----------2----- ----------------
3	---------------- ---------------- ---------------- ---------------- ---------------- ---------------- ---------------- ---------------- ---------------- ---------------- ----------2--- ---------------- ---------------- ---------------- ----------2--- ---------------- ------------1--- ---------------- ---------------- ----------------
4	---------------- -1-------------- ---------------- ---------------- ---------8------ ---------------- -------2-------- ---------------- -2-------------- ---------------- ----------2--- ---------------- ---------------- ---------------- ---------------- -----1---------- ---------------- ----------4----- ---------------- ---------------- ---------------- ---------------- --1------------ --4------------ ----------------

Table 3: Type-II linear characteristics for r rounds of the KEYAK permutation.

(a) $r = 3$: 12 active S-boxes, bias 2^{-13} (without last-round S-boxes).

Round	State
0	----------8---3 8--------4-8---- ------1----4---- ------1----4---4 -------8-8-----1 ---------8-8---2 8--------4------ ------1----4---- ----------4---4 -------8-8-----1 ----------8---2 8--------4-8---- ------1----4---- ----------4---4 -------8-8-----1 ----------8---2 8--------4-8---- ------1----4---- ----------4---4 -------8-8-----1 ----------8---2 8--------4-8---- ------1----c---- ----------4---4 -------8-8-----5
1	---------------1 8--------------- ---------------- ---------------- ----------1---- ---------------1 ---------------- ---------------- ---------------- ----------1---- ---------------- ---------------- ---------------- ---------------- ---------------- ---------------- 8--------------- ---------------- ---------------- ---------------- ---------------- ---------------- ---------------- ---------------- ----------------
2	---------------1 ---------------- ---------------- ---------------- ---------------- ---------------- ------1--------- ---------------- ----1----------- ----1----------- ---------------1 ---------------- ---------------- ---------------- ---------------- -----8---------- ------1--------- ---------------- ---------------- ---------------- ---------------- ---------------- ---------------- ---------------- ----------------
3	-----8---------1 ----------1---- ---------------- ---------------2 ---------------- ------------1-- ---------------1 ----4----------8 ----------2---- ---------------- ---------------- ---------------- ----------2- ---------------- 2--------------- ---------------- ------------8--- ---------------- ---------------- ------1--------- ---------------- ---------------- ---------------- ---------------- ----------------

(b) $r = 4$: 43 active S-boxes, bias 2^{-49} (without last-round S-boxes).

Round	State				
0	4---86-8a-2--c28	2---c21c-3--2-24	2-9-83-4481-22--	1--1e--86-4-1-52	--494418-6-8-84-
	4---8618a-2--468	6---c21c13--2424	2-9-8314481--2--	1--96--86-4-1-52	--484418-648-84-
	4---82-8a-2--428	2---c214-3--2424	2-9-83-4481--2-4	1--16--86-5-1-52	--484418-6-8-8--
	4---86--a-2--428	2---c21c83--2424	--9-83-4481--2--	1-816--86-4-1-52	--484418-6-8-85-
	4---86-8a62--428	2---c41c-3--2424	2-9-43-4481--2--	1-196--86-4-1-52	--48441866-8-84-
1	----------8--	-----4-------9--	----2----------	----2-------1---	----18------1---
	----------8--	-----4-------8--	----2----------	----------1---	----18------1---
	----------8--	-----4-------8--	----2----------	----------1---	----18------1---
	-----8------18--	-----4-------9--	----2----------	----------1---	----18----------
	----------8--	-----4-------8--	----2----------	----------1---	----18------1---
2	---------------	---------------	---------------	---------------	---------------
	----------2--	---------------	---------------	--2------------	---------------
	----------2--	---------------	---------------	----------1-----	---------------
	---------------	---------------	---------------	---------------	---------------
	---------------	---------------	---------------	--2-------1-----	---------------
3	---------------	---------------	---------------	---------------	---------------
	---------------	------------1---	------------1---	---------------	------------1---
	----2----------	----2----------	----2----------	----2----------	----2----------
	---------------	----2-------1---	----2-------1---	----2----------	------------1---
	---------------	----1----------	---------------	---------------	---------------
4	---------------	-1-------1------	---------------	---------------4	-8--------------
	---------------	-------1-------2	---1------------	-2-------6------	---------------
	----2----------	----------4----	--------------4-	--2-------1-----	---------------
	---------8---1--	---------------	--c------8------	1--------8------	---------------
	---------------	---------------	---------------	---------------	---------------

Table 4: Type-I linear characteristics for r rounds of the Ascon permutation.

(a) $r = 3$: 13 active S-boxes, bias 2^{-15}.

Round	State				
0	----------4---	-4-8-2---1---48-	-4-8-2---1---481	----------44---	----------4----
1	---------------	---------------	---------------	-----2--------81	-----2--------81
2	---------------	---------------	---------------	---------------	--------------1
3	---------------	---------------	fc1-c53d1c96ecd5	a423953bbde612d6	---------------

(b) $r = 4$: 43 active S-boxes, bias 2^{-50}.

Round	State				
0	---4-4-----1--1-	--1-3-2--4-8----	--1-3-28-c-8-2--	---4-4-4---11-1-	-------4----1---
1	---------------	---------------	---------------	-------8-8---2--	-------8-8---2--
2	8-4----8--------	---------------	---------------	---------------	---------------
3	-------8--------	---------------	2fc-----218a7a39	---------------	---------------
4	e4b766afe-8629e8	766572e7eeeed3b9	e4b766afe-8629e8	ef3-96b5211ca9dd	-3256--4-4df2ab1

(c) $r = 5$: 67 active S-boxes, bias 2^{-94}.

Round	State				
0	---4---4---11-1-	--1-3-2--4-8----	--1-3-28-c-8-2--	---4-4-4---11-1-	-----4----------
1	----------------	----------------	----------------	-------8-8---2--	-------8-8---2--
2	8-4----8--------	----------------	----------------	----------------	----------------
3	-------8--------	----------------	2fc-----218a7a39	----------------	----------------
4	-9--2-9-2-12----	844--8d--1b21-13	2d4-------821-21	254------------4	-c4-4-1--1--1-24
5	7d-1-1cda6c3f826	d94bc8bbde-4c573	6ecd5e-a7abb7629	56ef77b8fa68bc84	fe23b6e797a3fc66

Table 5: Type-II linear characteristics for r rounds of the Ascon-128 permutation.

(a) $r = 2$: 6 active S-boxes, bias 2^{-8}.

Round	State				
0	----------------	----------2-4--	----------2-4-1	-----2--------8-	-----2--------8-
1	----------------	----------------	----------------	---------------1	---------------1
2	9224b6d24b6eda49	----------------	----------------	----------------	----------------

(b) $r = 3$: 23 active S-boxes, bias 2^{-30}.

Round	State				
0	-1-2-------2----	-4-8-2---181-842	-4-8-2---185-8c2	-------------4-1	-1-2--------4---
1	----------------	----------2-4--	----------2-4-1	-----2--------81	-----2--------81
2	----------------	----------------	----------------	---------------1	---------------1
3	9224b6d24b6eda49	----------------	----------------	----------------	----------------

(c) $r = 4$: 61 active S-boxes, bias 2^{-83}.

Round	State				
0	---d234--4b-2d4a	-21-8--243-a----	-61-84--634a----	-1-2--------4--4	89-521----a-615a
1	-5-2--------44--	-4-a-2---181-cc2	-4-8-2---187-c43	---2------------	-1-2------------
2	----------------	----------2-4--	----------2-4-1	-----2--------8-	-----2--------81
3	----------------	----------------	----------------	---------------1	---------------1
4	9224b6d24b6eda49	----------------	----------------	----------------	----------------

Table 6: Type-I linear characteristics for r rounds of the ICEPOLE permutation.

(a) $r = 5$: 38 active S-boxes, bias $2^{-55.08}$.

Round	State				
0	2-------8-84-4--	-6-----9-----4--	-8-4-------a22--	-2--2--1--8-----	d2-----8--------
	5------1--------	-c------8-86----	-2-42--8--8-----	82---------822--	28-----1---2----
	34-----------4--	-------1--842---	-e-----8---2-4--	4--42-----88-2--	82-----18---24--
	94-----18-------	2--4------84----	-------9----2---	-8---------a-2--	4--42--1--8-----
1	---4------------	1--4------------	----------------	------4---------	----------------
	18--------------	----------------	1---------------	-84-------------	--4---4---------
	1-4-------------	----------------	----------------	-8--------------	-84---4---------
	-844------------	1---------------	---4------------	--4-------------	-8-4--4---------
2	-1--------------	----------------	---8------------	----------------	----------------
	----------------	-1--------------	----------------	---8------------	----------------
	----------------	-1--------------	----------------	---8------------	----------------
	----------------	-1--------------	----------------	---8------------	----------------
3	----------------	----------------	----------------	----------------	----------------
	----------------	----------------	----------------	--------1-------	--------1-------
	----------------	----------------	----------------	----------------	----------------
	----------------	----------------	--------1-------	----------------	--------1-------
4	-------------2--	----4-----------	----4-----------	----------------	----4--------2--
	----------------	----------------	----------------	----------------	----------------
	----------------	----------------	----------------	----------------	----------------
	----------------	----------------	----------------	----------------	----------------
5	---------8-----1	----------------	----------------	----------------	---------8------
	2---------------	2---------------	----------------	----2-----------	----2-----------
	--22--------1---	---2--------1--8	---------------8	----------------	--2-------------
	------------8---	------------8---	----------8-----	----------8-----	----------------

(b) $r = 6$: 104 active S-boxes, bias $2^{-126.32}$.

Round	State				
0	2-------8-84-4--	-6----------42-	-8-4----1--822--	-2--2-81--82----	52----------
	5------------2-	-c------8-84----	-2-42---1-8---2-	-2---------822--	28----81---2----
	34----------42-	----------842---	-e----8----2-4--	4--42---1-88-2--	-2-----18---24--
	14------8-------	2--4----1-84--2-	----------2---	-8----8----a-2--	4--42--11-8-----
1	---4----------	1--4----------	----------	-----4---------	----------
	18----------	----------	1----------	-84----------	--4---4---------
	1-4----------	----------	----------	-8----------	-84---4---------
	-844----------	1----------	---4----------	--4----------	-8-4--4---------
2	-1----------	----------	---8----------	----------	----------
	----------	-1----------	----------	---8----------	----------
	----------	-1----------	----------	---8----------	----------
	----------	-1----------	----------	---8----------	----------
3	----------	----------	----------	----------	----------
	----------	----------	----------	--------1-------	--------1-------
	----------	----------	----------	----------	----------
	----------	----------	--------1-------	----------	--------1-------
4	----------2--	----4----------	----4----------	----------	----4--------2--
	----------	----------	----------	----------	----------
	----------	----------	----------	----------	----------
	----------	----------	----------	----------	----------
5	---------8------	----------1	----------1	----------	---------8-----1
	2----------	2----------	----------	----2----------	----2----------
	---2--------1---	--2----------8	--2---------1---	----------8	--2----------
	----------8-8---	----------	----------8-8---	----------8---	----------8-8---
6	-----4-2---4---8	2------2--8-1--8	24--11----8-1-4-	----1--1------44	----24-14------4
	1----1--4-4-9-51	9-8-8---4---d--1	--c-8-------6---	--49--8-----2---	---8-18---4---4-
	-81-8---46----c9	--1-----442--4-d	---4--------1--4	---c-8------1---	-8-888-2-2----8-
	-----9-22------8	-88-----2------8	--8--2--44-----1	--2-----44----81	--28-9-------28-

Table 7: Type-II linear characteristics for r rounds of the ICEPOLE permutation.

(a) $r = 4$: 22 active S-boxes, bias $2^{-30.42}$.

Round	State				
0	-4a-------8-8---	1--1-----------	----4----4--81--	------48--------	-941-----------
	198-------------	----------8-81--	1--14----------	---------4--8---	-46---48-----1--
	156-------------	--8------4--8---	---1---8-----1--	-8--4-------8---	------4--48-----
	-1--------8-----	142-4-------8---	---1-----4------	--4----8----81--	-88-4-4---------
1	-1-------------	---------------	---8-----------	---------------	---------------
	---------------	-1-------------	---------------	---8-----------	---------------
	---------------	-1-------------	---------------	---8-----------	---------------
	---------------	-1-------------	---------------	---8-----------	---------------
2	---------------	---------------	---------------	---------------	---------------
	---------------	---------------	---------------	--------1-------	--------1-------
	---------------	---------------	---------------	---------------	---------------
	---------------	---------------	--------1-------	---------------	--------1-------
3	------------2--	----4----------	----4----------	---------------	----4--------2--
	---------------	---------------	---------------	---------------	---------------
	---------------	---------------	---------------	---------------	---------------
	---------------	---------------	---------------	---------------	---------------
4	---------------1	---------------	---------8------	---------------	---------------
	2--------------	2--------------	---------------	----2----------	---------------
	---2--------1---	---2--------1--8	--2----------8	---------------	---------------
	------------8---	------------8---	----------8-----	----------8-----	---------------

(b) $r = 5$: 38 active S-boxes, bias $2^{-59.49}$

Round	State				
0	2-------8-84-4--	-6-----9-----4--	-8-4-------a22--	-2--2--1--8-----	d2-----8--------
	5------1--------	-c------8-86----	-2-42--8--8-----	82---------822--	28-----1---2----
	34----------4--	-------1--842---	-e-----8---2-4--	4--42-----88-2--	82-----18---24--
	94-----18-------	2--4------84----	-------9----2---	-8---------a-2--	4--42--1--8-----
1	---4----------	1--4----------	----------	------4--------	----------
	18----------	----------	1----------	-84----------	--4---4--------
	1-4----------	----------	----------	-8----------	-84---4--------
	-844----------	1----------	---4----------	--4----------	-8-4--4--------
2	-1----------	----------	---8----------	----------	----------
	----------	-1----------	----------	---8----------	----------
	----------	-1----------	----------	---8----------	----------
	----------	-1----------	----------	---8----------	----------
3	----------	----------	----------	----------	----------
	----------	----------	----------	-------1-------	-------1-------
	----------	----------	----------	----------	----------
	----------	----------	--------1-------	----------	-------1-------
4	----------2--	----4----------	----4----------	----------	----4-------2--
	----------	----------	----------	----------	----------
	----------	----------	----------	----------	----------
	----------	----------	----------	----------	----------
5	---------8-----1	----------	----------	----------	----------
	2----------	2----------	----------	----2----------	----------
	--22--------1---	---2--------1--8	----------8	----------	----------
	----------8---	----------8---	---------8-----	---------8-----	----------

Table 8: Type-III linear characteristics for r rounds of the ICEPOLE permutation.

(a) $r = 3$: 10 active S-boxes, bias $2^{-16.66}$.

Round	State				
0	-----2----------	-------2--4-----	------1---------	----------	----------
	----------4-----	-----2-2--------	----------4-----	------1---------	----------
	-------2--4-----	-----2----------	-------2--------	------1---------	----------
	-------2--------	-----2----4-----	----------	------1---------	----------
1	----------	----------	----------	----------2---	----------
	----------	----------	----------	----------2---	----------
	----------	----------	----------	----------	----------2---
	----------	----------	----------2---	----------	----------
2	-4----------	----------	----------	----------	-4----------
	----------	----------	----------	----------	----------
	----------	----------	----------	----------	----------
	----------	----------	----------	----------	----------
3	----------	----------	---2----------	---2----------	----------
	----------	----------	----------	--------4-------	----------
	2----------	2----------	------4---------	----------	----------
	----------1	----------1	----------	----------	----------

(b) $r = 4$: 22 active S-boxes, bias $2^{-43.25}$.

Round	State				
0	----------1-8---	1---2-------1-82	------------8---	----------------	----------------
	1---2----------2	----------1-9-8-	1--------------2	------------8---	----------------
	1-----------1-82	----2-----1-8---	-------------1-8-	------------8---	----------------
	----2-------1-8-	1---------1-8--2	----2-----------	------------8---	----------------
1	-1--------------	----------------	---8------------	----------------	----------------
	----------------	-1--------------	----------------	---8------------	----------------
	----------------	-1--------------	----------------	---8------------	----------------
	----------------	-1--------------	----------------	---8------------	----------------
2	----------------	----------------	----------------	----------------	----------------
	----------------	----------------	----------------	--------1-------	--------1-------
	----------------	----------------	----------------	----------------	----------------
	----------------	----------------	--------1-------	----------------	--------1-------
3	-------------2--	----4-----------	----4-----------	----------------	----4--------2--
	----------------	----------------	----------------	----------------	----------------
	----------------	----------------	----------------	----------------	----------------
	----------------	----------------	----------------	----------------	----------------
4	---------8-----1	----------------	----------------	----------------	----------------
	2---------------	2---------------	----------------	----2-----------	----------------
	--22--------1---	---2--------1--8	---------------8	----------------	----------------
	------------8---	------------8---	----------8-----	----------8-----	----------------

(c) $r = 5$: 42 active S-boxes, bias $2^{-87.08}$.

Round	State				
0	----------8---2-	---4---a484-8-9-	--3--------4-22-	--1-------------	----------------
	--24---9484---9-	--2----2--8-8-2-	---4----4-4---1-	--2----1---4-22-	----------------
	--24---34-4-8-1-	--3----9-88---a-	--2----24---8---	--2--------4-2--	----------------
	-------a48--8-a-	---4---1--c---1-	--2----948----a-	--2--------4-2--	----------------
1	---4------------	---4------------	-----2------2---	-----2-24-------	-8--------------
	----------------	------------2---	----------------	-8--------------	------24---2---
	----------------	----------------	--------4---2---	----------------	-8-----2--------
	-8-4------------	-----2----------	---4------------	--------4---2---	---4---2--------
2	-1--------------	----------------	---8------------	----------------	----------------
	----------------	-1--------------	----------------	---8------------	----------------
	----------------	-1--------------	----------------	---8------------	----------------
	----------------	-1--------------	----------------	---8------------	----------------
3	----------------	----------------	----------------	----------------	----------------
	----------------	----------------	----------------	--------1-------	--------1-------
	----------------	----------------	----------------	----------------	----------------
	----------------	----------------	--------1-------	----------------	--------1-------
4	-------------2--	----4-----------	----4-----------	----------------	----4--------2--
	----------------	----------------	----------------	----------------	----------------
	----------------	----------------	----------------	----------------	----------------
	----------------	----------------	----------------	----------------	----------------
5	---------------1	----------------	----------------	---------8------	----------------
	2---------------	2---------------	----------------	----2-----------	----------------
	--22--------1---	---2--------1--8	---------------8	----------------	----------------
	------------8---	------------8---	----------8-----	----------8-----	----------------

Table 9: Type-III linear characteristics for r rounds of the ICEPOLE-256a permutation.

(a) $r = 5$: 42 active S-boxes, bias $2^{-89.49}$.

Round	State				
0	---------8---2-	---41--a494-8-9-	--1----------2-	--1----------	----------
	--241--9494---9-	--2----2--8-8-2-	---4----4-4---1-	-------1------2-	----------
	--24---34-4-8-1-	--3-1--9-98---a-	--2----24---8---	----------	----------
	----1--a49--8-a-	---4---1--c---1-	--2-1--949----a-	----------	----------
1	---4----------	---4----------	-----2------2---	-----2-24-------	-8----------
	----------	----------2---	----------	-8----------	-------24---2---
	----------	----------	--------4---2---	----------	-8-----2--------
	-8-4----------	-----2----------	---4----------	--------4---2---	---4---2--------
2	-1----------	----------	---8----------	----------	----------
	----------	-1----------	----------	---8----------	----------
	----------	-1----------	----------	---8----------	----------
	----------	-1----------	----------	---8----------	----------
3	----------	----------	----------	----------	----------
	----------	----------	----------	--------1-------	--------1-------
	----------	----------	----------	----------	----------
	----------	----------	--------1-------	----------	--------1-------
4	----------2--	----4----------	----4----------	----------	----4--------2--
	----------	----------	----------	----------	----------
	----------	----------	----------	----------	----------
	----------	----------	----------	----------	----------
5	----------1	----------	----------	---------8------	----------
	2----------	2----------	----------	----2----------	----------
	---2--------1---	--22--------1--8	----------8	----------	----------
	----------8---	----------8---	----------8-----	----------	----------

Table 10: Type-I linear characteristics for r rounds of the Minalpher permutation.

(a) $r = 4$: 22 active S-boxes, bias 2^{-23}

Round	State			
0	---------1------	------------1-1-	---------1------	--------1--1----
1	-------1--------	----------------	--1-------------	----------------
2	-1-------1------	----------------	-1-------1------	----------------
3	--1----1--1-----	----------1----1	--1----1-------1	----------1----1
4	1--1--1--1--1---	1--111------11--	1--1111--1---1--	-----11-----11--

(b) $r = 5$: 41 active S-boxes, bias 2^{-42}.

Round	State			
0	---1----------	---------1----1-	-------1-1------	-1------1-----1-
1	----------	-----1----------	--1-------------	------------1--
2	---------1------	----------	-1-------------	---------1------
3	---1---1--------	----------1	---1---1-------1	---1----------1
4	---111-------1--	1--1----11------	1--111--11---1--	11--1---1-------
5	---1-111--1-1-11	1-111--11--1-1--	1-----111--1111-	1-111--11-111---

(c) $r = 6$: 58 active S-boxes and bias 2^{-62}.

Round	State			
0	------1-8--1-1--	--1-----1--1----	---1----1-1---11	--1--1------1--1
1	-1-1--3---------	------2-------1-	---11-----------	-1----2-1-----1-
2	1-------1----1--	--1-------------	1----1--1-------	-----1----1-----
3	----------1----	------1---------	----------1--1-	----------
4	1----1-------1--	-------------1--	1------------1--	-----1----------
5	-------1---1--11	----------1--11	------11--------	----------1---1
6	1-11-1-1-1------	1-11-----1--11--	-111-1-1----11--	11--1--1-1--11--

Table 11: Type-I linear characteristics for r rounds of the PRØST-256 permutation.

(a) $r = 4$: 25 active S-boxes, bias 2^{-26}

Round	State
0	-4-----1 -------- -------- -------1 -------- -4-----1 -------- -------- -------- -4-----1 -------- -------- -4-----1 -------- -------- -4------
1	-------1 -------1 -------- -------- -------- -------- -------- -------- -------- -------- -------- -------- -------1 -------1 -------- --------
2	8------- -------- -------- -------- -----1-- -------- -----1-- -------- -------- -------- -------- -------- -------2 -------2 -------- --------
3	-----1-2 -------2 8------2 -----1-2 ------1- 28------ -------- -------- 1-28---- ---8---- 1-2----- 1-2----- ------a- -------- ----4--- ------8-
4	48142-9- -81----8 14-42-41 -81-2-8- -8-12-8- -8--22-- --419--- 2--18-b8 --1--48- -8--488e -a----24 -21-2--- 2-4-8325 7-5--32- 5-1--241 -----2-1

(b) $r = 5$: 41 active S-boxes, bias 2^{-42}

Round	State
0	-11----- -1--4--1 -------- -------- ----4--1 ----4--- -11----- --1----- --1----1 -11----- -1--4--- ----4--1 -1--4--- --1----1 -------- --------
1	-------- --1----- --1----- -------- --1----- -------- -------- -------- --1----- --1----- -------- -------- -------- --1----- --1----- --------
2	-------- -------- -------- -------- -------- -------- -------- -------- -------- -------- -------- -------- --2----- -------- -------- --------
3	--2----- -------- -------- -------- ---2---- ---2---- -------- -------- -------- -----2-- -------- -------- -------- -------- -8------ --------
4	-4-1---- -----1-- -411-1-- -4-1---- -2------ 2--2---- -------8 -2------ -88----2 -88----- --8----- --8-8--2 1-4----- ---4---- -----4-- --------
5	12d--5-8 -c858--- 26-384-2 -2---4-a 81ed1-1- -1c458-- 2-8-4-4- -2a12-1- 4-a-e-1- --23643- 1-8-294- 4-8-e-18 -4-14--2 a-8----8 14a-42-e -54-4-83

(c) $r = 6$: 105 active S-boxes, bias 2^{-107}

Round	State
0	---214b5 c--8-682 85---a-2 -------5 c--8-222 ----148- ----1-84 -5-2-a31 c----c22 85---825 -5-812-4 ---a--92 45-21415 -4-8-82- 8-----82 41--16--
1	------81 ------21 4-----2- 4-----8- 4-----2- ------8- -------- ------21 ------81 4-----8- 4------- ------2- 4-----a1 -------- -------- 4------1
2	------4- ------4- -------- -------- -------- -------- ------4- -------- -------- ------4- ------4- -------- -------- -------- ------4- --------
3	-------- -------- -------- -------- -------4 -------- -------- -------- -------- -------- -------- -------- -------- -------- -------- --------
4	-------- -------2 -------- -------- -------- -------- -----4-- -------- -----1-- -------- -------- -------- -------8 -------8 -------- --------
5	-----5-8 -----1-8 -------8 -----4-a ------5- 8------- 2------- -------- 4-8----- --2----- 5-a----- 4-8----- -----2-- ----4-8- ---1---- -----2--
6	3-4-828- 2-4-2269 481-81-4 3-4-8-8- 2--1-23- --4-88-- 2143d-1- a--7--d- -84-16-c 2---423- 2-112-18 --418-88 a142-6b4 81---6a1 4----a-1 2-4-83-4

(d) $r = 7$: 169 active S-boxes, bias 2^{-175}

Round	State
0	---8725- 1-2-1--9 14---88e -----21- 1-2---8d ----3244 -----21- -4-84882 1---18cd -4-8481a -42---12 1-2-224- -4-87-52 --2--884 1----28c -4--1--3
1	-----284 -------- -------- -----2-1 -------1 ------8- -------4 -----2-4 -----2-- -----2-1 -------1 ------84 -----285 -------- -------- ------81
2	-----1-- -----1-- -------- -------- -------- -------- -----1-- -------- -------- -------- -------- -----1-- -----1-- -----1-- -------- --------
3	-------- -------- -------- -------- -------- -------- -------- -------- 1------- 1------- 1------- 1------- -------- -------- -------- --------
4	-8------ -8------ -------- -------- ------1- -------- -------- -------- -------- -------- -------4 -------- -------- 2------- -------- --------
5	28-----4 28----14 -------- -8------ 42------ -------- -28----1 --8----- -1------ ---28--- --42---- ----8--- -----4-- -----5-2 -------a -------8
6	11c--2-6 a-8--281 -421-2-9 11--42-f 428--623 8--51--8 42-1156- 428--c2- b-81-6d2 --2--18a c--1448- a--1--8- d-85-82e c2----3a 44-----c 4184-a-c
7	12a4179c 6-a54fc6 c46-4f2- 24-419e6 7a521179 63-65-51 1-5--4-b 874a444- 56565c55 2847144- 926a6a51 44776a55 d9-2-2a8 a1411f7c 61441299 694-86e9

Forgery Attacks on Round-Reduced ICEPOLE-128

Publication Data

Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. "Forgery Attacks on Round-Reduced ICEPOLE-128". In: *Selected Areas in Cryptography – SAC 2015*. Ed. by Orr Dunkelman and Liam Keliher. Vol. 9566. LNCS. Springer, 2016, pp. 479–492. URL: https://doi.org/10.1007/978-3-319-31301-6_27

The appended paper is an author-created version available at https://eprint.iacr.org/2015/392.

Contributions

- One of the main authors.

Forgery Attacks on Round-Reduced ICEPOLE-128

Christoph Dobraunig, Maria Eichlseder, and Florian Mendel

IAIK, Graz University of Technology, Austria
christoph.dobraunig@iaik.tugraz.at

Abstract. ICEPOLE is a family of authenticated encryptions schemes submitted to the ongoing CAESAR competition and in addition presented at CHES 2014. To justify the use of ICEPOLE, or to point out potential weaknesses, third-party cryptanalysis is needed. In this work, we evaluate the resistance of ICEPOLE-128 against forgery attacks. By using differential cryptanalysis, we are able to create forgeries from a known ciphertext-tag pair with a probability of $2^{-60.3}$ for a round-reduced version of ICEPOLE-128, where the last permutation is reduced to 4 (out of 6) rounds. This is a noticeable advantage compared to simply guessing the right tag, which works with a probability of 2^{-128}. As far as we know, this is the first published attack in a nonce-respecting setting on round-reduced versions of ICEPOLE-128.

Keywords: CAESAR, ICEPOLE, forgery, differential cryptanalysis

1 Introduction

ICEPOLE is a family of authenticated encryption schemes, which has been presented at CHES 2014 [17] and submitted to CAESAR [16]. CAESAR [18] is an open cryptographic competition aiming to find a suitable portfolio of authenticated encryption algorithms for many use cases. For the first round, 57 candidates have been submitted. Due to the open nature of CAESAR, those candidates have different design goals ranging from high-speed software designs to designs suitable for compact hardware implementations. This makes comparison of the submitted ciphers difficult, which is nevertheless necessary to determine the ciphers for the next rounds. However, all designs have one goal in common: security. Thus, as much security analysis as possible is needed to sort out weak CAESAR candidates and get insight in the security of the others.

The goal of authenticated encryption is to provide confidentiality and authenticity. Our attacks focus solely on the authenticity in a forgery attack. The goal is to manipulate known ciphertext-tag pairs in a way such that they are valid with a certain probability. For ICEPOLE-128, the intended number of bits of security with respect to authenticity is 128. Therefore we consider only attacks with a success probability above the generic 2^{-128} applicable.

The method of choice for our attacks is differential cryptanalysis [7]. To create forgeries, we need differential characteristics which hold with a high probability

The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-31301-6_27

and fulfill certain constraints explained later. By using this technique, we are able to attack versions of ICEPOLE-128 where the last permutation is reduced to 4 out of 6 rounds. As far as we know, no forgery attacks have been performed on round-reduced versions of ICEPOLE. In addition, we have analyzed the main building block of ICEPOLE, its permutation. We were able to improve the results of the designers regarding high-probability characteristics for the ICEPOLE permutation without any additional constrains. Our results are summarized in Table 1. Note that we have verified the forgery for 3 rounds practically by using the reference implementation of ICEPOLE-128 submitted to CAESAR.

Table 1. Results for ICEPOLE.

	Type	Rounds	Probability
ICEPOLE-128	forgery	3/6	$2^{-14.8}$
		4/6	$2^{-60.3}$
ICEPOLE permutation	differential characteristic	5	$2^{-104.5}$
		6	$2^{-258.3}$

Related work. In the submission document [16], the designers bounded the minimum number of active S-boxes in a differential characteristic for the ICEPOLE permutation. They are able to show that for 3 rounds, the minimum number of active S-boxes is 9, and that there are no characteristics with 13 or fewer active S-boxes for 4 rounds. In addition, Morawiecki et al. [16] heuristically searched for differential characteristics. For 5 rounds, their best published differential characteristic has a probability of $2^{-186.2}$, and for 6 rounds $2^{-555.3}$.

Recently, Huang et al. [13] presented state-recovery attacks on ICEPOLE in a nonce-misuse scenario. They show that in this scenario, the internal state of ICEPOLE-128 and ICEPOLE-128a can be recovered with complexity 2^{46}, and the internal state of ICEPOLE-256a with complexity 2^{60}.

Outline. The remainder of the paper is organized as follows. We describe the design of ICEPOLE in Section 2. Afterwards, we give a high-level overview about the techniques used to find suitable differential characteristics in Section 3, followed by our attacks on round-reduced versions of ICEPOLE in Section 4. Finally, we conclude in Section 5.

2 Description of ICEPOLE

In this section, we give a short description of ICEPOLE-128 as it is specified in the CAESAR design document [16]. For more details about ICEPOLE-128 and the other members of the family, we refer to the CAESAR design document [16].

In case of disagreement between the specifications of ICEPOLE-128 in the CHES and CAESAR documents, we always stick to the version submitted to CAESAR and the available reference implementations of this version.

2.1 Mode of Operation

ICEPOLE uses a duplex-like [5] mode of operation, which operates on an internal state of 1280 bits. This state S is represented as 20 64-bit words $S[0..3][0..4]$. Bits on the same position of all 20 words are called slice. The encryption (as well as the decryption) can be split into the three subsequent phases: initialization, processing of data, and finalization (tag generation), and is shown in Fig. 1.

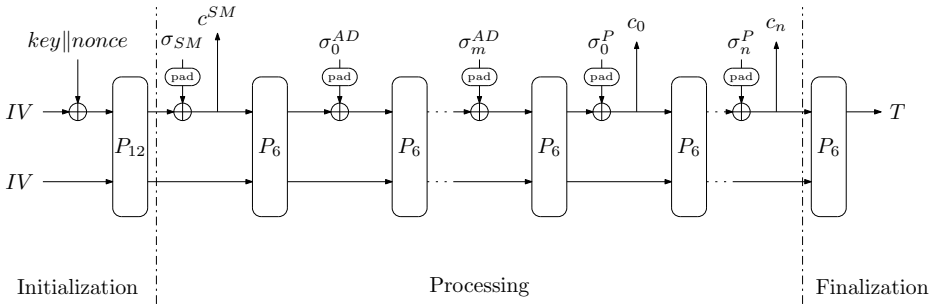


Fig. 1. Encryption of ICEPOLE-128.

Initialization. First, the state S is initialized with a constant value. Afterwards, the 128-bit key and the 128-bit nonce are xored to the internal state. Then, the 12-round variant P_{12} of the ICEPOLE permutation is applied.

Processing of Data. For processing, the associated data and the plaintext are split into 1024-bit blocks, with possibly smaller last blocks. Each of these blocks is padded to 1026 bits. The padding rule is to append a frame bit, followed by a single 1 and zeros until 1026 bits are reached.

After the initialization, the padded secret message number σ_{SM} is xored to the internal state and c_{SM} is extracted. Then, 6 rounds of the ICEPOLE permutation P_6 are applied. After the processing of the secret message number, the associated data blocks σ_i^{AD} are padded and injected, separated by the 6-round ICEPOLE permutation P_6. The plaintext blocks σ_i^P are processed in a similar way, except that ciphertext blocks c_i are extracted. For easier comparison with other sponge-based [3,4] primitives, we move the last permutation call P_6 (after the last plaintext block) to the finalization.

Finalization. Since we moved the last permutation call of the processing to the finalization, the finalization starts with calling P_6. Afterwards, the 128-bit tag T is extracted from the state:

$$T = S[0][0] \| S[1][0].$$

2.2 Permutation

Two variants of the ICEPOLE permutation are used: One with 6 rounds, P_6, and one with 12 rounds, P_{12}. Each round R consists of five steps, $R = \kappa \circ \psi \circ \pi \circ \rho \circ \mu$.

- $\boldsymbol{\mu}$: Mixing of every 20-bit slice through an MDS matrix over $GF(2^5)$.
- $\boldsymbol{\rho}$: Rotation within all 64-bit words.
- $\boldsymbol{\pi}$: Reordering of 64-bit words (words are swapped).
- $\boldsymbol{\psi}$: Parallel application of 256 identical 5-bit S-boxes.
- $\boldsymbol{\kappa}$: Constant addition.

For a detailed description of κ, ψ, π, ρ, and μ, we refer to the CAESAR design document [16].

3 Search for Differential Characteristics

As we will see later, the existence of differential characteristics holding with a high probability is crucial for our attacks on round-reduced ICEPOLE-128. Since ICEPOLE-128 is a bit-oriented construction, automatic search tools are helpful for finding complex differential characteristics with a high probability. ICEPOLE-128 has a rather big internal state of 1280 bits involving many operations per permutation round. Therefore, we have decided to use the guess-and-determine techniques already used for several attacks on hash functions [12,14,15] together with a greedy strategy, which has already been used to find differential characteristics with a high probability for SipHash [10].

We first describe the used concepts for representing differences within the used automatic search tool and propagating them in Section 3.1. Then, we give a high-level overview of our search strategy in Section 3.2.

3.1 Generalized Conditions and Propagation

To represent differential characteristics within the search tool, we have chosen generalized conditions [8]. These conditions are suitable for guess-and-determine-based searches, since they have a very high level of granularity. For instance, with a '`?`', it can be represented that no restrictions are given at some point of a search, or the value of a pair of bits can be completely determined, for instance by '`1`' denoting that both bits of the pair have to have the value 1. The complete set of all 16 generalized conditions can be found in Table 2.

Table 2. Generalized conditions [8].

x, x'	$0,0$	$1,0$	$0,1$	$1,1$
?	✓	✓	✓	✓
-	✓	–	–	✓
x	–	✓	✓	–
0	✓	–	–	–
u	–	✓	–	–
n	–	–	✓	–
1	–	–	–	✓
#	–	–	–	–

x, x'	$0,0$	$1,0$	$0,1$	$1,1$
3	✓	✓	–	–
5	✓	–	✓	–
7	✓	✓	✓	–
A	–	✓	–	✓
B	✓	✓	–	✓
C	–	–	✓	✓
D	✓	–	✓	✓
E	–	✓	✓	✓

Apart from the representation, the propagation of differences (or in this case of the generalized conditions) through the components of the ICEPOLE permutation has to be modeled. Here, we make the distinction between the linear part of one round, consisting of the application of μ, ρ, and π, and the nonlinear part ψ, which is the application of 256 5-bit S-boxes. The propagation for each S-box is done by exhaustively calculating all possible solutions for given input and output differences (basically look-ups in the difference distribution table). The propagation of the linear part of each round is modeled by techniques described in [11].

3.2 Search Strategy

On a high level, our search strategy can be split into the following two phases:

1. Search for a valid characteristic with a low number of active S-boxes.
2. Optimize the probability of the characteristic.

The first phase primarily serves to narrow the search space for the second phase. In this first phase, we search for truncated differentials with as few differentially active S-boxes as possible. In this context, an S-box is called active if there are differences on its inputs and outputs. The number of active S-boxes sets an upper bound on the best possible probability of a characteristic, since the maximum differential probability of the ICEPOLE S-box is 2^{-2}.

In the second phase, we search for the actual characteristic. In fact, just using the truncated differential and searching for the best assignment does not give us the best overall result. As we will see later, we search for characteristics having a special form, where a low number of active S-boxes does not necessarily give the best characteristic. Thus, we only fix the truncated differential for one or two rounds, leaving the other rounds completely undetermined, and search for high-probability characteristics by using the greedy algorithm presented in [10].

In summary, the first phase narrows the search space and gives us a good starting point for the second phase. Then, in the second phase, the actual characteristic is searched.

4 The Attack

The first thing to do when analyzing a cryptographic primitive is to find a promising point to attack. Thus, we explain the observations that have led to the attack on the finalization of ICEPOLE-128 using differential cryptanalysis in Section 4.1. After that, we discuss our first findings regarding forgeries in Section 4.2, and explain the trick leading to an improvement of the attack in Section 4.3. Finally, in Section 4.4, we show characteristics for 5 and 6 rounds of the ICEPOLE permutation which are not suitable for a forgery, but have a better probability than the best characteristics published by the designers [16].

4.1 Basic Attack Strategy

ICEPOLE uses a sponge-like mode of operation like several other CAESAR candidates including Ascon [9], Keyak [6], or NORX [1,2]. When comparing those Sponge constructions with ICEPOLE, it is noticeable that the last permutation, which separates the last plaintext injection from the extraction of the tag, has much fewer rounds in the case of ICEPOLE compared to the others.

For more detail, we have a closer look at the number of permutation rounds during the three different stages for the proposals of Ascon, ICEPOLE, Keyak, and NORX, which have the same security level of 128 bits. The number of rounds in each stage for these four primitives is given in Table 3. We can see that for Ascon and NORX, the number of rounds for data processing is reduced compared to finalization and initialization and for all three competitors of ICEPOLE, the finalization is equally strong as the initialization. In the case of ICEPOLE, the permutation used in the finalization has the same number of rounds as the permutation during the processing of data, and thus just half of the rounds of the permutation used during the initialization. So it is interesting to evaluate if the designers of the competitors of ICEPOLE have been overly conservative in the design of their respective finalization, or if their decision to invest more rounds can be justified.

Table 3. Permutation rounds for some sponge-like CAESAR candidates.

	Initialization	Data Processing	Finalization
Ascon	12	6	12
ICEPOLE	12	6	6
Keyak	12	12	12
NORX	8	4	8

4.2 Creating Forgeries

In this section, we first describe the principles of our attack on a high level. Afterwards, we discuss our preliminary results regarding suitable characteristics when just considering the 1024 bits of the ciphertext blocks to inject differences.

Attack Strategy. For creating forgeries with the help of differential characteristics, we have in principle two attack points in sponge-like constructions as ICEPOLE-128. We can either attack the data processing, or we can perform the attack on the finalization. In both cases, the key to a successful attack lies in the search for a suitable differential characteristic which holds with a high probability.

Fig. 2 shows how a forgery during the processing of the data works. This approach requires a differential characteristic with differences only in those parts of the state that can be modified with message blocks, while the rest of the state has to remain free of differences. In other words, we search for a characteristic capable of producing collisions on the internal state. If we have found such a characteristic with input difference Δ_0 that holds with probability 2^{-x}, we can create a forgery which succeeds with probability 2^{-x} as follows: Assume we know a valid ciphertext-tag pair consisting of two ciphertext blocks $(c_0\|c_1,\ T)$. Then, the ciphertext-tag pair $(c_0 \oplus \Delta_0\|c_1,\ T)$ is valid with probability 2^{-x}. Thus, a valid forgery can be created with complexity 2^x.

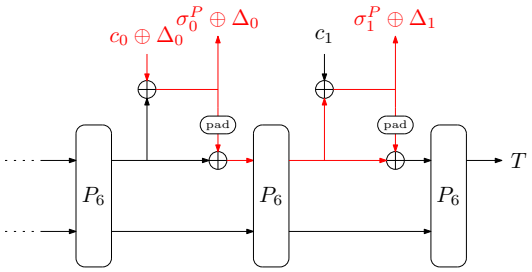


Fig. 2. Forgery during data processing.

The second option, attacking the finalization, is pictured in Fig. 3. In contrast to the previous attack, the requirements on a suitable characteristic can be relaxed. Here, we do not require a collision. It is sufficient that the difference Δ_1 for the tag T is known. The actual difference in the rest of the state does not matter in this attack. In other words, a forgery can be created from a known ciphertext-tag $(c_0\|c_1,\ T)$ by applying suitable differences to c_1 and T to get $(c_0\|c_1 \oplus \Delta_0,\ T \oplus \Delta_1)$.

In case of ICEPOLE, the permutation during the processing of the data and the finalization is equally strong. The requirements on suitable characteristics are

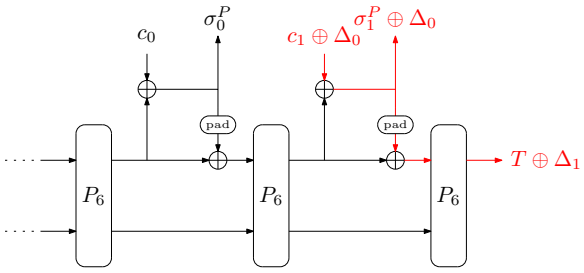


Fig. 3. Forgery during finalization.

less restrictive when attacking the finalization. Thus, attacks on the finalizations are easier to achieve. In addition, the fact that the linear layer is located before the application of the S-boxes comes in handy. ICEPOLE has a state size of 1280 bits. For the generation of the tag, only 128 bits of the 1280 bits are extracted. The other bits do not influence the tag. Since the S-boxes are located at the end of the permutation, 128 of the 256 S-boxes of the last round have no influence on the tag and therefore, do not contribute to the probability of creating a forgery. Moreover, the other 128 S-boxes of the last round only contribute a single bit, which also has a positive effect on the total probability.

Suitable Characteristics. As discussed before, we need characteristics with a good probability, where the input differences lie in the part of the state that can be controlled by a ciphertext block, and where as many of the active S-boxes as possible lie in parts which do not contribute to the probability. However, before we present our results, we describe the findings of the designers [16] and the results by Huang et al. [13].

The designers of ICEPOLE already searched for differential characteristics without any special restrictions. They have found characteristics for 3 rounds with probability $2^{-18.4}$, 4 rounds with $2^{-52.8}$, 5 rounds with $2^{-186.2}$ and 6 rounds with $2^{-555.5}$. Indeed, when considering that the last round of ICEPOLE only contributes partially to the probability, these results look promising from the perspective of an attacker. However, as already observed by Huang et al. [13], these characteristics cannot be used for attacks on the cipher. They showed that if only 1024 bits of a message block are considered suitable for introducing differences, it is impossible to find a 3-round path with 9 active S-boxes in the form 4-1-4. Moreover, they show that the minimum number of active S-boxes in the first round is 2 in this case.

Our search for suitable characteristics supports their result. If we just consider the 1024 bits of the message block suitable for differences, we can create forgeries for 3 rounds with probability $2^{-25.3}$ and, for 4 rounds with a probability close to 2^{-128}. However, in the next section, we explain how we improved the probability for the 4-round attack to $2^{-60.3}$ by exploiting the padding rule of the last processed plaintext block.

4.3 Exploiting the Padding

ICEPOLE uses at most 1024-bit message blocks, which are padded to 1026 bits by appending a frame bit, which is 0 for the last plaintext block, followed by a single 1 and as many zeros until 1026 bits are reached. So using, for instance, a 1016-bit block and a 1024-bit block (where the last byte fulfills the padding rule applied to the 1016-bit block) virtually flips a bit in an otherwise unaccessible part of the state. By using this trick, we are able to use characteristics where only one S-box is active in the first round.

With these improved differential characteristics, we are able to create forgeries for ICEPOLE-128 with the finalization reduced to 3 (out of 6) rounds with probability $2^{-14.8}$, and for 4 rounds (out of 6) with probability $2^{-60.3}$. The characteristics for the 3-round attack can be found in Table 4, and for the 4-round attack in Table 5 of Appendix A.

The 3-round attack on ICEPOLE-128 has been verified using the reference implementation ICEPOLE128v1 submitted to CAESAR with a modified number of rounds for permutation P_6. We fixed a random key at the beginning and encrypted random 1024-bit messages (last byte of messages has to be equal to the padding `0x2`) with random nonces to get 1024-bit ciphertexts. The forgeries are created by applying the difference shown in Table 4 to ciphertext and tag and discarding the last byte of the ciphertext. Removing the last byte of the ciphertext introduces a difference at bit 1026. Backed up by our experiments (2^{28} message-tag pairs), a forgery for round-reduced ICEPOLE-128, where the finalization is reduced to 3 out of 6 rounds, can be created with probability $2^{-11.7}$. For the 4-round attack, the probability is too low to be verified experimentally. However, parts of the used characteristic which have a high probability have been verified.

To introduce differences with the help of the padding, we can either extend or truncate known ciphertexts. As already discussed before, creating forgeries by truncating the last byte of the ciphertext only works if the last byte of the message before encryption equals the padding. Extending 1016-bit ciphertexts requires to guess 8 bits of the internal state correctly and hence decreases the probability by 2^{-8}. In the case of messages consisting of a fractional number of bytes, 1022-bit ciphertexts can be extended, leading to a decrease of 2^{-2}.

4.4 Characteristics for the Permutation

We also considered characteristics without any special restrictions. We have been able to improve the results published in the design documents. We have found a 5-round characteristic with an estimated probability of $2^{-104.5}$ and a 6-round characteristic with an estimated complexity of $2^{-258.4}$. The characteristics are given in Table 6 and Table 7 of Appendix A. Both characteristics are a perceptible improvement over the characteristics given in the design document [16], which have a probability of $2^{-186.2}$ and $2^{-555.3}$, respectively.

5 Conclusion

In this work, we have analyzed the resistance of ICEPOLE-128 against forgery attacks. Our attacks work for versions of ICEPOLE-128 where the permutation used during the finalization is reduced to 4 (out of 6) rounds. This means that ICEPOLE-128 has a security margin of 2 rounds, which is lower than the 3 rounds expected by the designers [16].

Acknowledgments. The work has been supported in part by the Austrian Science Fund (project P26494-N15) and by the Austrian Research Promotion Agency (FFG) and the Styrian Business Promotion Agency (SFG) under grant number 836628 (SeCoS).

References

1. Aumasson, J., Jovanovic, P., Neves, S.: NORX. Submission to the CAESAR competition: `http://competitions.cr.yp.to/round1/norxv1.pdf` (2014)
2. Aumasson, J., Jovanovic, P., Neves, S.: NORX: Parallel and Scalable AEAD. In: Kutylowski, M., Vaidya, J. (eds.) Computer Security – ESORICS 2014, Part II. LNCS, vol. 8713, pp. 19–36. Springer (2014)
3. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Sponge Functions. ECRYPT Hash Workshop 2007 (May 2007)
4. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the Indifferentiability of the Sponge Construction. In: Smart, N.P. (ed.) Advances in Cryptology – EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer (2008)
5. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In: Miri, A., Vaudenay, S. (eds.) Selected Areas in Cryptography – SAC 2011. LNCS, vol. 7118, pp. 320–337. Springer (2011)
6. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: Keyak. Submission to the CAESAR competition: `http://competitions.cr.yp.to/round1/keyakv1.pdf` (2014)
7. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. J. Cryptology 4(1), 3–72 (1991)
8. De Cannière, C., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In: Lai, X., Chen, K. (eds.) Advances in Cryptology – ASIACRYPT 2006. LNCS, vol. 4284, pp. 1–20. Springer (2006)
9. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon. Submission to the CAESAR competition: `http://competitions.cr.yp.to/round1/asconv1.pdf` (2014)
10. Dobraunig, C., Mendel, F., Schläffer, M.: Differential Cryptanalysis of SipHash. In: Joux, A., Youssef, A.M. (eds.) Selected Areas in Cryptography – SAC 2014. LNCS, vol. 8781, pp. 165–182. Springer (2014)
11. Eichlseder, M., Mendel, F., Nad, T., Rijmen, V., Schläffer, M.: Linear Propagation in Efficient Guess-and-Determine Attacks. In: Lilya Budaghyan, Tor Helleseth, M.G.P. (ed.) International Workshop on Coding and Cryptography. pp. 193–202 (2013)

12. Eichlseder, M., Mendel, F., Schläffer, M.: Branching heuristics in differential collision search with applications to SHA-512. In: Cid, C., Rechberger, C. (eds.) Fast Software Encryption – FSE 2014. LNCS, vol. 8540, pp. 473–488. Springer (2014)
13. Huang, T., Tjuawinata, I., Wu, H.: Differential-linear cryptanalysis of ICEPOLE. In: Leander, G. (ed.) Fast Software Encryption – FSE 2015. LNCS, vol. 9054, pp. 243–263. Springer (2015)
14. Mendel, F., Nad, T., Schläffer, M.: Finding SHA-2 Characteristics: Searching through a Minefield of Contradictions. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology – ASIACRYPT 2011. LNCS, vol. 7073, pp. 288–307. Springer (2011)
15. Mendel, F., Nad, T., Schläffer, M.: Improving Local Collisions: New Attacks on Reduced SHA-256. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology – EUROCRYPT 2013. LNCS, vol. 7881, pp. 262–278. Springer (2013)
16. Morawiecki, P., Gaj, K., Homsirikamol, E., Matusiewicz, K., Pieprzyk, J., Rogawski, M., Srebrny, M., Wójcik, M.: ICEPOLE. Submission to the CAESAR competition: `http://competitions.cr.yp.to/round1/icepolev1.pdf` (2014)
17. Morawiecki, P., Gaj, K., Homsirikamol, E., Matusiewicz, K., Pieprzyk, J., Rogawski, M., Srebrny, M., Wójcik, M.: ICEPOLE: High-Speed, Hardware-Oriented Authenticated Encryption. In: Batina, L., Robshaw, M. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2014. LNCS, vol. 8731, pp. 392–413. Springer (2014)
18. The CAESAR committee: CAESAR: Competition for authenticated encryption: Security, applicability, and robustness (2014), `http://competitions.cr.yp.to/caesar.html`

A Differential Characteristics

Table 4. 3-round characteristic suitable for forgery with probability $2^{-14.8}$.

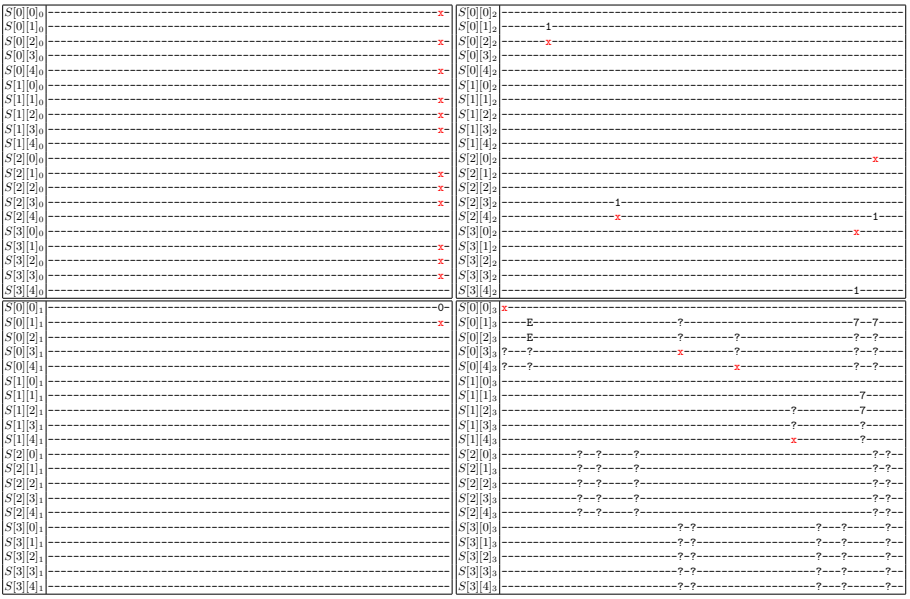


Table 5. 4-round characteristic suitable for forgery with probability $2^{-60.3}$.

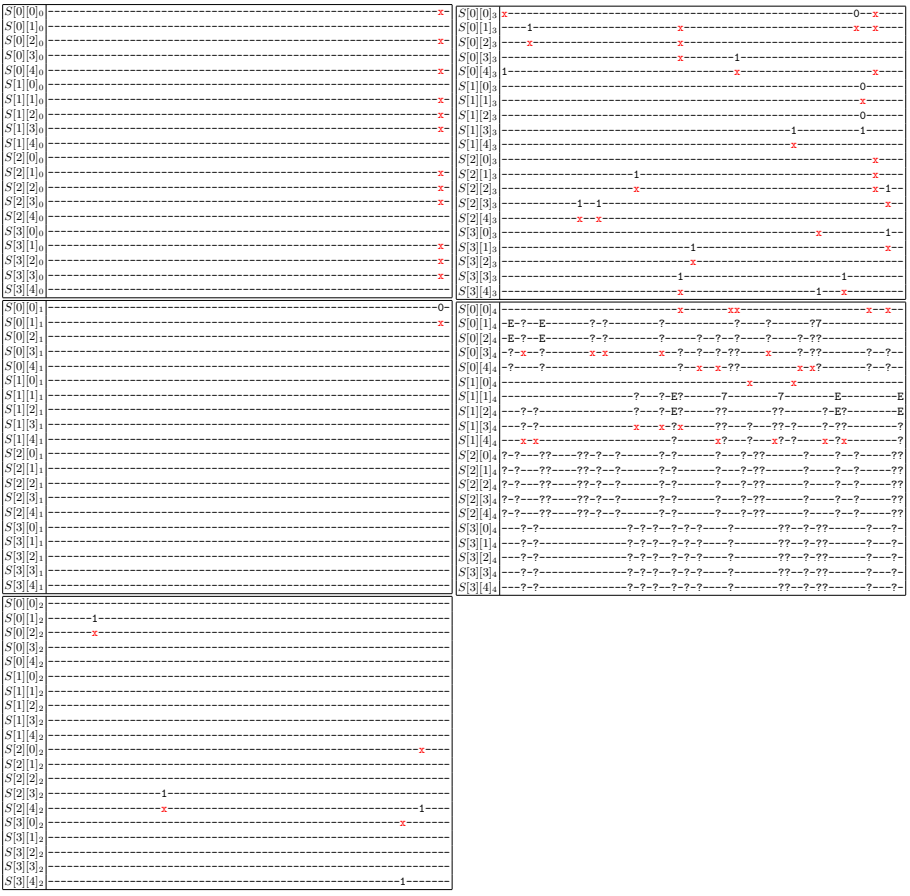


Table 6. 5-round characteristic with probability $2^{-104.5}$.

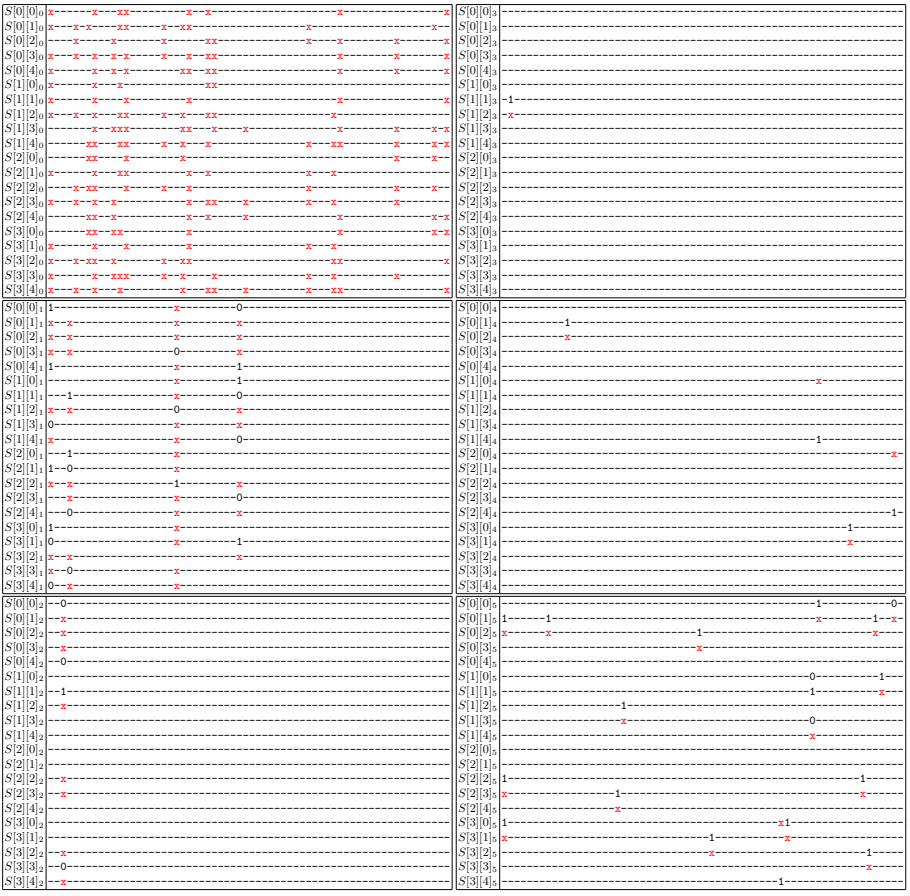
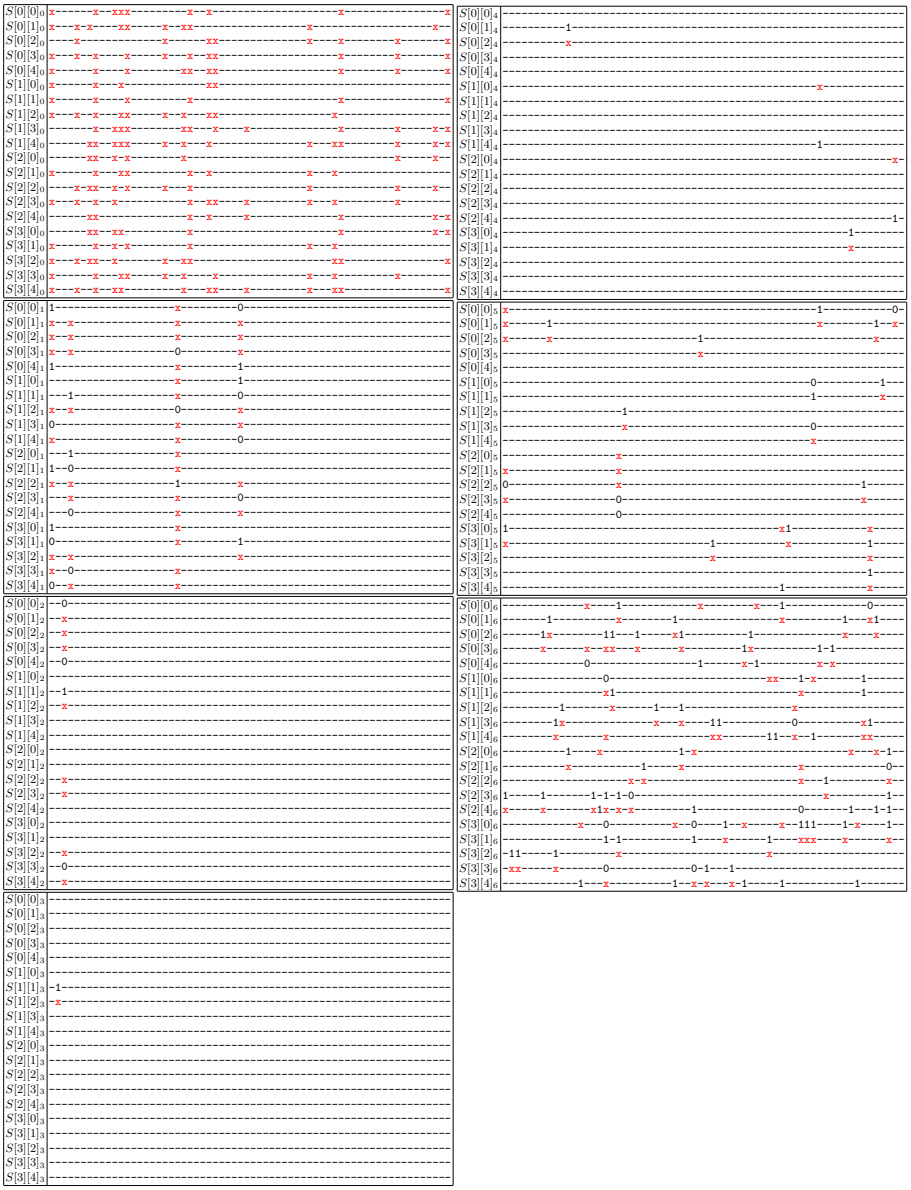


Table 7. 6-round characteristic with probability $2^{-258.3}$.



Cryptanalysis of Ascon

Publication Data

Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. "Cryptanalysis of Ascon". In: *Topics in Cryptology – CT-RSA 2015*. Ed. by Kaisa Nyberg. Vol. 9048. LNCS. Springer, 2015, pp. 371–387. URL: `https://doi.org/10.1007/978-3-319-16715-2_20`

The appended paper is an author-created version available at `https://eprint.iacr.org/2015/030`.

Contributions

- **Technical:** Contributed to cube-like attacks, differential forgery attacks, and differential-linear attacks. Performed practical experiments for cube-like attacks, differential forgery attacks, and differential-linear attacks (except key recovery for differential-linear attacks). Performed heuristic search for linear characteristics. No contributions to the search for differential characteristics, providing linear and differential bounds, and improvement of zero-sum distinguishers using Walsh spectrum analysis.
- **Writing:** Contributions to the writing of Sections 3, 4, 5.2, 5.3, and 5.4.

Cryptanalysis of Ascon

Christoph Dobraunig[1], Maria Eichlseder[1], Florian Mendel[1], and Martin Schläffer[2]

[1] IAIK, Graz University of Technology, Austria
firstname.lastname@iaik.tugraz.at
[2] Infineon Technologies AG, Austria
martin.schlaeffer@gmail.com

Abstract. We present a detailed security analysis of the CAESAR candidate Ascon. Amongst others, cube-like, differential and linear cryptanalysis are used to evaluate the security of Ascon. Our results are practical key-recovery attacks on round-reduced versions of Ascon-128, where the initialization is reduced to 5 out of 12 rounds. Theoretical key-recovery attacks are possible for up to 6 rounds of initialization. Moreover, we present a practical forgery attack for 3 rounds of the finalization, a theoretical forgery attack for 4 rounds finalization and zero-sum distinguishers for the full 12-round Ascon permutation. Besides, we present the first results regarding linear cryptanalysis of Ascon, improve upon the results of the designers regarding differential cryptanalysis, and prove bounds on the minimum number of (linearly and differentially) active S-boxes for the Ascon permutation.

Keywords: Ascon, CAESAR initiative, cryptanalysis, authenticated encryption

1 Introduction

The CAESAR competition [20] is an ongoing cryptographic competition, where numerous authenticated encryption schemes are challenging each other with the goal of finding a portfolio of ciphers, suitable for different use-cases. Currently, more than 45 ciphers are still participating in the competition. In the near future, this portfolio will be further reduced to focus the attention of the crypto community on a few candidates. Therefore, analyzing the security of the candidate ciphers is of great importance to enable the committee to judge them adequately.

Ascon is a submission by Dobraunig et al. [11] to the CAESAR competition. In the submission document, the designers discuss the design rationale for the cipher and give first cryptanalytic results, in particular on the differential properties of the Ascon permutation. Since the cipher was only recently presented, results of external cryptanalysis are scarce so far. Jovanovic et al. [15] prove the security of Ascon's mode of operation under idealness assumptions for the permutation.

The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-16715-2_20

Our contribution. We present a detailed security analysis of the CAESAR candidate Ascon-128. Based on the low algebraic degree of Ascon, we are able to construct a zero-sum distinguisher with complexity 2^{130} for the full 12-round Ascon permutation in Section 3. In Section 4, we use similar algebraic properties to construct a distinguisher based on cube testers. We also use cube-like techniques to obtain a key-recovery attack for a round-reduced version of Ascon with 5-round initialization with practical complexity. Theoretical key-recovery attacks are possible for up to 6 rounds of initialization. Moreover, in Section 5, we present the first results on linear cryptanalysis, and improve the results by the designers on differential cryptanalysis. Our results include linear and differential characteristics obtained with heuristic search, as well as a computer-aided proof of security bounds against linear and differential cryptanalysis (minimum number of active S-boxes). Using our results on linear-differential analysis, we present a practical forgery attack for 3 rounds of the finalization and a theoretical forgery attack for 4-round finalization. Our results are summarized in Table 1.

Table 1. Results for Ascon-128. Attacks performed on the initialization or finalization.

type	rounds	time	method	source
permutation distinguisher	12 / 12	2^{130}	zero-sum	Section 3
key recovery	6 / 12	2^{66}	cube-like	Section 4.4
	5 / 12	2^{35}		
	5 / 12	2^{36}	differential-linear	Section 5.4
	4 / 12	2^{18}		
forgery	4 / 12	2^{101}	differential	Section 5.3
	3 / 12	2^{33}		

2 Ascon

Ascon is a submission by Dobraunig et al. [11] to the ongoing CAESAR competition. It is based on a sponge-like construction with a state size of 320 bits (consisting of five 64-bit words $x_0, \ldots, x_4$). Ascon comes in two flavors, Ascon-128 and Ascon-96, with different security levels and parameters, as summarized in Table 2. The analysis in this paper is focused on Ascon-128. In the following, we give a brief overview about the mode of operation and the permutation of Ascon. For a complete description, we refer to the design document [11].

Mode of operation. Ascon's mode of operation is based on MonkeyDuplex [8]. As illustrated in Fig. 1, the encryption is partitioned into four phases: initialization, processing associated data, processing the plaintext, and finalization. Those phases use two different permutations p^a and p^b. The stronger

Table 2. Parameters for Ascon [11].

name	bit size of					rounds	
	key	nonce	tag	data block		p^a	p^b
Ascon-128	128	128	128	64		12	6
Ascon-96	96	96	96	128		12	8

variant p^a is used for initialization and finalization, while p^b is used in the data processing phases.

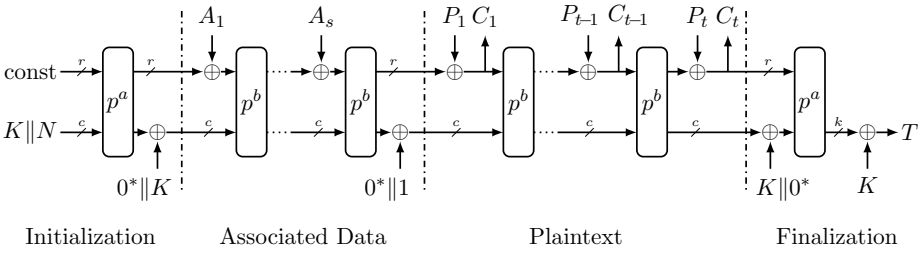


Fig. 1. The encryption of Ascon [11].

The initialization takes as input the secret key K and the public nonce N. The initialization ensures that we start with a random-looking state at the beginning of the data procession phase for every new nonce. In the subsequent processing of the associated data, r-bit blocks are absorbed by xoring them to the state, separated by invocations of p^b. If no associated data needs to be processed, the whole phase can be omitted. Plaintext is processed in r-bit blocks in a similar manner, with ciphertext blocks extracted from the state right after adding the plaintext. For domain separation between associated data and plaintext, a constant is xored to the secret part of the internal state. After all data is processed, the finalization starts and the k-bit tag T is returned.

Table 3. The S-box of Ascon [11].

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\mathcal{S}(x)$	4	11	31	20	26	21	9	2	27	5	8	18	29	3	6	28
x	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$\mathcal{S}(x)$	30	19	7	14	0	13	17	24	16	12	1	25	22	10	15	23

Permutation. Ascon uses the two permutations p^a and p^b. Both iteratively apply the same round function p: a rounds for p^a, and b rounds for p^b. The round transformation p consists of a constant addition to x_2, followed by the application of a nonlinear substitution layer and a linear layer.

The substitution layer uses a 5-bit S-box (Table 3), which is affine equivalent to the Keccak [2] χ mapping. The Ascon S-box is applied 64 times in parallel on the state. Each bit of the 5 64-bit words $(x_0, \ldots, x_4)$ contributes one bit to each of the 64 S-boxes, where x_0 always serves as most significant bit.

The linear layer is derived from the Σ-function of SHA-2 [19]. The Σ-function is applied to each of the 5 state-words and uses different rotation values for each word:

$$\begin{aligned}
\Sigma_0(x_0) &= x_0 \oplus (x_0 \ggg 19) \oplus (x_0 \ggg 28) \\
\Sigma_1(x_1) &= x_1 \oplus (x_1 \ggg 61) \oplus (x_1 \ggg 39) \\
\Sigma_2(x_2) &= x_2 \oplus (x_2 \ggg \ 1) \oplus (x_2 \ggg \ 6) \\
\Sigma_3(x_3) &= x_3 \oplus (x_3 \ggg 10) \oplus (x_3 \ggg 17) \\
\Sigma_4(x_4) &= x_4 \oplus (x_4 \ggg \ 7) \oplus (x_4 \ggg 41)
\end{aligned}$$

3 Zero-sum distinguishers

In this section, we apply zero-sum distinguishers used in the analysis of Keccak [1,6,7] to Ascon. Zero-sum distinguishers have been used to show non-ideal properties of round-reduced versions for the Keccak permutation. With the help of zero-sum distinguishers, Boura et al. have have been able to distinguish the full 24-round Keccak permutation from a random permutation. Since the core of the Ascon S-box corresponds to the Keccak S-box, we are able to construct distinguishers for the full 12 rounds (or up to 20 rounds) of the Ascon permutation.

Algebraic model of Ascon. As the name zero-sum distinguishers suggests, we search for a set of inputs and corresponding outputs of an n-bit permutation which sum to zero over $\mathbb{F}_2^n$. To create this set of input-output pairs, we start in the middle of the permutation and compute outwards. Furthermore, we keep a set of $320 - d$ bits constant and vary the other d bits through all possible assignments. Thus, we get 2^d possible intermediate states. For all these 2^d intermediate states, we calculate the respective outputs. If the degree of the function determining the output bits is strictly smaller than d, the resulting outputs will sum to zero over $\mathbb{F}_2^n$ [1,6]. After that, we calculate the input values of the permutation using the 2^d intermediate states. Again, if the degree of the inverse function is smaller than d, the inputs sum to zero over $\mathbb{F}_2^n$. The result is a zero-sum distinguisher, or rather, a family of zero-sum distinguishers.

To apply the technique to Ascon, we have to bound the degree of multiple rounds of the Ascon permutation and its inverse. The algebraic degree of one

Ascon S-box is 2, with respect to $\mathbb{F}_2$, and can be easily determined from its algebraic normal form (ANF):

$$
\begin{aligned}
y_0 &= x_4x_1 + x_3 + x_2x_1 + x_2 + x_1x_0 + x_1 + x_0 \\
y_1 &= x_4 + x_3x_2 + x_3x_1 + x_3 + x_2x_1 + x_2 + x_1 + x_0 \\
y_2 &= x_4x_3 + x_4 + x_2 + x_1 + 1 \\
y_3 &= x_4x_0 + x_4 + x_3x_0 + x_3 + x_2 + x_1 + x_0 \\
y_4 &= x_4x_1 + x_4 + x_3 + x_1x_0 + x_1
\end{aligned}
$$

Here, x_0, x_1, x_2, x_3, x_4, and y_0, y_1, y_2, y_3, y_4 represent the input, and output of an S-box, with x_0/y_0 representing the most significant bit. The S-boxes in one substitution layer are applied in parallel to the state, and the linear layer and constant addition do not increase the algebraic degree. Consequently, the overall degree of one Ascon permutation round is 2, and the degree of r rounds is at most 2^r.

To determine the degree of the inverse permutation, we use the ANF of the inverse Ascon S-box:

$$
\begin{aligned}
x_0 &= y_4y_3y_2 + y_4y_3y_1 + y_4y_3y_0 + y_3y_2y_0 + y_3y_2 + y_3 + y_2 + y_1y_0 + y_1 + 1 \\
x_1 &= y_4y_2y_0 + y_4 + y_3y_2 + y_2y_0 + y_1 + y_0 \\
x_2 &= y_4y_3y_1 + y_4y_3 + y_4y_2y_1 + y_4y_2 + y_3y_1y_0 + y_3y_1 + y_2y_1y_0 \\
&\quad + y_2y_1 + y_2 + 1 + x_1 \\
x_3 &= y_4y_2y_1 + y_4y_2y_0 + y_4y_2 + y_4y_1 + y_4 + y_3 + y_2y_1 + y_2y_0 + y_1 \\
x_4 &= y_4y_3y_2 + y_4y_2y_1 + y_4y_2y_0 + y_4y_2 + y_3y_2y_0 + y_3y_2 + y_3 \\
&\quad + y_2y_1 + y_2y_0 + y_1y_0
\end{aligned}
$$

The algebraic degree of the ANF of the inverse Ascon S-box is 3. Therefore, the degree for an r-round inverse Ascon permutation is at most 3^r.

Basic distinguisher for 12 rounds. To create a zero-sum distinguisher for the 12-round Ascon permutation that is used for the cipher's initialization and finalization, we target the intermediate state after round 5. Thus, we attack 5 backward (inverse) rounds and 7 forward rounds. An upper bound for the degree of the 7-round permutation is $2^7 = 128$, while for the 5 inverse rounds, an upper bound is $3^5 = 243$. So we choose $d = 244$, fix $320 - 244 = 76$ bits of the intermediate state and vary the remaining 244 bits to create a set of 2^{244} intermediate states. For all these states, we calculate 7 rounds forward and 5 rounds backward. The sum of all the resulting input and output values over $\mathbb{F}_2^n$ is zero. A similar attack is possible for $11 = 4+7$ rounds (with $d = \max\{81, 128\} + 1 = 129$) and for $13 = 5 + 8$ rounds (with $d = \max\{243, 256\} + 1 = 257$).

Improvement using Walsh spectrum analysis. The complexity of the 12-round distinguisher can be further improved by analyzing the permutation's

Walsh spectrum and applying the techniques by Boura and Canteaut [6]: If the Walsh spectrum of a function $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is 2^ℓ-divisible, then for any $G : \mathbb{F}_2^n \to \mathbb{F}_2^n$, we have

$$\deg(G \circ F) \leq n - \ell + \deg(G).$$

As Boura and Canteaut show, the Walsh spectrum of the Keccak S-box is 2^3-divisible. The affine linear preprocessing and postprocessing that the Ascon S-box adds compared to the Keccak S-box does not change this number. The same holds true for the inverse S-box. The Ascon nonlinear layer applies this S-box 64 times in parallel. The Walsh spectrum of a parallel composition is the multiplication of the individual Walsh spectra [6]. Thus, the Walsh spectrum of the complete nonlinear layer is divisible by $2^{3 \cdot 64} = 2^{192}$. Let p denote one round of the Ascon permutation, and p^{-1} its inverse. A closer bound on the degree of 5 rounds of the inverse permutation, p^{-5}, is then obtained by

$$\deg(p^{-5}) = \deg(p^{-4} \circ p^{-1}) \leq 320 - 192 + \deg(p^{-4}) \leq 320 - 192 + 81 = 209.$$

Thus, $d = \max\{209, 128\} + 1 = 210$ is sufficient for $12 = 5 + 7$ rounds of the Ascon permutation.

Adding a free round in the middle. Additionally, as Boura and Canteaut [6] observe, an additional round can be added to the attack (almost) for free as follows: The original attack requires an intermediate state where $n - d$ bits are fixed to a constant, while d bits loop through all possible valuations. Now, we set d to be a multiple of the 5-bit S-box size and furthermore, choose the d variable bits such that they always include complete S-boxes. Then, the inputs (and consequently outputs) of some S-boxes are constant, while the other S-boxes have their inputs (and consequently outputs) loop through all possible values. If we look at the output of the nonlinear layer after this intermediate step, we observe it adheres to the same pattern as the input: $n - d$ bits are fixed and d bits enumerate through all their possible values. We can now use the original intermediate step as the starting point for the backwards rounds, and the output of the nonlinear layer as the starting point for the forward rounds (plus an additional, free linear layer). This way, we can extend the previous attacks by one round each, with the only additional cost of choosing d as a multiple of 5. We get zero-sum distinguishers on 12, 13, and 14 rounds with $d = 130$, 210, and 260, respectively.

More rounds. Finally, the results of Boura et al. [7, Theorem 2] are also directly applicable to our previous results to distinguish up to 20 permutation rounds with $d = 319$ (using 9 backward rounds with degree ≤ 318 and 11 forward rounds with degree ≤ 317, no free middle round possible).

Using a zero-sum distinguisher, we can show non-random properties for the full 12-round permutation of Ascon. However, the designers already state [11]

that the permutation is not ideal and are aware of such distinguishers. The non-ideal properties of the permutation do not seem to affect the security of Ascon. In particular, the complexity of 2^{130} is above the cipher's claimed security level.

4 Cube attacks

Recently, Dinur et al. [9] published various cube and cube-like attacks on several keyed primitives using the Keccak permutation. Those cube-like attacks include cube testers, which can serve as distinguishers, and also cube-like attacks to recover the secret key. In this section, we apply two attacks presented by Dinur et al. [9] to Ascon.

4.1 Brief description of cube attacks

The cube attack is an algebraic attack developed by Dinur and Shamir [10]. This algebraic attack builds on the fact that for most ciphers, each output bit can be represented as a polynomial over $\mathbb{F}_2^n$ in algebraic normal form (ANF). The variables x_i of this polynomial may be single bits of plaintext, key-bits, or constants. Dinur and Shamir made the following observation: If a carefully chosen set of plaintext bits is varied over all possible values and the other bits are kept constant, the sum of one bit of the output (cube sum) might be the result of a linear polynomial (called superpoly) consisting solely of bits of the secret key. By gathering many of these linear polynomials, the secret key can be found.

To perform such a cube attack on a cipher, two things have to be done. First, an attacker has to find such cubes (variables to vary and the resulting linear key relations). This is done in an offline preprocessing phase. Here, the attacker determines the cubes by selecting the cube variables randomly and check if the resulting superpoly is linear and contains the key. This preprocessing phase has to be carried out once for each cipher. In an online phase, the attacker uses the knowledge of the cubes to recover the secret key of his target. To perform the attack, the attacker has to be able to choose the plaintext according to his needs and obtain the corresponding ciphertext outputs.

4.2 Cube attack on Ascon

Now we want to investigate the potential threat of cube attacks to Ascon. If we look at the different phases of Ascon, the only phase where a nonce-respecting adversary can easily keep some inputs of the permutation constant and deterministically influence others is the initialization. In this scenario, the key is kept secret and the attacker has the ability to choose the nonce according to his needs.

As evaluated in Section 3, the degree of a 5-round initialization of Ascon is at most 32. Thus, if we search for cubes of 31 variables, the resulting superpoly is definitely linear or constant. Considering 6 rounds of the initialization, we

have to look for cubes with at most 63 variables, for 7 rounds with at most 127 variables and so on. So it is likely that a practical cube attack on 6 rounds is already hard to achieve. However, we have not searched for cubes, but instead performed cube-like attacks on Ascon to recover the secret key in Section 4.4.

4.3 Distinguishers using cube testers

Below, we describe a cube tester for 6 rounds of the Ascon permutation with the property that the generated output bits sum to zero over $\mathbb{F}_2$. Moreover, this cube tester has a practical complexity of only 2^{33}, although the expected degree for 6 rounds of the Ascon permutation is about 64. To achieve this, we have to take a closer look at the internal structure of Ascon.

The permutation of Ascon starts with the substitution layer. In this layer, the 5-bit S-box is applied 64 times in parallel to the internal state of Ascon. Each of the five 64-bit words of the internal state contributes exactly one bit to each instantiation of a 5-bit S-box. So if all cube variables lie within the same word of the state, they do not appear together in one term after the application of the S-box layer. Hence, after 5 more rounds, at most 32 variables of one state-word appear together in one term. As a consequence, selecting a cube of 33 variables of the same state-word definitely results in an empty superpoly and all 2^{33} generated outputs sum to zero.

This distinguisher can be used to distinguish the key-stream generated by Ascon-128 in a nonce-misuse scenario, where the attacker can keep the nonce constant while varying the plaintext. For Ascon-128, 64-bit blocks of plaintext are xored with the state-word x_0. Thus, the attacker can vary 33 bits of the first plaintext block, while keeping the remaining 31 bits and the bits of a second plaintext block constant. The resulting 2^{33} second ciphertext blocks will sum to zero. However, the designers of Ascon strictly forbid nonce reuse, and no security claims are made for such a scenario.

Similar cube testers can be applied to reduced versions of Ascon with only 6 rounds (instead of 12 rounds) of initialization. Then, an attacker with control over the nonce can observe the first key-stream block. In contrast to the nonce-misuse scenario, attacks on round-reduced versions of Ascon in a nonce-respecting scenario give insight in the expected security of Ascon and are therefore of more value. Next, we will show how to extend the observations made in this section to a key-recovery attack on round-reduced versions of Ascon.

4.4 Key recovery using cube-like attacks

Dinur et al. [9] published a key recovery attack where the superpoly does not necessarily have to be a linear function of the secret key bits, but can also be non-linear. Such attacks are also possible for round-reduced versions of Ascon, with the initialization reduced to 5 or 6 out of 12 rounds. The attack on 5 rounds has practical complexity and has been implemented. We will discuss the working principle of the attack by means of a 5-round version of Ascon-128. For a 6-round initialization, the attack works similarly. The attack itself is divided into

two steps, each with an online and an offline phase, and relies on the following two observations.

Observations. The first observation has already been discussed in the context of cube testers: If all cube variables are located within one state-word, they do not appear in the same term of the output polynomial after one application of the substitution layer.

To discuss the second observation, we have to take a look at the ANF of the S-box and consider the positions of the initial values. During the initialization, the constant C is written to x_0, the first word K_1 of the key to x_1, the second key word K_2 to x_2, the first word N_1 of the nonce to x_3, and the second nonce word N_2 to x_4. We use the ANF of the S-box to get the relations for the state words $x_0, \ldots, x_4$ after the first call of the substitution layer. The index i represents the corresponding bit position of the 64-bit word.

$$
\begin{aligned}
x_0[i] &= N_2[i]K_1[i] + N_1[i] + K_2[i]K_1[i] + K_2[i] + K_1[i]C[i] + K_1[i] + C[i] \\
x_1[i] &= N_2[i] + N_1[i](K_2[i] + K_1[i]) + N_1[i] + K_2[i]K_1[i] + K_2[i] + K_1[i] + C[i] \\
x_2[i] &= N_2[i]N_1[i] + N_2[i] + K_2[i] + K_1[i] + 1 \\
x_3[i] &= N_2[i]C[i] + N_2[i] + N_1[i]C[i] + N_1[i] + K_2[i] + K_1[i] + C[i] \\
x_4[i] &= N_2[i]K_1[i] + N_2[i] + N_1[i] + K_1[i]C[i] + K_1[i]
\end{aligned}
$$

Observe that $N_2[i]$ is only combined nonlinearly with key bit $K_1[i]$, and $N_1[i]$ only with $K_1[i]$ and $K_2[i]$. As demonstrated by Dinur et al. [9], we can make use of this fact to build a so-called borderline cube. For instance, we select $N_2[0..15]$ as our cube variables. The rest of the nonce is kept constant. After round 1, our cube variables only appear with $K_1[0..15]$ in one term and definitely not together with the other bits of the secret key. After 4 more rounds, all of the cube variables may appear together in one term, possibly combined with a selection of the key bits $K_1[0..15]$, but never together with the rest of the key bits. Thus, the cube sum depends on $K_1[0..15]$, but it does not depend on $K_1[16..63]$, or $K_2[0..63]$. This fact leads to the following attack.

Step 1. In the first step, we recover the key-word K_1 in 16-bit chunks. Therefore, we select 4 different borderline cubes with 16 variables in N_2 and probe the online oracle with each of these 4 sets. So we get 4 sums of key-stream blocks, each dependent on 16 different key bits of K_1. In the upcoming offline phase, we use the fact that the sum of the outputs (key-stream blocks) only depends on 16 key bits. So we set the rest of the key bits to a constant and calculate cube sums for every possible 16-bit key part. If such a cube sum corresponds to the cube sum received in the online phase, we get a key candidate. In our experiments, we only received one key candidate per 16-bit block on average. Therefore, we only have one key candidate on average for K_1.

Step 2. In the second step, we recover K_2 in 16-bit chunks. To do so, we use $N_1[i]$ to create our borderline cubes. In contrast to the step before, we

have a dependency of the output on bits of K_1, too. So we have to repeat the offline phase for every guess of K_1 received in the previous step. The rest of the procedure works in the same manner as for the recovery of K_1. Again, we only received one key guess for K_2 on average in our implementation of the attack.

The complexity of the described attack depends on the number of key candidates for K_1 and K_2. Since the attack on 5 rounds is practical and we have implemented it, we can state that we only have one key candidate on average. So we estimate that the time complexity is about $8 \cdot 2^{32}$. The attack works similarly for reduced versions of Ascon with only 6 initialization rounds. Here, we need borderline cubes of size 32. If we make the optimistic assumption that we only have one key guess for each recovered key word, the estimated time complexity for the 6 round attack is $4 \cdot 2^{64}$.

5 Differential and linear cryptanalysis

Differential [5] and linear [18] cryptanalysis are two standard tools for cryptanalysis. New designs are typically expected to come with some kind of arguments of security against these attacks. For this reason, the designers of Ascon provided security arguments for the individual building blocks (S-box, linear layer), and included first practical results on the differential analysis of Ascon in the design document. In this section, we show some improvements over the existing differential characteristics and present the first linear characteristics for Ascon, including computer-aided proofs on the minimum number of active S-boxes for 3-round characteristics. In addition, we use the combination of differential and linear characteristics to perform practical key-recovery attacks on round-reduced versions of Ascon.

5.1 Linear and differential bounds

Beside using heuristic search techniques to find actual characteristics for Ascon (see Section 5.2), we have also used complete search tools (MILP and SAT) to prove bounds on the best possible linear and differential characteristics. The results are given in this section.

Linear programming. We have first modelled the problem of minimizing the number of active S-boxes in differential characteristics for round-reduced versions of the Ascon permutation as a mixed integer linear program (MILP). The model for R rounds uses the following variables:

- $x_{r,w,b} \in \{0,1\}$ specifies whether bit b of word w of the S-box input in round r is different between the two messages, where $b = 0,\ldots,63$ and $w = 0,\ldots,4$.
- $y_{r,w,b} \in \{0,1\}$ specifies whether bit b of word w of the S-box output in round r is different between the two messages, where $b = 0,\ldots,63$ and $w = 0,\ldots,4$.
- $d_{r,b} \in \{0,1\}$ specifies if S-box b of round r is active, $b = 0,\ldots,63$.
- $u_{r,w,b} \in \{0,1,2\}$ is a helper for the linear layer model in word w of round r.

The optimization objective is to minimize the number of active S-boxes,

$$\min \sum_{r=1}^{R} \sum_{b=0}^{63} d_{r,b}.$$

The S-box is modelled only by specifying its branch number, and linking it with the S-box activeness for each $r = 1, \ldots, R$ and $b = 0, \ldots, 63$:

$$d_{r,b} \leq \sum_{w=0}^{63} x_{r,w,b} \leq 5d_{r,b}, \quad \sum_{w=0}^{63} (x_{r,w,b} + y_{r,w,b}) \geq 3d_{r,b}, \quad d_{r,b} \leq \sum_{w=0}^{63} y_{r,w,b} \leq 5d_{r,b}$$

The linear layer is modelled explicitly for $r = 1, \ldots, R$ and $b = 0, \ldots, 63$:

$$\begin{aligned}
y_{r,0,b} + y_{r,0,b+19} + y_{r,0,b+28} + x_{r+1,0,b} &= 2 \cdot u_{r,0,b} \\
y_{r,1,b} + y_{r,1,b+61} + y_{r,1,b+39} + x_{r+1,1,b} &= 2 \cdot u_{r,1,b} \\
y_{r,2,b} + y_{r,2,b+1} + y_{r,2,b+6} + x_{r+1,2,b} &= 2 \cdot u_{r,2,b} \\
y_{r,3,b} + y_{r,3,b+10} + y_{r,3,b+17} + x_{r+1,3,b} &= 2 \cdot u_{r,3,b} \\
y_{r,4,b} + y_{r,4,b+7} + y_{r,4,b+41} + x_{r+1,4,b} &= 2 \cdot u_{r,4,b}
\end{aligned}$$

Finally, at least one S-box needs to be active:

$$\sum_{w=0}^{4} x_{0,w,0} \geq 1$$

The model for linear cryptanalysis is essentially identical, except for different rotation values. This MILP can then be solved using an off-the-shelf linear optimization tool, such as CPLEX. Unfortunately, it turns out that the highly combinatorial nature of the problem is not well suited for linear solvers, and that SAT solvers are a better fit for this type of problem.

SAT solvers. For SAT solvers, we can model essentially the same description by using an extended modelling language, as is used by Satisfiability Modulo Theory (SMT) solvers. We used the constraint solver `STP` by Ganesh et al. [13] to translate a bitvector-based CVC model to conjunctive normal form (CNF). This CNF model can then be solved using a parallel SAT solver, such as Biere's `Treengeling` [3]. Instead of an optimization problem, the problem has to be phrased in terms of satisfiability; i.e., the questions is whether solutions below a specific bound exist.

Modelling the S-box only in terms of its branch number is not very effective for obtaining tight bounds. As a trade-off between the all-too-simplistic branch number model and the complex complete differential description of the S-box (differential distribution table), we chose the following approximation. The linear preprocessing and postprocessing part of the S-box can easily be modelled

exactly for both differential and linear cryptanalysis. The nonlinear core (equivalent to the Keccak S-box) is approximated, i.e., the model allows a few transitions that are not possible according to the differential or linear distribution table. For the differential model, we use the following word-wise constraint in terms of input difference words $a_0, \ldots, a_4 \in \mathbb{F}_2^{64}$ and output difference words $b_0, \ldots, b_4 \in \mathbb{F}_2^{64}$:

$$b_i = a_i \oplus ((a_{i+1} \vee a_{i+2}) \wedge t_i), \qquad t_i \in \mathbb{F}_2^{64}, \quad i = 0, \ldots, 4.$$

For the linear model with word-wise linear input mask $a_0, \ldots, a_4 \in \mathbb{F}_2^{64}$ and output mask $b_0, \ldots, b_4 \in \mathbb{F}_2^{64}$, the constraints are similar:

$$a_i = b_i \oplus ((b_{i-1} \vee b_{i-2}) \wedge t_i), \qquad t_i \in \mathbb{F}_2^{64}, \quad i = 0, \ldots, 4.$$

With this model, we can easily prove that the 3-round Ascon permutation has at least 15 differentially active S-boxes (probability $\leq 2^{-30}$), and at least 13 linearly active S-boxes (bias $\leq 2^{-14}$, complexity $\geq 2^{28}$). The bounds on the number of active S-boxes are tight, but not necessarily those on the probability. Using these results, we can prove that the full 12-round initialization or finalization has at least 60 differentially active S-boxes (probability $\leq 2^{-120}$) and at least 52 linearly active S-boxes (bias $\leq 2^{-53}$, complexity $\geq 2^{106}$). These bounds are almost certainly not tight, but we were not able to derive bounds for more than 3 rounds using SAT solvers. This motivates the use of heuristic search tools to find explicit characteristics.

5.2 Differential and linear characteristics

In Table 4, we present an overview of our best differential and linear characteristics for different round numbers of the Ascon permutation. We have been able to improve the differential characteristic for 4 rounds of the Ascon permutation compared to the previous best results by the designers [11]. Since the designers included no results on linear cryptanalysis in the submission document, we provide the first linear analysis. When comparing the best differential characteristics with the best linear characteristics, we see that for more than two rounds of the Ascon permutation, the linear characteristics have fewer active S-boxes. This might indicate that Ascon is more vulnerable to linear cryptanalysis. Nevertheless, for 5 rounds of Ascon, the best found linear characteristic has more than 64 active S-boxes. Assuming the best possible bias for all active S-boxes, the attack complexity is already higher than 2^{128}.

5.3 Forgery attack on round-reduced Ascon

Usually, the characteristics from Section 5.2 cannot be directly used in an attack, since there might be additional requirements that the characteristic has to fulfill. In the case of an attack on the finalization of Ascon-128, suitable characteristics may only contain differences in stateword x_0 at the input of the permutation.

Table 4. Minimum number of active S-boxes for the Ascon permutation.

result	rounds	differential	linear
proof	1	1	1
	2	4	4
	3	15	13
heuristic	4	44	43
	≥ 5	> 64	> 64

The rest of the statewords have to be free of differences. For the output of the finalization, the only requirement is that there is some fixed difference pattern in x_3 and x_4. Knowledge about the expected differences in x_0, x_1, and x_2 at the output of the permutation is not required.

For round-reduced versions of Ascon, we have found suitable characteristics for a reduced 3-round finalization with a probability of 2^{-33} and for 4-round finalization with a probability of 2^{-101}. The used characteristic for the three round attack is given in Table 6 and the differential for the four round attack is given in Table 7 in Appendix A.

5.4 Differential-linear cryptanalysis

In differential-linear cryptanalysis, differential and linear characteristics are used together in an attack. This kind of analysis was introduced by Langford and Hellman [17]. Later on, it was demonstrated that this type of analysis is also suitable for cases where the differential and the linear part have a probability different from 1 [4, 16]. Differential-linear cryptanalysis is especially useful if the combined success probability of one short differential characteristic and one short linear characteristic is better than the probability of a longer linear or differential characteristic. One reason for such a behavior might be a bad diffusion for fewer rounds. For the attack to work, the individual probabilities of the two used characteristics have to be relatively high. According to Dunkelman et al. [12], the bias at the output of such a differential-linear characteristic is about $2pq^2$, where q is the bias of the linear part and p the probability of the differential characteristic. This results in a data complexity of $\mathcal{O}(p^{-2}q^{-4})$.

Outline of the attack. For Ascon-128, we can use differential-linear characteristics as key-stream distinguisher. Like for cube-tester (Section 4.3), we can target either the initialization in a nonce-respecting scenario, or the processing of the plaintext in a nonce-misuse scenario. Here, we focus on the initialization. Therefore, differences are only allowed in the nonce (x_3, x_4), whereas the linear active bits have to be observable and therefore must be in x_0.

Analysis of the initialization. We start with the analysis of a 4-round initialization and create a differential-linear characteristic for it. For the differential

part, we place two differences in the same S-box of round 1. With probability 2^{-2}, we have one active bit at the output of this S-box. The linear layer ensures that 3 S-boxes are active in the second round. Those 3 S-boxes have the difference at the same bit-position of their input. All 3 active S-boxes of round 2 have the same output pattern of 2 active bits with probability 2^{-3}. Due to the linear layer, we then have differences on 11 S-boxes of round 3. For the linear characteristic, we use a characteristic with one active S-box in round 4 and 5 active S-boxes in round 3. The bias of the linear characteristic is 2^{-8}. In addition, we place the S-boxes in a way that the linear active S-boxes in round 3 do not overlap with the 11 S-boxes that have differences at their inputs. The bias of the generated differential-linear characteristic is $2pq^2 = 2^{-20}$. In practice, we are only interested in the bias of the output bit for the specific differences at the input. Due to the vast amount of possible combinations of differential and linear characteristics that achieve these requirements, we expect a much better bias.

Practical evaluation of the bias. In the best case, we place differences in bit 63 of x_3 and x_4, and get a bias of 2^{-2} in bit 9 of x_0 on the output of the substitution layer of round 4. This is much better than the result of 2^{-20} that we obtained from the theoretical analysis. It is possible to combine multiple characteristics to also get to a bias of 2^{-2} in theory. However, we decided to reduce our differential-linear analysis to statistical tests, where we place differences at the input and try to measure a bias at the output bits. We think that this method is sufficient for practical attacks. For a 5-round initialization, we observe a bias of 2^{-10} on $x_0[16]$ (last substitution layer) for differences in $x_3[63]$, and $x_4[63]$. This bias can be improved to 2^{-9} if we only use nonces with the same sign of the difference (the concrete pairs for both $x_3[63]$ and $x_4[63]$ are either $(0, 1)$ or $(1, 0)$). In the case of a 6-round initialization, we were not able to observe a bias by using a set of 2^{36} inputs. The biases were averaged for randomly-chosen keys.

Observing key-dependency of the bias. As shown by Huang et al. [14], the bias observed at the output depends on the concrete values of secret and constant bits. They used this observation to recover the secret state of ICEPOLE in a nonce-misuse scenario. So we expect that a similar attack is possible on round-reduced versions of Ascon-128. In contrast to Huang et al., we want to recover the secret key directly and attack round-reduced versions of the initialization. This also transfers the attack to a nonce-respecting scenario. For a reduced initalization of 4 out of 12 rounds, we observed the bias patterns shown in Table 5. This table shows that the observable bias depends on the concrete values of two key bits which contribute to the same S-box as the used difference. Moreover, the bias is completely independent of the concrete value of the constant in x_0. This leads to the following straightforward attack.

Key-recovery attack on round-reduced Ascon. The target of this attack is a round-reduced version of Ascon-128, where the initialization is reduced to

Table 5. Bias of bit $x_0[i+1]$ in the S-box outputs of round 4 for differences in input bits $x_3[i]$ and $x_4[i]$ (2^{30} different inputs).

inputs $(x_1[i], x_2[i])$	key-bit pair	$(0,0)$	$(0,1)$	$(1,0)$	$(1,1)$
output $x_0[i+1]$	sign	$+1$	-1	$+1$	-1
	bias	$2^{-2.68}$	$2^{-3.68}$	$2^{-3.30}$	$2^{-2.30}$

4 out of 12 rounds. In this setting, the attacker has the ability to choose the nonce and is able to observe the resulting key stream. The attacker performs a sufficient amount of queries, with pairs of nonces which have differences in $x_3[63]$ and $x_4[63]$, and calculates the bias of $x_0[0]$ of the key-stream. With the help of Table 5, the attacker is able to recover two bits of the key by matching the expected bias with his calculated bias. Since the characteristics of Ascon are rotation-invariant within the 64-bit words, the same method can be used to recover the other key bits by placing differences in bits i and observing the bias at position $i+1 \mod 64$. Already 2^{12} samples per bit position i are sufficient to get stable results. This results in an expected time complexity of 2^{18} for the key-recovery attack on 4 rounds. However, in practice, we use the bias of all the bits and compute the correlation with the results of a precomputation (fingerprinting) phase to get better results. This way, we were also able to mount a key-recovery attack on the initialization of Ascon-128 reduced to 5 out of 12 rounds. In particular, we can reliably recover all key-bit pairs with values $(0,0)$ and $(1,1)$ with a low complexity of 2^{36}. However, we need to brute-force the other pairs, which results in an additional complexity of 2^{32} on average and 2^{64} in the worst case. Thus, the expected attack complexity is about 2^{36}. The complexities of both attacks on 4 and 5 rounds of the initialization have been practically verified.

Acknowledgments. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. The work has been supported in part by the Austrian Science Fund (project P26494-N15) and by the Austrian Research Promotion Agency (FFG) and the Styrian Business Promotion Agency (SFG) under grant number 836628 (SeCoS).

References

1. Aumasson, J.P., Meier, W.: Zero-sum distinguishers for reduced Keccak-f and for the core functions of Luffa and Hamsi. CHES rump session (2009)
2. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak Specifications. Submission to NIST (Round 3) (2011), `http://keccak.noekeon.org`
3. Biere, A.: Lingeling, Plingeling and Treengeling entering the SAT Competition 2013. In: Balint, A., Belov, A., Heule, M., Järvisalo, M. (eds.) SAT Competition 2013. vol. B-2013-1, pp. 51–52 (2013), `http://fmv.jku.at/lingeling/`

4. Biham, E., Dunkelman, O., Keller, N.: Enhancing Differential-Linear Cryptanalysis. In: Zheng, Y. (ed.) Advances in Cryptology – ASIACRYPT 2002. LNCS, vol. 2501, pp. 254–266. Springer (2002)
5. Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. In: Menezes, A., Vanstone, S.A. (eds.) Advances in Cryptology – CRYPTO 1990. LNCS, vol. 537, pp. 2–21. Springer (1990)
6. Boura, C., Canteaut, A.: A zero-sum property for the Keccak-f permutation with 18 rounds. In: IEEE International Symposium on Information Theory. pp. 2488–2492. IEEE (2010)
7. Boura, C., Canteaut, A., Cannière, C.D.: Higher-order differential properties of keccak and *Luffa*. In: Joux, A. (ed.) Fast Software Encryption – FSE 2011. LNCS, vol. 6733, pp. 252–269. Springer (2011)
8. Daemen, J.: Permutation-based Encryption, Authentication and Authenticated Encryption. DIAC – Directions in Authenticated Ciphers (2012)
9. Dinur, I., Morawiecki, P., Pieprzyk, J., Srebrny, M., Straus, M.: Cube Attacks and Cube-attack-like Cryptanalysis on the Round-reduced Keccak Sponge Function. IACR Cryptology ePrint Archive 2014, 736 (2014), `http://eprint.iacr.org/2014/736`
10. Dinur, I., Shamir, A.: Cube Attacks on Tweakable Black Box Polynomials. In: Joux, A. (ed.) Advances in Cryptology – EUROCRYPT 2009. LNCS, vol. 5479, pp. 278–299. Springer (2009)
11. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon. Submission to the CAESAR competition: `http://ascon.iaik.tugraz.at` (2014)
12. Dunkelman, O., Indesteege, S., Keller, N.: A Differential-Linear Attack on 12-Round Serpent. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) Progress in Cryptology – INDOCRYPT 2008. LNCS, vol. 5365, pp. 308–321. Springer (2008)
13. Ganesh, V., Dill, D.L.: A decision procedure for bit-vectors and arrays. In: Computer Aided Verification (CAV '07). Springer (2007), `https://sites.google.com/site/stpfastprover/`
14. Huang, T., Wu, H., Tjuawinata, I.: Practical State Recovery Attack on ICEPOLE, `http://www3.ntu.edu.sg/home/huangtao/icepole/icepole_attack.pdf`
15. Jovanovic, P., Luykx, A., Mennink, B.: Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014. LNCS, vol. 8873, pp. 85–104. Springer (2014), `http://dx.doi.org/10.1007/978-3-662-45611-8_5`
16. Langford, S.K.: Differential-linear cryptanalysis and threshold signatures. Ph.D. thesis, Stanford University (1995)
17. Langford, S.K., Hellman, M.E.: Differential-linear cryptanalysis. In: Desmedt, Y. (ed.) Advances in Cryptology – CRYPTO 1994. LNCS, vol. 839, pp. 17–25. Springer (1994)
18. Matsui, M., Yamagishi, A.: A New Method for Known Plaintext Attack of FEAL Cipher. In: Rueppel, R.A. (ed.) Advances in Cryptology – EUROCRYPT 1992. LNCS, vol. 658, pp. 81–91. Springer (1992)
19. National Institute of Standards and Technology: FIPS PUB 180-4: Secure Hash Standard. Federal Information Processing Standards Publication 180-4, U.S. Department of Commerce (March 2012), `http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf`
20. The CAESAR committee: CAESAR: Competition for authenticated encryption: Security, applicability, and robustness (2014), `http://competitions.cr.yp.to/caesar.html`

A Differentials to create forgery

Table 6 contains the differential characteristic and Table 7 contains the differential used for the forgery attacks of Section 5.3. One column corresponds to the five 64-bit words of the state, and the xor differences are given in hexadecimal notation (truncated in the last round).

Table 6. Differential characteristic to create forgery for round-reduced Ascon-128 with a 3-round finalization. The differential probability is 2^{-33}.

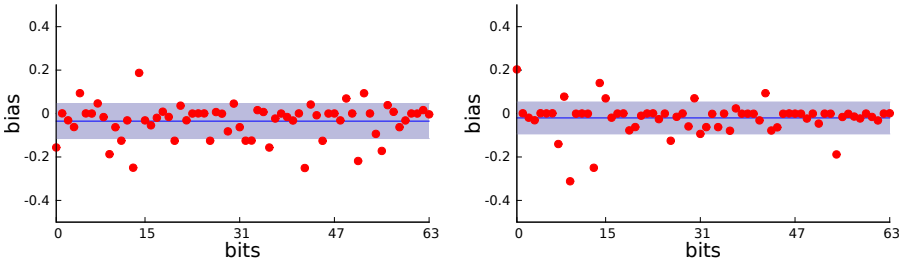
	input difference		after 1 round		after 2 rounds		after 3 rounds
x_0	8000000000000000		8000100800000000		8000000002000080		????????????????
x_1	0000000000000000		8000000001000004		9002904800000000		????????????????
x_2	0000000000000000	$\rightarrow$	0000000000000000	$\rightarrow$	d200000001840006	$\rightarrow$	????????????????
x_3	0000000000000000		0000000000000000		0102000001004084		4291316c5aa02140
x_4	0000000000000000		0000000000000000		0000000000000000		090280200302c084

Table 7. Differential to create forgery for round-reduced Ascon-128 with a 4-round finalization. The differential probability is 2^{-101}.

	input difference		after 4 rounds
x_0	8000000000000000		????????????????
x_1	0000000000000000		????????????????
x_2	0000000000000000	$\rightarrow$	????????????????
x_3	0000000000000000		280380ec6a0e9024
x_4	0000000000000000		eb2541b2a0e438b0

B Differential-linear key recovery attack on 4 rounds

Fig. 2 illustrates the observed bias in bit $x_0[i]$ in the key-stream for the differential-linear attack of Section 5.4, grouped by the values of the key-bit pair $(x_1[63], x_2[63])$.



(a) key (0,0) (b) key (1,1)

(c) key (0,1) (d) key (1,0)

Fig. 2. Biases for the differential-linear attack on the initialization of Ascon reduced to 4 (out of 12) rounds for the key-bit pair values $(0,0)$, $(0,1)$, $(1,0)$, $(1,1)$.

Square Attack on 7-Round Kiasu-BC

Publication Data

Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. "Square Attack on 7-Round Kiasu-BC". In: *Applied Cryptography and Network Security, ACNS 2016*. Ed. by Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider. Vol. 9696. LNCS. Springer, 2016, pp. 500–517. URL: `https://doi.org/10.1007/978-3-319-39555-5_27`

The appended paper is an author-created version available at `https://eprint.iacr.org/2016/326`.

Contributions

- One of the main authors.

Square Attack on 7-Round Kiasu-BC

Christoph Dobraunig, Maria Eichlseder, and Florian Mendel

Graz University of Technology, Austria
christoph.dobraunig@iaik.tugraz.at

Abstract. Kiasu-BC is a tweakable block cipher presented within the TWEAKEY framework at AsiaCrypt 2014. Kiasu-BC is almost identical to AES-128, the only difference to AES-128 is the tweak addition, where the 64-bit tweak is xored to the first two rows of every round-key. The security analysis of the designers focuses primarily on related-key related-tweak differential characteristics and meet-in-the-middle attacks. For other attacks, they conclude that the security level of Kiasu-BC is similar to AES-128. In this work, we provide the first third-party analysis of Kiasu-BC. We show that we can mount Square attacks on up to 7-round Kiasu-BC with a complexity of about $2^{48.5}$ encryptions, which improves upon the best published 7-round attacks for AES-128. Furthermore, we show that such attacks are applicable to the round-reduced ΘCB3-like mode of the CAESAR candidate Kiasu. To be specific, we show a key-recovery attack on 7-round Kiasu$\neq$ with a complexity of about 2^{82} encryptions.

Keywords: Cryptanalysis · TWEAKEY · Kiasu · Square Attack

1 Introduction

In contrast to standard block ciphers, tweakable block ciphers provide an additional input called tweak. This tweak is usually public and is used to select one specific instance of the block cipher. The concept of tweakable block ciphers was first formalized by Liskov et al. [15, 16]. Since then, tweakable block ciphers have proven to be a valuable building block of cryptographic schemes for various applications, like encryption, authentication, or authenticated encryption. For example, several of the authenticated encryption schemes in the ongoing CAESAR competition [19] are based on tweakable block ciphers [8, 12, 13].

Recently, Jean et al. presented the TWEAKEY framework [10] for designing tweakable block ciphers. The extended version of their paper [11] specifies three instances: Deoxys-BC, Joltik-BC, and Kiasu-BC. Kiasu-BC is a tweakable variant of AES-128, accepting a 64-bit tweak T in addition to the 128-bit key and 128-bit data block. The specification of Kiasu-BC is essentially identical to AES-128, except that T is xored to the first two rows of every round key. Hence, Kiasu-BC exactly matches AES-128 if $T = 0$. This has several advantages. First of all, it allows easy reuse or updates of existing implementations of AES-128. Moreover, the trust of the industry and academia in AES-128 has been steadily

The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-39555-5_27

growing over the past years and it might be easier in practice to promote the use of AES-128 with slight modifications instead of proposing new tweakable block ciphers. Another advantage of the similarity of Kiasu-BC and AES-128 is that AES-128 has been very thoroughly analyzed due to its prominence and widespread adoption. Since Kiasu-BC corresponds to AES-128 if $T = 0$, existing and also new analysis results for AES-128 directly carry over to Kiasu-BC. However, it is not trivial to determine the effects of the tweak on the security of the design. Therefore, we provide—to the best of our knowledge—the first third-party analysis of Kiasu-BC.

The existing cryptanalysis of Kiasu-BC by its designers [9,11] focuses mainly on meet-in-the-middle attacks and related-key related-tweak differential attacks. The designers argue that the existing meet-in-the-middle attacks for AES-128 also apply to Kiasu-BC. Regarding related-key related-tweak differential characteristics, the designers were able to show that the minimum number of active S-boxes for 7 rounds of Kiasu-BC is 22 and thus, an upper bound for the probability is 2^{-132}. Since this bound is not tight, the designers conclude that Kiasu-BC suffers at most one round security loss compared to AES [9] in the framework of related-key related-tweak differential attacks. For the remaining types of attacks, the designers claim: *"As we keep the original round function and key schedule of AES, we believe that the security level of KIASU-BC against the remaining types of attacks stays the same"* [9]. In Table 1, we have listed some of these remaining attacks. The best-performing attacks that cover 7 rounds of AES-128 fall into the category of impossible differential and meet-in-the-middle attacks. Our goal is to find stronger attacks than these.

Table 1. Excerpt of best attacks on AES-128.

Rounds	Type	Data (CP)	Time	Ref
6	Partial sum	$2^{34.6}$	2^{44}	[6]
7	Partial sum	$2^{128-\varepsilon}$	2^{120}	[6]
7	Collisions	2^{32}	$2^{128-\varepsilon}$	[7]
7	Impossible differential	$2^{112.2}$	$2^{117.2}$ MA	[17]
7	Meet-in-the-middle	2^{80}	2^{123}	[4]
7	Impossible differential	$2^{106.2}$	$2^{110.2}$	[18]
7	Meet-in-the-middle	2^{97}	2^{99}	[5]

MA – memory accesses

All our attacks are based on the Square attack [1]. In the attack, a so-called Λ-set of 256 different plaintexts is observed during the encryption. In the case of AES, it is possible to construct 3-round distinguishers based on the Square property [2,3]. This leads to efficient 6-round key-recovery attacks on AES-128 by prepending 1 round and appending 2 rounds to the 3-round distinguisher [6]. To extend these attacks, we use the additional freedom introduced by the tweak of Kiasu-BC to create a Square-based distinguisher covering 4 rounds. This leads to

7-round attacks on Kiasu-BC (shown in Table 2), which are significantly better than the best published attacks on 7 rounds of AES-128 (see Table 1 for an overview of attacks on AES-128). Furthermore, we show that variants of our Square attack are also applicable to round-reduced variants of an authenticated encryption mode of the CAESAR candidate Kiasu [9]. To be more specific, we target a round-reduced variant of Kiasu≠, which uses 7-round Kiasu-BC in a ΘCB3-like [14] mode of operation. The attacks on round-reduced Kiasu≠ are performed in a nonce-respecting scenario, and also comply with the very low data complexity limits imposed by Kiasu≠.

Table 2. Dedicated attacks on round-reduced Kiasu-BC and Kiasu≠.

Target	Rounds	Type	Data (CP)	Time	Ref
Kiasu-BC	7/10	Square	2^{40}	2^{82}	4.1
	7/10	Square	$2^{43.6}$	$2^{48.5}$	4.2
Kiasu≠	7/10	Square	$2^{28} \times 2^{16}$	2^{82}	5.2

The remainder of the paper is organized as follows. First, we describe the design of Kiasu-BC in Section 2. Afterwards, we construct a 4-round distinguisher based on the Square attack (Section 3), followed by two key-recovery attacks on 7-round Kiasu-BC in Section 4. Next, we demonstrate the applicability of variants of the key-recovery attacks on the mode of operation Kiasu≠ in Section 5. Finally, we conclude in Section 6.

2 Description of Kiasu-BC

The tweakable block cipher Kiasu-BC was introduced as building block of the Kiasu authenticated cipher family [9], a candidate in the CAESAR competition [19]. Kiasu-BC is an instantiation of the TWEAKEY framework [10], a general construction framework for tweakable block ciphers. For each 128-bit key and public 64-bit tweak, Kiasu-BC defines a 128-bit permutation.

Kiasu-BC is essentially identical to AES, except that the 64-bit tweak value is xored to the state in each round after the round-key addition. Thus, like for AES, the 128-bit Kiasu-BC state S is represented as a 4×4 matrix of bytes, labeled $x_0, \ldots, x_{15}$:

$$S = \begin{array}{|c|c|c|c|} \hline x_0 & x_4 & x_8 & x_{12} \\ \hline x_1 & x_5 & x_9 & x_{13} \\ \hline x_2 & x_6 & x_{10} & x_{14} \\ \hline x_3 & x_7 & x_{11} & x_{15} \\ \hline \end{array} .$$

In each of Kiasu-BC's 10 rounds, the round operations SubBytes, ShiftRows, MixColumns and AddRoundTweakey are applied to the state in turn. Except for AddRoundTweakey, they are identical to the AES round operations:

- SubBytes: Applies the 8-bit AES S-box $\mathcal{S}$ to each of the 16 state bytes.
- ShiftRows: Rotates row i of the state, $0 \leq i \leq 3$, by i bytes to the left.
- MixColumns: Multiplies each byte column of the state by the MDS-matrix M over $\mathbb{K} = \mathbb{F}_2[\alpha]/(\alpha^8 + \alpha^4 + \alpha^3 + \alpha + 1)$,

$$M = \begin{pmatrix} \alpha & \alpha+1 & 1 & 1 \\ 1 & \alpha & \alpha+1 & 1 \\ 1 & 1 & \alpha & \alpha+1 \\ \alpha+1 & 1 & 1 & \alpha \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$$

- AddRoundTweakey: In round i, xors the 128-bit round key RK_i and the tweak T to the state, where

$$\mathsf{RK}_i = \begin{array}{|c|c|c|c|} \hline \mathsf{RK}_{i,0} & \mathsf{RK}_{i,4} & \mathsf{RK}_{i,8} & \mathsf{RK}_{i,12} \\ \hline \mathsf{RK}_{i,1} & \mathsf{RK}_{i,5} & \mathsf{RK}_{i,9} & \mathsf{RK}_{i,13} \\ \hline \mathsf{RK}_{i,2} & \mathsf{RK}_{i,6} & \mathsf{RK}_{i,10} & \mathsf{RK}_{i,14} \\ \hline \mathsf{RK}_{i,3} & \mathsf{RK}_{i,7} & \mathsf{RK}_{i,11} & \mathsf{RK}_{i,15} \\ \hline \end{array}, \qquad T = \begin{array}{|c|c|c|c|} \hline T_0 & T_2 & T_4 & T_6 \\ \hline T_1 & T_3 & T_5 & T_7 \\ \hline & & & \\ \hline & & & \\ \hline \end{array}.$$

  We omit the details of the AES key schedule that derives the round subkeys RK_i from the key K, since they are not relevant for our attack. Note that there is no tweak schedule, i.e., the same tweak T is xored in each round. So for the all-zero tweak $T = 0$, Kiasu-BC is equivalent to AES-128.

To refer to intermediate states of Kiasu-BC, we denote by S_i the state after i rounds: $S_0 = P \oplus T \oplus \mathsf{RK}_0$, S_1, ..., $S_{10} = C$. In addition, the state after SubBytes of round i is denoted S_i^{SB}, after ShiftRows S_i^{SR}, after MixColumns S_i^{MC}, and after AddRoundTweakey $S_i^{\mathsf{AK}} = S_i$. So the states of full-round Kiasu-BC are

$$\begin{aligned} P \xrightarrow{\mathsf{AK}} S_0 \xrightarrow{\mathsf{SB}} S_1^{\mathsf{SB}} \xrightarrow{\mathsf{SR}} S_1^{\mathsf{SR}} \xrightarrow{\mathsf{MC}} S_1^{\mathsf{MC}} \xrightarrow{\mathsf{AK}} S_1 \\ \vdots \\ S_9 \xrightarrow{\mathsf{SB}} S_{10}^{\mathsf{SB}} \xrightarrow{\mathsf{SR}} S_{10}^{\mathsf{SR}} \xrightarrow{\mathsf{AK}} S_{10} = C \ . \end{aligned}$$

3 Distinguisher for 4 rounds of Kiasu-BC

The distinguisher presented in this section is based on the Square attack. This attack, originally demonstrated for the block cipher Square [1], is also applicable to AES [2, 3]. As in the Square attack on AES, we will observe a Λ-set of 256 different plaintexts through the encryption. By making use of the tweak input

of Kiasu-BC, we show that a distinguisher for 4 rounds can be created. This is one round more than the distinguisher used in the Square attack on AES. Before giving the distinguisher, we recall the effect of the round functions of AES on Λ-sets.

3.1 Preliminaries

For the Square attack, we will make statements about the 256 values for single byte positions x_i of a Λ-set. We index the individual byte value of byte position i in Λ-set element k as $x_i[k]$, where the index k is in the range from 0 to 255. We call a byte of a Λ-set active (`A`) if it takes all possible 256 values; constant (`C`) if all 256 values are equal; balanced (`B`) if the sum of all 256 values is 0; or unknown (`?`) if we cannot make any statements about the 256 values for this byte position.

SubBytes. SubBytes affects each byte of the state individually. Therefore, we can put our focus on the effects of the S-box on our four different byte states: active, constant, balanced, and unknown. The AES S-box is a permutation. Hence, if the input of the S-box iterates over all 256 possible values, then so will the output. Thus, an active byte remains active after SubBytes. Since the AES S-box is deterministic, a certain value at the input of the S-box will always map to the same value at the output. This means a constant byte remains constant after SubBytes. However, a balanced byte becomes unknown, because the S-box is non-linear. An unknown byte remains, of course, unknown.

ShiftRows. The ShiftRows operation works on byte-level. To be more concrete, it simply reorders the bytes of the state. Hence, our statements about the bytes remain the same, just the position differs after ShiftRows.

MixColumns. MixColumns is a linear transformation that mixes the single bytes of one column. Clearly, an all-constant input set will be mapped to an all-constant output set. Furthermore, if at least one of the input byte positions of the set is unknown, the entire output will be unknown.

Since MixColumns is based on an MDS matrix, it has a branch number of 5. This implies that if two input columns differ only in one byte, the output will differ in all 4 bytes. In particular, if the 4 input byte positions of a set are all constant except for one active byte, then all output bytes will be active. (Assume that one byte is not active, but takes one particular value twice. The corresponding pair of inputs will have a difference in only 1 input byte and at most 3 output bytes, violating the branch number property.) The same reasoning also clearly applies for the inverse operation of MixColumns.

AddRoundTweakey. Here, the specific round key as well as the tweak are xored to the state. Our attacks are performed in the single-key setting, so each key

byte is constant. This means that an active byte of the state remains active, a constant byte constant, a balanced byte balanced (since the constant key is added an even number of times and cancels out), and an unknown byte remains unknown.

The situation changes if we take a look at the tweak addition. For the distinguisher, we want to use Λ-sets where one byte of the tweak is active, so we have to consider the following situations. The xor of an active byte with an active byte definitely results in a balanced byte. If the tweak byte as well as the state byte are active and $T_i[k] \oplus x_i[k] = c$ for each k, the byte gets constant. The xor of an active byte with a balanced byte results in a balanced byte.

3.2 The 4-round distinguisher

The distinguisher used in the Square attack against AES [2, 3] spans over 3 rounds. It starts with a Λ-set that is active in one byte of the plaintext and constant in the rest of the state. The distinguisher ends after the key addition of the third round with an all-balanced state. By introducing an active tweak byte, we are able to extend the distinguisher by one round. However, the condition we get after round 4 is slightly more difficult to exploit (see Fig. 1).

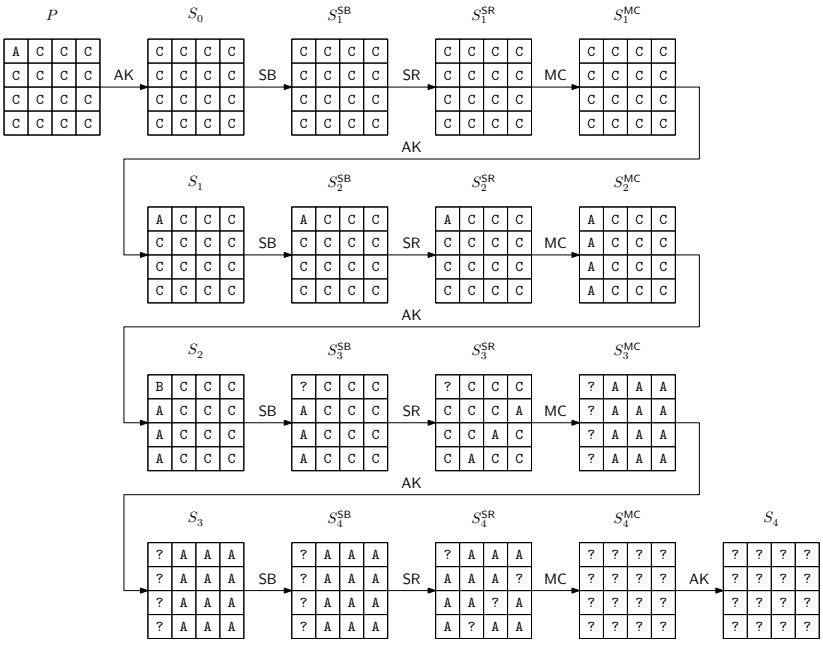


Fig. 1. Distinguisher for 4 rounds of Kiasu-BC.

As shown in Fig. 1, we start with a Λ-set of 256 plaintexts P, where one byte is active and the others remain constant. Additionally, we require that byte T_0 of

the tweak is active as well. Since always the same tweak is xored to every round key, every resulting round key xored with the tweak can be described as a Λ-set that is active at byte 0 and constant in the rest of the bytes. The tweak and plaintext values have to be chosen in a way that the xor of the tweak and the plaintext is constant. For instance, $T_0[k]$ can always be chosen to be equal to the first byte of the plaintext $x_0[k]$ for all 256 values of k. In this way, it is ensured that state S_0 is constant at every byte position. The state remains constant until S_1, where byte x_0 becomes active again due to the addition of the tweak.

The second round of our distinguisher for Kiasu-BC corresponds to the first round of the distinguisher used in the AES Square attack, except for the addition of the active tweak byte at the end. Since SubBytes and ShiftRows affect neither active nor constant bytes, we get to state S_2^{SR}, where still only the byte at position 0 remains active. The rest of the state is still constant. The next MixColumns operation leads to an active column in state S_2^{MC}. In contrast to the first tweak addition, the tweak addition at the end of round 2 leads to a balanced byte at position 0. We get a balanced byte here, because we cannot make any assumption on the concrete ordering of the 256 values of x_0 of state S_2^{MC}.

In the third round, we have one balanced byte before SubBytes. This byte becomes unknown after the S-box application. The ShiftRows operation shifts the active bytes away from the first column. So we have at state S_3^{SR} one unknown, and three constant bytes in the first column and one active, and 3 constant bytes in every other column. This leads to one completely unknown first column, and three completely active columns in state S_3^{MC}. The next tweak addition does not change anything.

For the fourth round, we only go with active or unknown bytes through the S-box layer, thus SubBytes does not influence our knowledge about the Λ-set at this point. ShiftRows shifts one unknown byte to every column, so we get a completely unknown state S_4^{MC} if we only limit our view to single byte positions. Hence, we have to take a closer look at the MixColumns operation. To do so, we represent the bytes of S_4^{SR} as x_i and the bytes of S_4^{MC} as y_i. Now, let us take a look at what happens if we xor y_1 with y_2:

$$\begin{aligned} y_1 \oplus y_2 &= \mathsf{01}{\cdot}x_0 \oplus \mathsf{02}{\cdot}x_1 \oplus \mathsf{03}{\cdot}x_2 \oplus \mathsf{01}{\cdot}x_3 \oplus \mathsf{01}{\cdot}x_0 \oplus \mathsf{01}{\cdot}x_1 \oplus \mathsf{02}{\cdot}x_2 \oplus \mathsf{03}{\cdot}x_3 \\ &= \mathsf{03}{\cdot}x_1 \oplus \mathsf{01}{\cdot}x_2 \oplus \mathsf{02}{\cdot}x_3 \end{aligned} \tag{1}$$

As shown in (1), x_0 cancels and thus does not influence $y_1 \oplus y_2$. In the first column of S_4^{SR}, x_0 is the only byte which is unknown. The rest of the bytes are active. Since (1) only contains active coefficients, $y_1 \oplus y_2$ is balanced. The next key and tweak addition is an addition with constant bytes. This addition with constant values does not influence the balanced property and therefore, also the xor of byte 1 and 2 of state S_4 is balanced.

4 Attacking 7 Rounds of Kiasu-BC

For attacking 7 rounds of Kiasu-BC, we extend the distinguisher by one round in the backward and two rounds in the forward direction. At first we present a

basic version of the attack. Then, we improve the attack by using partial sums in a similar way as Ferguson et al. [6].

4.1 Basic Square attack

The key-recovery attack is based on a set of plaintexts with differences only on one of the diagonals of the state, combined with a set of tweaks with differences only in the top left byte T_0. Fig. 2 shows the trail we use to attack 7 rounds of Kiasu-BC, where rounds 2 to 5 correspond to the distinguisher explained in Section 3. To perform this attack, we first collect the encryption of all 2^{32} plaintexts P where the diagonal bytes $(x_0, x_5, x_{10}, x_{15})$ loop through all possible values, whereas the remaining 12 bytes are fixed to some constant. Each of these plaintexts is encrypted under all 2^8 possible tweaks where all bytes except T_0 are fixed to some constant, and T_0 loops through all values. Thus, in total, we require the ciphertext for $2^8 \cdot 2^{32} = 2^{40}$ plaintext-tweak combinations.

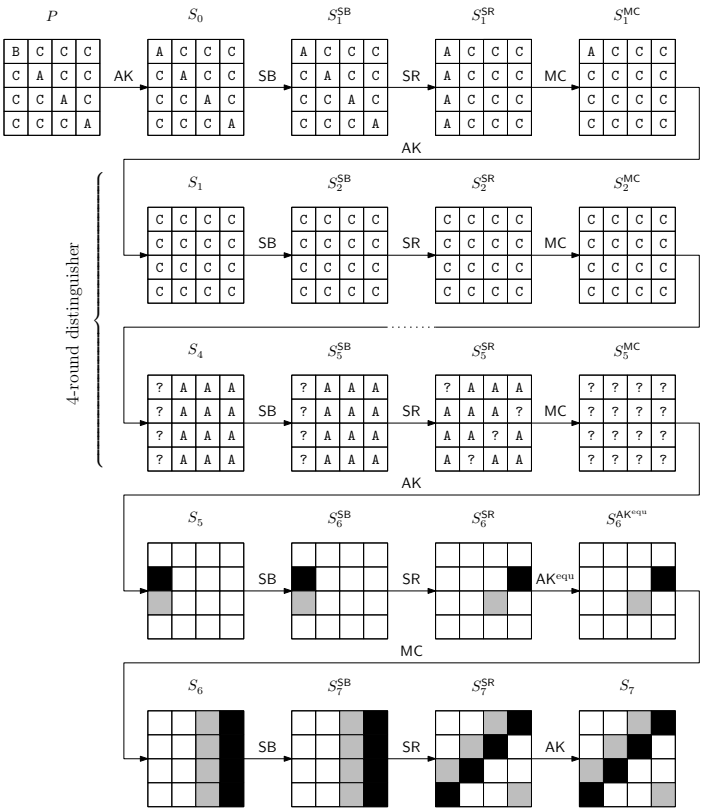


Fig. 2. Square attack for 7 rounds of Kiasu-BC.

Building Λ-sets. Next, we want to group this data into suitable Λ-sets, so that the previously introduced distinguisher can be applied to state S_1. This has to be done separately for each possible key guess of the 32 key bits $\mathsf{RK}_{0,0}$, $\mathsf{RK}_{0,5}$, $\mathsf{RK}_{0,10}$, and $\mathsf{RK}_{0,15}$, which determine the values of the first column of state S_1^{MC}. What we want to achieve is that this first column has only 1 active byte in x_0, and that this activity is canceled by AddRoundTweakey. Thus, we can fix the 3 constant bytes x_1, x_2, x_3 in S_1^{MC} to some arbitrary value, and set $x_0 = T_0$ for each of the 2^8 tweaks. If we decrypt these 2^8 set elements by 1 round, MixColumns will produce 4 active S-boxes in S_1^{SR}, which will be shifted to active S-boxes in x_0, x_5, x_{10}, x_{15} in state S_0. Depending on the different tweak bytes T_0 and the current key guess for the partial first-round key RK_0, we get a Λ-set of 2^8 plaintexts. We can repeat this procedure for a few different constant values in S_1^{MC} in order to build 16 Λ-sets for each of the 2^{32} key guesses of RK_0. For the correct key guess, all 16 Λ-sets will follow the 4-round distinguisher from Section 3.

Applying the distinguisher. We now want to partially decrypt all ciphertexts of each Λ-set back to state S_5, in order to verify the distinguishing property. Remember that we are interested in computing the xor sum $y_1 \oplus y_2$ of each Λ-set, marked in black (y_1) and gray (y_2) in Fig. 2. To do so, we have to calculate all intermediate values marked in black and gray in Fig. 2. We can do this for the black and the gray trail separately, requiring to guess 5 byte of key material for each trail. Note that we swapped the order of MixColumns and AddRoundTweakey in round 6, so that we only have to guess 1 byte of an equivalent round-key $\mathsf{RK}_6^{\mathrm{equ}} = \mathsf{MC}^{-1}(\mathsf{RK})$, rather than 4 bytes of the original RK_6.

We end up building two lists L_1 and L_2 (per key guess of RK_0). Each list has 2^{40} entries of 16-byte length each. For L_1, each entry represents the 16 xor sums of y_1 that result when decrypting the 16 Λ-sets for one guess of $\mathsf{RK}_{7,3}$, $\mathsf{RK}_{7,6}$, $\mathsf{RK}_{7,9}$, $\mathsf{RK}_{7,12}$, and $\mathsf{RK}_{6,13}^{\mathrm{equ}}$. In the case of L_2, each entry represents the 16 xor sums of y_2 that result when decrypting the 16 Λ-sets for one guess of $\mathsf{RK}_{7,2}$, $\mathsf{RK}_{7,5}$, $\mathsf{RK}_{7,8}$, $\mathsf{RK}_{7,15}$, and $\mathsf{RK}_{6,10}^{\mathrm{equ}}$.

As explained in Section 3, y_1 and y_2 of state S_5 sum to 0 for the correct key guess. Hence, we have to search for matching 16-byte entries between lists L_1 and L_2. A match indicates a key guess combination for the 10 guessed bytes of $\mathsf{RK}_6^{\mathrm{equ}}$, RK_7 that satisfies the distinguishing property for all 16 Λ-sets of one key guess for 4 bytes of RK_0; that is, a candidate for 14 bytes (or 112 bits) of key material for the correct key. The probability that a wrong key fulfills our 16-byte distinguisher is 2^{-128} (distinguishing property is the zero value for 128 bits, all other values reveal wrong keys). Therefore, we expect that only one candidate for the correct key bytes remains.

Attack complexity. To determine the overall complexity of this attack, we first take a look at the complexity per first-round key guess (guess of $\mathsf{RK}_{0,0}$, $\mathsf{RK}_{0,5}$, $\mathsf{RK}_{0,10}$, and $\mathsf{RK}_{0,15}$). To generate the 16 Λ-sets, we have to partially decrypt $16 \cdot 2^8$ plaintext-tweakey combinations for one round, for one column of the

state. This will allow us to select suitable Λ-sets from the 2^{40} chosen-plaintext queries encrypted under the target key. Then, we have to create our two lists L_1 and L_2. For creating one list, we have to decrypt $2^4 \cdot 2^8$ ciphertexts for 2^{40} key guesses 2 rounds back to one byte at S_5. Since we decrypt for 2 rounds to one byte, we only have to look at one column of the state. Hence, we estimate the costs for such a partial decryption with half a Kiasu-BC round. So, creating one list has approximately the complexity of $2^4 \cdot 2^8 \cdot 2^{40} = 2^{52}$ half-round decryptions, which corresponds to less than 2^{49} 7-round Kiasu-BC decryptions. For creating both lists, we require about 2^{50} 7-round Kiasu-BC decryptions. This complexity dominates both the complexity of 2^{12} one-round encryptions for creating the 16 Λ-sets and the complexity for finding a match between the two lists, which is approximately $40 \cdot 2^{40}$ comparison operations for sorting one list and $40 \cdot 2^{40}$ memory look-ups for finding a match.

Since we have to build the two lists for each of the 2^{32} first-round key guesses, we end up having a total attack complexity of 2^{82} 7-round Kiasu-BC encryptions. For carrying out this attack, we have to query $2^8 \cdot 2^{32} = 2^{40}$ chosen plaintexts. In addition to the plaintext-ciphertext pairs, we have to store our two lists L_1 and L_2. One entry of the lists corresponds to the memory complexity of storing one plaintext. Thus, we have an additional memory requirement of roughly 2^{41} Kiasu-BC states.

4.2 Improvements using partial sums

Ferguson et al. [6] showed that the complexity of the Square attack on AES can be significantly improved by using the partial sum technique. Their first observation is that for AES, the effort of guessing the 32 bits of RK_0 can be traded for summing over larger sets (of all 2^{32} plaintexts, rather than only 2^8 Λ-set messages), thus reducing the complexity by a factor of 2^8. Then, as a second improvement, the increased number of operations necessary for evaluating the distinguisher can be rearranged into partial sums to significantly cut down the computational complexity. In this section, we will show that a similar reasoning applies to Kiasu-BC, and that the techniques of Ferguson et al. [6] can be adapted to improve the complexity of the attack on 7-round Kiasu-BC significantly.

Summing all messages. In the basic attack, we had to guess 4 bytes of the first round key RK_0 in order to select a suitable Λ-set of 2^8 plaintext-tweak combinations and apply the distinguisher. For such a Λ-set, which is characterized by a single active byte x_0 in state S_1^{MC} and a constant difference between this byte and tweak byte T_0 (e.g., $x_0 = T_0$), we know that in S_5, the values $y_1 \oplus y_2$ sum to 0. Clearly, the same distinguishing property also applies if we sum not just over one, but over several Λ-sets.

Now consider again our set of $2^8 \cdot 2^{32}$ plaintext-tweak combinations. This set can actually be grouped into $2^8 \cdot 2^{24}$ Λ-sets as follows. For every value of T_0, the state bytes x_0, x_1, x_2, x_3 in state S_1^{MC} take all 2^{32} values. Therefore, for each of the 2^{24} fixed constant values of x_1, x_2, x_3 and each fixed value $x_0 \oplus T_0$ in

state S_1^{MC}, we can find exactly 2^8 plaintext-tweak combinations that map to this state, where x_1, x_2, x_3 and $x_0 \oplus T_0$ are constant. Each of these 2^8 plaintext-tweak combinations fulfills our conditions for a Λ-set. Thus, if we sum over all plaintext-tweak combinations, we actually sum over many Λ-sets, so the distinguishing property for $y_1 \oplus y_2$ will apply – and we do not have to guess the round key RK_0 in order to evaluate it. In other words, we can trade guessing the 32 key bits of RK_0 for summing over 2^{40} instead of 2^8 messages. Unfortunately, in contrast to the original attack on AES [6], this first improvement described so far does not, by itself, decrease the attack complexity, since we have to sum over all values of T_0. However, as we will show next, this modified distinguisher can be evaluated in an optimized way by reorganizing the order of summation.

Adapting the distinguisher. To evaluate the distinguisher, we now need to decrypt our 2^{40} ciphertexts back to y_1 and y_2. To identify valid key candidates, we calculate the sum in y_1 for each key guess of $\mathsf{RK}_{7,3}$, $\mathsf{RK}_{7,6}$, $\mathsf{RK}_{7,9}$, $\mathsf{RK}_{7,12}$, and $\mathsf{RK}_{6,13}$, storing the result in L_1 (indexed by the key guess); and we do the same for y_2 in L_2, based on all guesses of $\mathsf{RK}_{7,2}$, $\mathsf{RK}_{7,5}$, $\mathsf{RK}_{7,8}$, $\mathsf{RK}_{7,15}$, and $\mathsf{RK}_{6,10}$. Since we guess in total 10 bytes of key material, a 1-byte distinguisher is not enough to filter all wrong key guesses. Hence, we repeat the whole procedure for a total of 12 collections (of 2^{40} ciphertexts each), so that L_1 and L_2 are in the end populated with 12-byte entries (and indexed by 5-byte key guesses). We expect only one 12-byte match between L_1 and L_2, providing us with the correct 10 bytes of key material.

We now want to optimize the costs for calculating the entries of L_1 and L_2, which dominate the overall runtime by making use of the partial-sum technique described by Ferguson et al. [6]. They show that the cost for computing the 2^{40} sums (for each key guess) of one byte located 2 AES rounds before the end (similar to our case, y_1 or y_2 of State S_5), using 2^{32} ciphertexts, can be reduced to approximately 2^{50} S-box applications. Assuming that one encryption under a new key is equivalent to 2^8 S-box applications, the overall cost is only about 2^{42} encryptions. In contrast to the original attack, we actually want to sum over 2^{40} values, and additionally have to consider the tweak input. However, it turns out that the original partial-sum technique can be adapted to allow this with no significant computational overhead.

First, observe that in each $\mathsf{AddRoundTweakey}$ step, the different values of T_0 only influence the first byte x_0 of the state; and in the $\mathsf{AddRoundTweakey}^{\mathrm{equ}}$ step that we apply in round 6, T_0 modifies the equivalent round key of the first column (state bytes x_0, x_1, x_2, x_3). As illustrated in Fig. 2, neither L_1 nor L_2 depend on these state bytes, so we do not need to know T_0 in order to partially decrypt. Second, note that for building L_1 (or L_2), we are only interested in 32 bits of each of the 2^{40} encrypted messages (per collection). Thus, instead of decrypting each message with each key guess, we can count how often each possible 32-bit value occurs among the encrypted messages, and then only decrypt based on each 32-bit value once. Furthermore, since the effects of two occurrences of the same 32-bit value will simply cancel out in the final xor-sum, it is sufficient

to count occurrences modulo 2. We can store the counters in a 2^{32}-bit vector $\delta^{\text{cccc}} = (\delta_0^{\text{cccc}}, \ldots, \delta_{2^{32}-1}^{\text{cccc}})$, indexed by the possible values $x = x_0 \| x_1 \| x_2 \| x_3$.

Equipped with these two observations, we can now directly apply Ferguson et al.'s partial-sum technique, which we summarize below.

Ferguson et al.'s partial sums [6]. Consider the byte y_1 we need to evaluate for one entry of L_1, i.e., the sum over the 2^{40} messages of one collection. If we denote the 4 relevant (black) ciphertext bytes of message i in state S_7 by $c_{i,0}, \ldots, c_{i,3}$ and the 5 guessed round-key bytes (after xoring the known tweak) by $k_0, \ldots, k_4$, and summarize the inverse SubBytes in round 7 and the constant multiplications by MixColumns in round 6 in the bytewise functions $\mathcal{S}_0, \ldots, \mathcal{S}_3$, then the value we want to compute is

$$\begin{aligned}\sigma &= \bigoplus_{i=0}^{2^{40}-1} \mathcal{S}^{-1}[\mathcal{S}_0[c_{i,0} \oplus k_0] \oplus \mathcal{S}_1[c_{i,1} \oplus k_1] \oplus \mathcal{S}_2[c_{i,2} \oplus k_2] \oplus \mathcal{S}_3[c_{i,3} \oplus k_3] \oplus k_4] \\ &= \bigoplus_{x=0}^{2^{32}-1} \delta_x^{\text{cccc}} \cdot \mathcal{S}^{-1}[\mathcal{S}_0[x_0 \oplus k_0] \oplus \mathcal{S}_1[x_1 \oplus k_1] \oplus \mathcal{S}_2[x_2 \oplus k_2] \oplus \mathcal{S}_3[x_3 \oplus k_3] \oplus k_4].\end{aligned}$$

To optimize this computation, we first count for every key guess of k_0 and k_1 the modulo-2 frequency of the values $(\mathcal{S}_0[c_{i,0} \oplus k_0] \oplus \mathcal{S}_1[c_{i,1} \oplus k_1], c_{i,2}, c_{i,3})$ and store it in the 2^{24}-bit vector δ^{scc}. This vector can easily be computed from δ^{cccc} as

$$\delta_{x_0,x_1,x_2}^{\text{scc}} = \bigoplus_{s=0}^{2^8-1} \delta_{s,\mathcal{S}_1^{-1}[x_0 \oplus \mathcal{S}_0[s \oplus k_0]] \oplus k_1, x_1, x_2}^{\text{cccc}} . \tag{2}$$

Similarly, after guessing k_2 and subsequently k_3, we can compute the frequency δ^{sc} of $(\mathcal{S}_0[c_{i,0} \oplus k_0] \oplus \mathcal{S}_1[c_{i,1} \oplus k_1] \oplus \mathcal{S}_2[c_{i,2} \oplus k_2], c_{i,3})$ (2^{16} entries) and then δ^{s} of $(\mathcal{S}_0[c_{i,0} \oplus k_0] \oplus \mathcal{S}_1[c_{i,1} \oplus k_1] \oplus \mathcal{S}_2[c_{i,2} \oplus k_2] \oplus \mathcal{S}_3[c_{i,3} \oplus k_3])$ (2^8 entries) via

$$\delta_{x_0,x_1}^{\text{sc}} = \bigoplus_{s=0}^{2^8-1} \delta_{s,\mathcal{S}_2^{-1}[x_0 \oplus s] \oplus k_2, x_1}^{\text{scc}} , \tag{3}$$

$$\delta_{x_0}^{\text{s}} = \bigoplus_{s=0}^{2^8-1} \delta_{s,\mathcal{S}_3^{-1}[x_0 \oplus s] \oplus k_3}^{\text{sc}} . \tag{4}$$

Finally, we guess k_4 and compute the desired result byte via

$$\sigma = \bigoplus_{s=0}^{2^8-1} \delta_s^{\text{s}} \cdot \mathcal{S}^{-1}[s \oplus k_4]. \tag{5}$$

The same procedure can be applied to compute the entries of L_2, and needs to be repeated for each of the 12 collections. Afterwards, L_1 and L_2 can be sorted and matched as before to identify the correct partial key for 10 bytes of key material. The remaining 6 bytes of key information can be recovered with a brute-force approach.

Overall complexity. The data complexity for the improved attack is $12 \cdot 2^{40} \approx 2^{43.6}$ chosen plaintext-tweak combinations. Per list and collection, we have the following complexity. The original 2^{32}-bit vector δ^{cccc} can be constructed with negligible overhead to each chosen-plaintext query. The 2^{24}-bit vector δ^{scc} is computed for 2^{16} key guesses, and requires $2 \cdot 2^8 \cdot 2^{24} = 2^{33}$ S-box lookups, so the computations of (2) contribute 2^{49} S-box lookups per list and collection. Similarly, computations (3), (4) and (5) contribute 2^{48} S-box lookups each. Overall, computing lists L_1 and L_2 require $2 \cdot 12 \cdot (2^{49} + 3 \cdot 2^{48}) \approx 2^{54.9}$ S-box lookups, or roughly $2^{46.9}$ 7-round encryptions.

Sorting the 2^{40} entries of L_1 and L_2 can be implemented, for example, with less than $40 \cdot 2^{40} \approx 2^{45.3}$ comparisons (worst-case) and $2 \cdot 2^{40.1}$ Kiasu-BC states of memory per list via MergeSort, or a total of $2^{46.3}$ comparisons and $2^{41.7}$ memory for both lists. Finding all matches between the sorted lists takes a negligible $2 \cdot 2^{40}$ comparisons (worst-case).

We expect to find only one match, and guessing the remaining 6 bytes of key information takes, in the worst case, 2^{48} encryptions (assuming that the known 10 bytes of key information can be combined efficiently). In total, the worst-case attack complexity is about $2^{48.5}$ 7-round Kiasu-BC encryptions, and requires about $2^{41.7}$ Kiasu-BC states of memory, and $2^{43.6}$ chosen-plaintext-tweak queries.

5 Application to Authenticated Cipher Kiasu≠

In this section, we show that variants of the previously presented Square attacks are applicable when Kiasu-BC is used in a ΘCB3-like [14] mode of operation. To be specific, we demonstrate the feasibility of a variant of the attack presented in the previous section on Kiasu≠. Kiasu≠ is one of two proposed modes of the CAESAR candidate Kiasu [9], which only claims security when used in a nonce-respecting way. Thus, the attacks presented in this section follow this restriction and never require the nonce to be equal for queries on the encryption oracle. Before describing the attack, we give a short description of Kiasu≠.

5.1 Description of Kiasu≠

Fig. 3 shows the plaintext processing part of the authenticated encryption scheme Kiasu≠. Here, each plaintext block P_i is encrypted with the help of Kiasu-BC using always a different value for its tweak. The tweak value is constructed by concatenating a 3-bit 0, the 32-bit nonce N and a 29-bit value representing the index i of the plaintext block P_i that is encrypted. To generate the tag T, the sum of the plaintext blocks is encrypted and xored with Auth, which is derived from processing the authenticated data.

5.2 A Key-Recovery Attack on Round-Reduced Kiasu≠

Our attack targets the encryption of the plaintexts blocks. For the attack to be carried out, we need an encryption oracle that encrypts plaintexts chosen

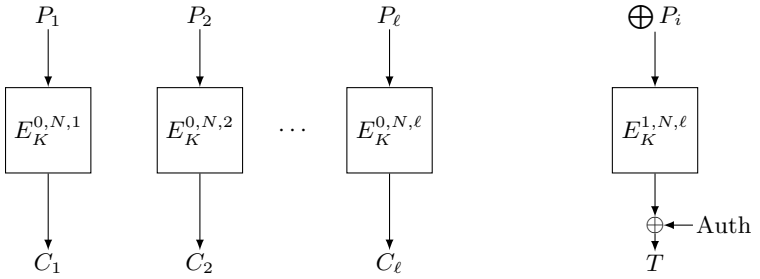


Fig. 3. Plaintext processing for the nonce-respecting mode Kiasu≠ for a multiple of the block length.

by the attacker. We use the block counter to iterate over the tweak byte T_7 to construct our Λ-sets. Since the least significant byte of the block counter is xored to byte 13 of the state, we have to use a slightly different distinguisher, which is shown in Fig. 4. Similar to the attacks presented in Section 4, we prepend one round to the distinguisher and append two rounds. Then, we can apply a slight modification of the Square attack described in Section 4.1.

The attack of Section 4.1 can be partitioned in two phases. The first one is the generation of 16 Λ-sets under a specific guess of 32-bits of RK_0, the second part is the evaluation of the Λ-sets to see if the distinguishing property holds for partial guesses of RK_6 and RK_7. While this evaluation of the Λ-sets works equivalent as in Section 4.1 for the attack on Kiasu≠, we have to change the way we built our Λ-sets. For building the Λ-sets, the attack of Section 4.1 uses the same 2^8 tweak values for every Λ-set. This is no longer an option, since the attacks on round-reduced Kiasu≠ are performed in a nonce-respecting setting. Therefore, we have to build each Λ-set using different tweak values and respecting the data limits of Kiasu≠, which limit the number of encrypted blocks per message to 2^{29}, and the total number of encrypted messages to 2^{32}. Next, we will describe how to select suitable plaintexts to obtain Λ-sets under these constraints.

Observe that for a single multi-block plaintext message, the tweaks used for encrypting the individual plaintext blocks will be constant in the first 35 bits, where 32 bits represent the nonce value. Dependent on the attack model, the nonce may be known before we make an encryption query (e.g., it is implemented as a counter, to avoid collisions of the very short nonces), or the oracle picks a random nonce. Note that one byte of the nonce at tweak position T_1 influences our key guess at $\mathsf{RK}_{0,1}$ in the upcoming attack. Hence, for sake of simplicity, we assume that the nonce value is known before we make each encryption query (we discuss the case of unpredictable nonces at the end of this section). The remaining bits of the tweak represent a 29-bit block counter and are always known in advance. In our attack we want to use the least significant 8 counter bits in T_7 for the active tweak byte. Since the counter starts with a value of 1, we actually can only start building Λ-sets from block 256 on. So the first Λ-set includes blocks $256, \ldots, 511$, i.e., $T_6 = 1$ and T_7 is active. Now, we need to define

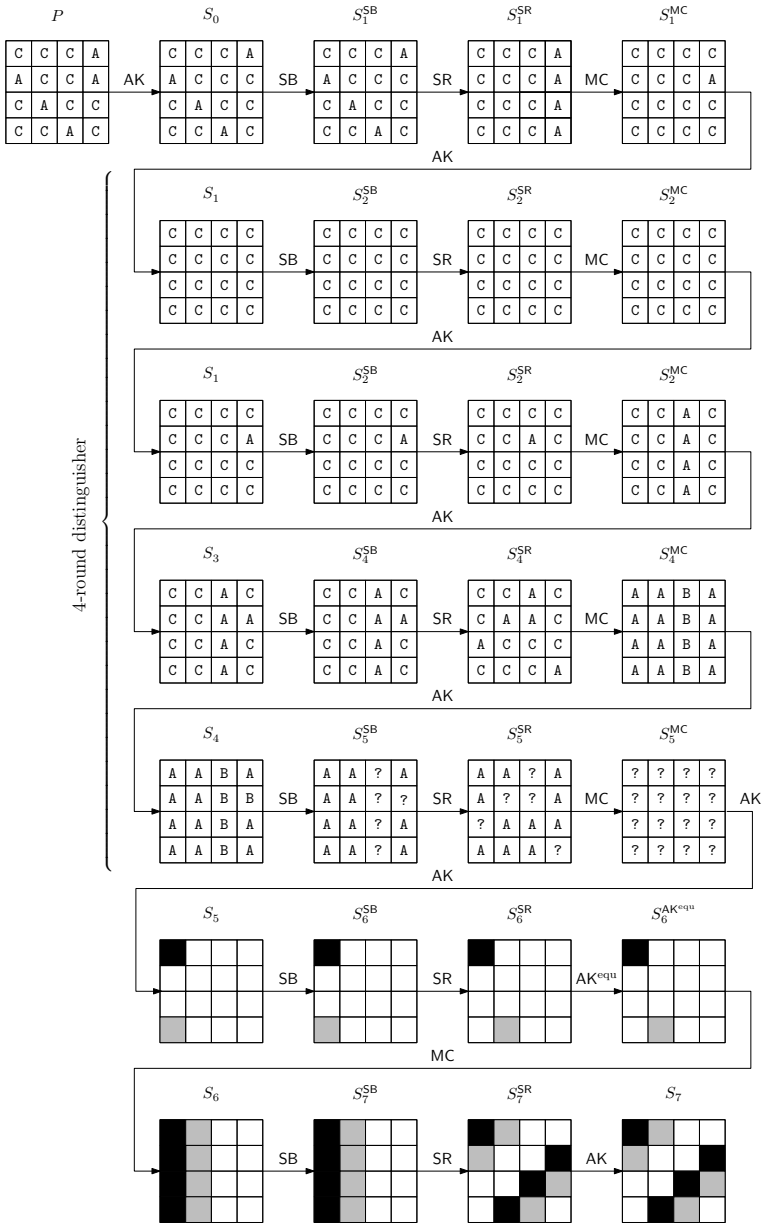


Fig. 4. Attack for 7 rounds of Kiasu≠.

suitable plaintext blocks to query, so that the ciphertext blocks of one 511-block message will allow us to evaluate the distinguisher.

Let p_i denote the individual state bytes of the plaintext, x_i the bytes of S_1^{MC}, and z_i the bytes of the state right after adding the tweak (but before adding the round key). We start by choosing some arbitrary constant value for the bytes $z_{12}, z_{13}, z_{14}, z_{15}$. Then, we apply the inverse tweak-addition to obtain $x_{12}, \ldots, x_{15}$, which will add a constant value of $T_6 = 1$ to z_{12}, and an active $T_7 = 0, \ldots, 255$ to z_{13}. The inverse first round will map this column to some set of states with 4 active bytes in S_0. For one key guess of $\mathsf{RK}_{0,1}$, $\mathsf{RK}_{0,6}$, $\mathsf{RK}_{0,11}$, and $\mathsf{RK}_{0,12}$, we obtain a set of 256 values for $(p_1, p_6, p_{11}, p_{12})$. The only other active byte, p_{13}, needs to be chosen so that the difference $p_{13} \oplus T_7$ is fixed, e.g., by setting $p_{13} = T_7$. The rest of the state can be chosen as some arbitrary constant. The resulting plaintext blocks have to be encrypted by the encryption oracle at block positions P_{256} to P_{511} and form a Λ-set for the right guess of $\mathsf{RK}_{0,1}$, $\mathsf{RK}_{0,6}$, $\mathsf{RK}_{0,11}$, and $\mathsf{RK}_{0,12}$.

The second part of the attack is evaluating the constructed Λ-sets. Since we changed the position of the active tweak byte from T_0 to T_7 compared to the original attack of Section 4.1, we also need to adapt the distinguishing property and evaluate, for instance, $y_0 \oplus y_3$ in state S_5, instead of $y_1 \oplus y_2$. The indices of the guessed round keys and ciphertext bytes need to be adapted accordingly, but otherwise, the attack procedure remains the same. This modification also has no influence on the attack runtime, so the computational complexity is still a total of 2^{82} encryptions to recover 12 bytes of key information.

Accomodating the data complexity limit. Note that with the above strategy, we would need to encrypt $16 \cdot 2^{32}$ messages to obtain 16 Λ-sets per 32-bit guess of RK_0. Thus, we would exceed the maximum number of messages that can be encrypted per key. However, it is possible and necessary to build more than one Λ-set per message following block 511, so that we do not exceed the maximum number of possible messages in our attack. Assume we construct 2^8 Λ-sets per message. This means the first Λ-set covers blocks $256, \ldots, 511$, so $T_6 = 1$ and T_7 is active, the second Λ-set covers blocks $512, \ldots, 767$, so $T_6 = 2$ and T_7 is active, and so on, until we have 2^8 Λ-sets. Thus, every message we query has a length of $2^{16} + 255$ blocks. This means we need 2^{28} chosen messages sent to the encryption oracle, corresponding to $2^{44} + 2^{36}$ chosen plaintext blocks for the attack.

Adaptation for unpredictable nonces. For simplicity, we assumed that the nonce value for each encryption query is predictable, since we needed the value of the nonce byte at tweak position T_1 in order to derive the plaintext values p_1 for each key guess of $\mathsf{RK}_{0,1}$. However, the attack can also be adapted for cases where the nonce is not known as follows. The attacker assumes $T_1 = 0$ and simply queries one message per guess of RK_0. The actual values of T_1 will be random, so for each value of $\mathsf{RK}_{0,6}, \mathsf{RK}_{0,11}, \mathsf{RK}_{0,12}$, the attacker effectively queried sets for 2^8 random values of $\mathsf{RK}_{0,1}$. Due to possible collisions, these queries will, on

average, cover a fraction of about $1 - \frac{1}{e} \approx 63.2\,\%$ of all 2^8 possible values of $\mathsf{RK}_{0,1}$. The attack is only successful if the correct value of $\mathsf{RK}_{0,1}$ is among the covered fraction, so the success probability of the overall attack will be about 63.2 %. This can be improved by asking several queries per key guess, e.g., 4 queries for a success probability of about $1 - \frac{1}{e^4} \approx 98.2\,\%$, at the cost of an increase in data complexity by a factor of 2^2 (but no increase in computational complexity).

An alternative, deterministic approach is to query 2^8 Λ-sets per guess of $\mathsf{RK}_{0,1}$, one for each possible value of T_1. All 2^8 Λ-sets need to be queried in one message, to get a constant nonce value and thus definitely cover the correct guess of T_1. Each message now contains $2^{24} + 255$ blocks, and we query a total of $2^{52} + 2^{36}$ blocks. Again, the computational complexity remains at 2^{82} encryptions.

6 Conclusion

In this work, we presented the first third-party analysis of Kiasu-BC. We showed that the additional tweak input can be exploited to create a distinguisher based on the Square property spanning 4 rounds. This is one more round compared to the distinguisher used in Square attacks on AES-128. Hence, we were able to perform key-recovery attacks on 7-round Kiasu-BC with a computational complexity of only about $2^{48.5}$ encryptions, which is faster than the best 7-round attacks for AES-128. However, we cannot attack more rounds compared to AES-128 and hence our analysis does not contradict the claim of the designers that Kiasu-BC has a sufficient security margin.

Variants of the Square attacks on Kiasu-BC are also applicable if Kiasu-BC is used in one of its recommended modes of operation. We demonstrated this with a nonce-respecting key-recovery attack on Kiasu$\neq$, a ΘCB3-like mode of the CAESAR candidate Kiasu. The computational complexity of this attack is approximately 2^{82} encryptions for 7-round Kiasu-BC, and the attack also respects the low data query limits.

Acknowledgements

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644052 (HECTOR).
Furthermore, this work has been supported in part by the Austrian Science Fund (project P26494-N15) and by the Austrian Research Promotion Agency (FFG) under grant number 845589 (SCALAS).

References

1. Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher Square. In: Biham, E. (ed.) Fast Software Encryption – FSE '97. LNCS, vol. 1267, pp. 149–165. Springer (1997)

2. Daemen, J., Rijmen, V.: AES proposal: Rijndael. National Institute of Standards and Technology (1998)
3. Daemen, J., Rijmen, V.: The Design of Rijndael: AES – The Advanced Encryption Standard. Information Security and Cryptography, Springer (2002)
4. Demirci, H., Taskin, I., Çoban, M., Baysal, A.: Improved meet-in-the-middle attacks on AES. In: Roy, B.K., Sendrier, N. (eds.) Progress in Cryptology – INDOCRYPT 2009. LNCS, vol. 5922, pp. 144–156. Springer (2009)
5. Derbez, P., Fouque, P., Jean, J.: Improved key recovery attacks on reduced-round AES in the single-key setting. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology – EUROCRYPT 2013. LNCS, vol. 7881, pp. 371–387. Springer (2013)
6. Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D.: Improved cryptanalysis of Rijndael. In: Schneier, B. (ed.) Fast Software Encryption – FSE 2000. LNCS, vol. 1978, pp. 213–230. Springer (2000)
7. Gilbert, H., Minier, M.: A collision attack on 7 rounds of Rijndael. In: AES Candidate Conference. pp. 230–241 (2000)
8. Grosso, V., Leurent, G., Standaert, F., Varici, K., Journault, A., Durvaux, F., Gaspar, L., Kerckhof, S.: SCREAM. Submission to the CAESAR competition: `http://competitions.cr.yp.to/round2/screamv3.pdf` (2015)
9. Jean, J., Nikolic, I., Peyrin, T.: KIASU. Submission to the CAESAR competition: `http://competitions.cr.yp.to/round1/kiasuv1.pdf` (2014)
10. Jean, J., Nikolic, I., Peyrin, T.: Tweaks and keys for block ciphers: The TWEAKEY framework. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology – ASIACRYPT 2014, Part II. LNCS, vol. 8874, pp. 274–288. Springer (2014)
11. Jean, J., Nikolic, I., Peyrin, T.: Tweaks and keys for block ciphers: the TWEAKEY framework. IACR Cryptology ePrint Archive 2014, 831 (2014), `http://eprint.iacr.org/2014/831`
12. Jean, J., Nikolic, I., Peyrin, T.: Deoxys. Submission to the CAESAR competition: `http://competitions.cr.yp.to/round2/deoxysv13.pdf` (2015)
13. Jean, J., Nikolic, I., Peyrin, T.: Joltik. Submission to the CAESAR competition: `http://competitions.cr.yp.to/round2/joltikv13.pdf` (2015)
14. Krovetz, T., Rogaway, P.: The software performance of authenticated-encryption modes. In: Joux, A. (ed.) Fast Software Encryption – FSE 2011. LNCS, vol. 6733, pp. 306–327. Springer (2011)
15. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. In: Yung, M. (ed.) Advances in Cryptology – CRYPTO 2002. LNCS, vol. 2442, pp. 31–46. Springer (2002)
16. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. J. Cryptology 24(3), 588–613 (2011)
17. Lu, J., Dunkelman, O., Keller, N., Kim, J.: New impossible differential attacks on AES. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) Progress in Cryptology – INDOCRYPT 2008. LNCS, vol. 5365, pp. 279–293. Springer (2008)
18. Mala, H., Dakhilalian, M., Rijmen, V., Modarres-Hashemi, M.: Improved impossible differential cryptanalysis of 7-round AES-128. In: Gong, G., Gupta, K.C. (eds.) Progress in Cryptology – INDOCRYPT 2010. LNCS, vol. 6498, pp. 282–291. Springer (2010)
19. The CAESAR committee: CAESAR: Competition for authenticated encryption: Security, applicability, and robustness (2014), `http://competitions.cr.yp.to/caesar.html`

ISAP – Towards Side-Channel Secure Authenticated Encryption

Publication Data

Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. "ISAP – Towards Side-Channel Secure Authenticated Encryption". In: *IACR Transactions on Symmetric Cryptology* 2017.1 (2017), pp. 80–105. URL: http://tosc.iacr.org/index.php/ToSC/article/view/585

Contributions

- **Technical:** Contributed to the idea and design of the mode, the sponge based instantiations (except the used re-keying functions), the choice of the parameters, and the analysis. No contributions to the implementations.
- **Writing:** Contributions to the writing of Sections 3, 4.1, 4.3, 4.5, 4.6, and 5. Minor contributions to the writing of Sections 1, 4.2, and 7.

ISAP – Towards Side-Channel Secure Authenticated Encryption

Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel and Thomas Unterluggauer

Graz University of Technology, Austria
firstname.lastname@iaik.tugraz.at

Abstract. Side-channel attacks and in particular differential power analysis (DPA) attacks pose a serious threat to cryptographic implementations. One approach to counteract such attacks are cryptographic schemes based on fresh re-keying. In settings of pre-shared secret keys, such schemes render DPA attacks infeasible by deriving session keys and by ensuring that the attacker cannot collect side-channel leakage on the session key during cryptographic operations with different inputs. While these schemes can be applied to secure standard communication settings, current re-keying approaches are unable to provide protection in settings where the same input needs to be processed multiple times.
In this work, we therefore adapt the re-keying approach and present a symmetric authenticated encryption scheme that is secure against DPA attacks and that does not have such a usage restriction. This means that our scheme fully complies with the requirements given in the CAESAR call and hence, can be used like other nonce-based authenticated encryption schemes without loss of side-channel protection. Its resistance against side-channel analysis is highly relevant for several applications in practice, like bulk storage settings in general and the protection of FPGA bitfiles and firmware images in particular.

Keywords: authenticated encryption · fresh re-keying · passive side-channel attacks · sponge construction · permutation-based construction

1 Introduction

Motivation. Passive side-channel attacks and in particular differential power analysis (DPA) pose a serious threat to the security of cryptographic implementations. These attacks allow to learn information about the secret key that is processed in a device by observing physical properties, like the power consumption [KJJ99] or the electromagnetic (EM) field [QS01]. They are a threat whenever a device performs cryptographic operations with a key that is not known to the holder of a device. This is the case, for example, when a sensor device is installed in a non-protected area to communicate data to some backend, when a manufacturer performs an encrypted firmware update on devices in the field, when a device working on encrypted data is lost, or when a device is rented by one party to another.

While passive side-channel attacks have mainly been a threat to ATM and pay TV cards at the time of their publication, these attacks are now relevant to a wide range of devices of the Internet of Things (IoT). A recent example is the IoT attack by Ronen et al. [ROSW16], where adjacent Philips Hue smart lamps infect each other with a worm that has the potential to control the device. One crucial part of this attack is the recovery of the global AES-CCM key that is used to encrypt and verify firmware updates with the help of a sophisticated DPA attack. As another prominent example, the keys for

IACR Transactions on Symmetric Cryptology ISSN 2519-173X, Vol. 2017, No. 1, pp. 80–105
DOI:10.13154/tosc.v2017.i1.80-105

FPGA bitfile encryption of several generations of FPGAs have been revealed by DPA attacks [MBKP11, MS16].

DPA attacks are the most powerful passive side-channel attacks in practice. They accumulate information about a cryptographic key by observing multiple en-/decryptions of different inputs. The fact that different inputs are used allows statistical techniques, like Bayesian distinguishers [CRR02] or correlation techniques [BCO04], to extract keys very efficiently.

While these attacks typically require a standard oscilloscope, there are now also open source projects for attack setups on software implementations [OC14]. Unprotected software implementations of cryptographic algorithms typically can be broken by observing less than 100 en- or decryptions with a key [MOP07]. Given the low effort of the attacks, there is great need for countermeasures.

State of the Art. In order to protect cryptographic keys against side-channel attacks, a lot of research has been conducted during the last two decades. Today, there essentially exist two approaches to counteract the attacks. The first approach works by hardening the implementation of cryptographic algorithms with techniques like hiding [MOP07] or masking [PR13]. The drawback of this approach is that the overhead for securing a cryptographic primitive against side-channel attacks is very high and depends on the cryptographic primitive itself. Therefore, in the past several ciphers have been proposed to reduce this cost. For example, the authenticated ciphers Ascon [DEMS14], Ketje/Keyak [BDP+14a, BDP+14b], PRIMATES [ABB+14], and SCREAM [GLS+14] of the ongoing CAESAR competition [CAE14] have all been designed with this goal in mind. However, the protected implementation of these designs still leads to a significant overhead and the cost of masking still increases significantly with the protection order [ISW03].

The second approach to counteract side-channel attacks is to change cryptographic protocols in such a way that certain types of side-channel attacks cannot be performed at all on the underlying cryptographic primitive. In particular, if the protocol design inherently prevents DPA attacks, the underlying cryptographic primitive only needs to be secured against attacks that extract information about the key by observing cryptographic operations for a single fixed input. Following the definitions in [MOP07], we refer to the class of attacks that require to observe a device processing the same or a few inputs as simple power analysis (SPA), whereas we refer to the class of attacks that require to observe a device processing many different inputs under the same key as differential power analysis (DPA). A protected implementation of the primitive against SPA attacks induces a significantly lower overhead than against DPA attacks. An example of such an approach of inherently preventing DPA attacks is fresh re-keying [MSGR10, MPR+11, BDH+14, DKM+15] and leakage-resilient cryptography, which brought forth encryption schemes [Pie09, FPS12], message authentication codes (MACs) [PSV15] and authenticated encryption schemes [BKP+16].

Schemes with inherent protection against DPA attacks require a side-channel secure initialization in order to obtain a fresh session key for every cryptographic operation. This session key is typically derived from a pre-shared master key using a nonce. The purpose of the secure initialization is to ensure that cryptographic operations for different data inputs are always done using different keys. Hence, whenever a party encrypts or authenticates data, a new nonce has to be generated to derive a new session key.

While this effectively prevents DPA attacks on the sender's encryption or authentication process, the situation is more challenging for the receivers who perform decryptions or verifications. While the sender can generate the nonce and thus ensure that session keys are always fresh, the receiver must process any data he receives, with no control over the nonce. In order to prevent DPA attacks in these cases, one possible approach is that all communicating parties contribute to the nonce that is used to derive the session key from

a pre-shared master key [MPR+11]. This prevents an attacker from collecting side-channel information for the decryption of several different ciphertexts under the same nonce (and thus the same session key). However, this approach requires additional communication or synchronization between parties, which is often not possible in practice.

Our Contribution. We propose ISAP, a symmetric authenticated encryption scheme that is designed to prevent DPA on both encryption and decryption. ISAP fulfills all functional requirements for nonce-based authenticated encryption as defined by the CAESAR call [CAE14] and at the same time provides protection against DPA attacks for all involved parties. In addition, ISAP limits the attack surface against decryption to SPA attacks and thus might be the first step towards a fully side-channel secure authenticated encryption scheme, addressing an open research problem mentioned by Pereira et al. [PSV15] and Berti et al. [BKP+16].

One of the main observations is that verifying authenticity before decryption protects the decryption procedure from DPA attacks, whereas the verification itself can be protected by a suitable derivation of the authentication session key. In addition, we show that sponges provide an elegant way to argue the resistance of permutation-based designs to SPA attacks. This flexibility motivates the fact that all building bocks of ISAP are based on sponges.

The results of our hardware implementation show that the concrete instances ISAP-128 and ISAP-128a (that are based on 400-bit KECCAK permutations) can be implemented in a straightforward manner with an area of 14 kGE, with the benefit compared to existing schemes that ISAP provides DPA security up to the same order as the used re-keying function even for multiple decryption.

Open Questions. ISAP protects against DPA and is designed to cope with limited SPA leakage. However, we still require dedicated countermeasures against SPA on implementation level. Such countermeasures are particularly crucial for the decryption unit, since the same data can be decrypted multiple times. This may reduce the measurement noise, making an SPA attack easier. Quantifying the SPA leakage of an implementation remains an open problem in practice. Another open question concerns the formal verification of our side-channel assumptions. While a security proof using state-of-the-art concepts of leakage-resilient cryptography might be out of reach, since ISAP allows multiple decryption of the same data without introducing new randomness, it is still an open question if parts of our scheme or some specific properties like its resistance against DPA attacks can be formally proven.

Outline. We first recall the idea and limitations of fresh re-keying in Section 2. In Section 3, we specify the sponge-based authenticated cipher ISAP. We give the design rationales of ISAP in Section 4, and analyze its security in Section 5. Finally, we provide implementation results in Section 6 and conclude in Section 7.

2 Background to Re-keying

While cryptographic implementations can be protected via mechanisms like hiding or masking, frequent re-keying is a countermeasure to DPA that can be seen to work on protocol level. The idea of frequent re-keying is to prevent DPA on the cryptographic primitive by limiting the number of processed inputs per key. In other words, it limits the data complexity for each key by a small number q that renders DPA on the key infeasible (q-limiting [SPY+10]). It is nowadays a common assumption that small data complexities, i.e., $q = 1$ and $q = 2$, have sufficiently small side-channel leakage and do not allow for successful key recovery from DPA attacks [BDH+14, Pie09, SPY+10, TS15].

Frequent re-keying was first proposed for protecting embedded devices such as RFID tags [MSGR10, Koc03]. On the encryption of every new plaintext P, the block cipher E is provided with a new session key K^*. This session key K^* is derived from a pre-shared master secret K and a nonce N that is randomly generated on the tag. This inherently prevents DPA on the session key K^* of the block cipher E. However, for key derivation it requires a re-keying function $g : (K, N) \mapsto K^*$ that is easy to protect against both SPA and DPA attacks.

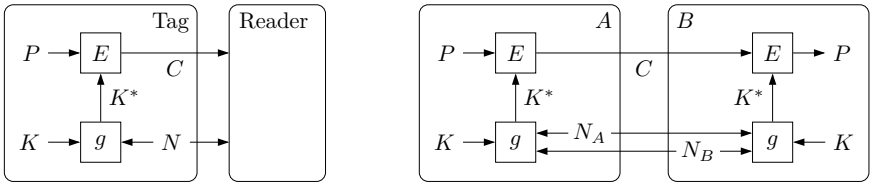
2.1 Secure Re-Keying Function

A secure re-keying function $g : (K, N) \mapsto K^*$ derives a new session key K^* from a master key K and a fresh nonce N and needs to be secure against both SPA and DPA attacks. This security against side-channel attacks can be achieved either on an algorithmic level, or by countermeasures for implementations. Hence, several options for choosing and implementing secure re-keying functions have been proposed.

For instance one option is to build g in such a way that it is easy to secure by classical countermeasures like masking. This is the basic idea of fresh re-keying described in [MSGR10, MPR+11], which uses a polynomial multiplication of K and N to implement g. This multiplication can be masked easily. However, as pointed out in [BFG14, BCF+15, GJ16, PM16], the algebraic structure of a multiplication opens the door to attacks on g and the encryption. Recently, this issue has been addressed by Dziembowski et al. [DFH+16], who propose two new schemes based on learning parity with leakage and learning with rounding.

A second option presented in [SPY+10, FPS12] is based on the classical GGM construction [GGM86]. The GGM construction can be used to mix a secret K with a public N in a tree-like approach, where on each tree level, exactly one bit of the public N is absorbed. Starting with $s_0 = K$, the key s_{i+1} is computed by encrypting one of two predefined plaintexts P_0, P_1 with the key s_i, depending on the i-th bit of N. The output of the last level is then, after postprocessing, used as the session key K^*. In this approach, an attacker only obtains the leakage for two inputs P_0 and P_1 to collect information about each s_i. The construction is thus 2-limiting and is usually considered to be secure against DPA.

Another option presented in [MSJ12, BDH+14] also originates from the classical GGM construction. It follows the idea of [SPY+10] by extending the number of observable measurements per key and deriving a leakage-resilient pseudo-random function (LR-PRF) from common block cipher designs to achieve secure re-keying. The main assumption of this approach is that the attacker is not able to distinguish the leakage of different hardware components on a chip.



(a) Protecting one party.

(b) Protecting both parties.

Figure 1: Re-keying of block ciphers.

2.2 Limitations and Open Problems

One major problem of re-keying schemes such as in Figure 1a is that the reader remains vulnerable to DPA attacks. For instance, such a re-keying scheme can successfully prevent DPA attacks on a device that solely performs encryption or authentication of messages, i.e., the sender of a message, but fails to protect a device performing decryptions or verifications, i.e., the receiver of a message. This is caused by the lack of control of a decryption device on the nonce N and allows attackers to send arbitrary messages to the decryption device using the same nonce N for all sent messages. This malicious procedure results in different messages being decrypted using the same session key K^*. As a result, decryption is vulnerable to DPA, and more concretely, it is the multiple decryption with the same session key K^* that causes this DPA vulnerability. This problem of securing decryption against side-channel attacks was also mentioned by Pereira et al. [PSV15] and Berti et al. [BKP+16].

In order to prevent this kind of DPA attacks, the receiver either needs to be protected by other means [MSGR10], or the receiver needs to be stateful in order to prevent decryption with the same session key twice, or all communication parties are required to contribute to the nonce that is used to derive the session key from a pre-shared master key [MPR+11] as shown in Figure 1b. However, the requirement of both sender and receiver being stateful bears some practical downsides ranging from synchronisation issues between sender and receiver to potential denial-of-service attacks, e.g., if the nonce is a counter and the receiver rejects all messages with a nonce smaller than the last valid nonce which is stored at the receiver. Also the option that all communication parties are required to contribute to the nonce is impractical in several prominent use cases, such as unidirectional/broadcast communication and encrypted storage. Recently, the need for DPA protection in these settings has been pointed out by attacks targeting the decryption of firmware images [ROSW16], or FPGA bitfiles [MBKP11, MS16]. While it is impossible to let a receiver contribute to the nonce in unidirectional communication settings, the additional overhead of letting each receiver contribute to the nonce in a broadcast setting could potentially make an application unpractical. In encrypted storage, the receiving device simply cannot contribute to the nonce, but must be able to decrypt the encrypted data, e.g., an encrypted FPGA bitfile, in all situations and possibly multiple times. To maintain DPA security in this case, one idea would be to re-encrypt the stored data whenever it is read. In practice, however, this is often not possible, e.g., due to the limited number of write operations in flash memory. Moreover, repeated re-encryption can eventually result in a loss of confidentiality [UWM17]. In the next section, we therefore present Isap, a new authenticated encryption scheme that is also secure against DPA attacks in these scenarios.

3 Specification of ISAP

Isap is a family of authenticated ciphers focusing to be secure against passive side-channel attacks. Its functional interface is the same as specified by the CAESAR competition for authenticated encryption [CAE14]: Isap encrypts a plaintext P to a ciphertext C. Additionally, an attached authentication tag T asserts the authenticity of both the plaintext and any optional (unencrypted) associated data A. Each encryption call requires a unique nonce N as an additional input to "randomize" the encryption. Corresponding to the CAESAR call, Isap maintains security no matter how the nonce N is chosen, as long as the same nonce is never used for encryption with the same secret key twice. In this section, we define the Isap authenticated cipher (Subsection 3.1) and its building blocks:

- IsapEnc, a cipher that computes the ciphertext C from the plaintext P and nonce N using the secret key K_E (Subsection 3.3).

- IsapMac, a message authentication code that computes the authentication tag T from the ciphertext C, associated data A, and nonce N using the secret key K_A (Subsection 3.2).

- IsapRk, a function used internally by IsapMac to absorb the secret key K_A (Subsection 3.2).

We propose to implement each building block with variants of the sponge construction using the same permutation size, but different round numbers and rates. We specify several recommended parameter sets in Subsection 3.4.

3.1 Authenticated Encryption Scheme

Isap is a family of sponge-based authenticated encryption schemes $\text{Isap}_{a,b,c}^{r_1,r_2,r_3}$-$k$, where the key size k defines the security level. Isap is an Encrypt-then-MAC design and uses two k-bit keys K_A and K_E ($K = K_A \| K_E$) for IsapMac and IsapEnc, respectively. The length of the tag T and nonce N is also k bits. Each family member is additionally parametrized by several parameters: different round numbers a, b, and c for the permutations and different rates r_1, r_2, and r_3.

The inputs for the authenticated encryption algorithm $\mathcal{E}$ are the secret key $K = K_A \| K_E$, the public nonce N, and associated data A and plaintext P of arbitrary length. Its outputs are the tag T and the ciphertext C with the exact same length as the plaintext P:

$$\mathcal{E}(K, N, A, P) = (C, T).$$

The inputs for the authenticated decryption algorithm $\mathcal{D}$ are the secret key $K = K_A \| K_E$, the public nonce N, the tag T, and associated data A and ciphertext C of arbitrary length. Its outputs are the plaintext P if the verification succeeds, or $\perp$ if the verification fails:

$$\mathcal{D}(K, N, A, C, T) \in \{P, \perp\}.$$

Isap is based on the well-established Encrypt-than-MAC paradigm. Hence, Isap is composed of an encryption algorithm $\text{IsapEnc}_{b,c}^{r_2,r_3}$-$k$ and a message authentication code $\text{IsapMac}_{a,b,c}^{r_1,r_2}$-$k$. The interaction between them is captured in Algorithm 1, where the authenticated encryption $\mathcal{E}$ and authenticated decryption $\mathcal{D}$ are specified.

Algorithm 1: Authenticated encryption and decryption procedures.

Auth. Encryption $\mathcal{E}(K, N, A, P)$	Auth. Decryption $\mathcal{D}(K, N, A, C, T)$
Input: key $K = K_A \| K_E$, $K_A \in \{0,1\}^k$, $K_E \in \{0,1\}^k$, Nonce $N \in \{0,1\}^k$, associated data $A \in \{0,1\}^*$, plaintext $P \in \{0,1\}^*$ **Output:** ciphertext $C \in \{0,1\}^*$, tag $T \in \{0,1\}^k$	**Input:** key $K = K_A \| K_E$, $K_A \in \{0,1\}^k$, $K_E \in \{0,1\}^k$, Nonce $N \in \{0,1\}^k$, associated data $A \in \{0,1\}^*$, ciphertext $C \in \{0,1\}^*$, Tag $T \in \{0,1\}^k$ **Output:** plaintext $P \in \{0,1\}^*$, or $\perp$
Encryption $C \leftarrow \text{IsapEnc}_{b,c}^{r_2,r_3}\text{-}k(K_E, N, P)$ **Authentication** $T \leftarrow \text{IsapMac}_{a,b,c}^{r_1,r_2}\text{-}k(K_A, N, A, C)$ **return** C, T	**Verification** $T' \leftarrow \text{IsapMac}_{a,b,c}^{r_1,r_2}\text{-}k(K_A, N, A, C)$ **if** $T \neq T'$ **return** $\perp$ **Decryption** $P \leftarrow \text{IsapEnc}_{b,c}^{r_2,r_3}\text{-}k(K_E, N, C)$ **return** P

3.2 Authentication Part

For our message authentication code IsapMac, we turn a sponge-based hash function into a suffix-MAC as shown in Figure 2. While the data is absorbed as in a sponge, we use a duplex-like approach to inject the secret key K_A. Here, the k-bit outer part of the state is processed together with the secret key K_A to derive a session key K_A^*, which is then further used as the outer part.

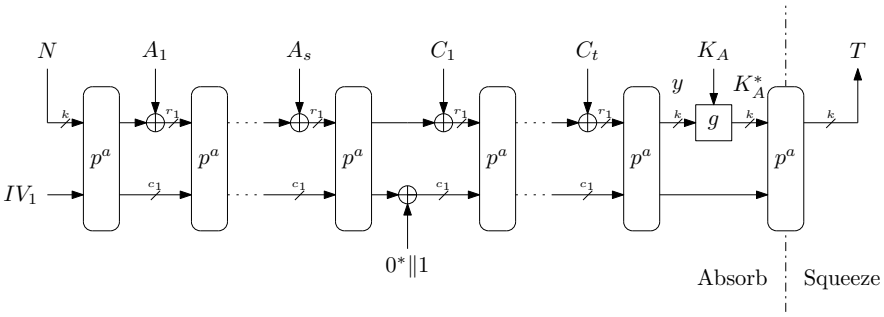


Figure 2: IsapMac used for authentication.

Alternatively, IsapMac can be seen as a sponge-based suffix-MAC, which uses a function g to absorb the secret key K_A instead of an XOR operation. Similar to fresh re-keying schemes [MSGR10], the sole purpose of g is to protect the static master key K_A against various classes of passive side-channel attacks, most prominently differential power analysis. Hence, we will subsequently call g our re-keying function. In our case, we will use IsapRk as re-keying function, which is shown in Figure 3.

IsapMac computes the tag T as follows. Both associated data A and ciphertext C are each padded using a 10^* padding to a length that is a multiple of the rate r_1. The internal state is initialized with the k-bit nonce N and a constant initial value IV_1. Then, s blocks of associated data $A_{1...s}$ and t blocks of ciphertext $C_{1...t}$ are absorbed using the a-round permutation p^a. Similar to Ascon [DEMS14], the XOR of a single bit '1' to the inner part of the state serves as domain separation between associated data and ciphertext. Note that a dedicated domain separation between nonce and associated data is not needed, since the nonce is of a fixed length of k bits. Finally, the key K_A is absorbed via g and the k-bit tag T is squeezed after a final call of the permutation.

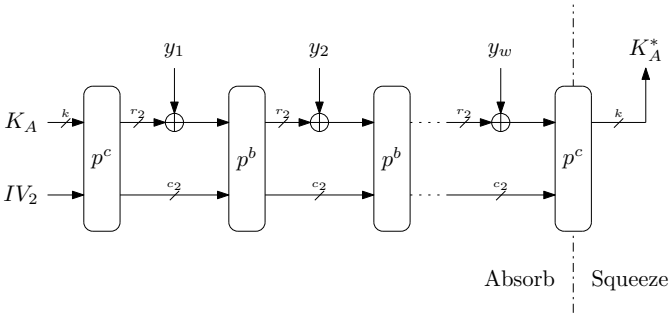


Figure 3: IsapRk used to process the master key K_A.

Instead of a plain XOR, the function $g(K_A, y) = K_A^*$ is used to absorb K_A. To evaluate g, its internal state is initialized with the key K_A and a constant IV_2, followed by an application of the c-round permutation p^c. The k-bit value y is absorbed using a rate size r_2 and the b-round permutation p^b. Finally, the output K_A^* is squeezed using a rate size k and the c-round permutation p^c. The details of IsapMac and IsapRk are also summarized in Algorithm 3 in the appendix. For verification, the tag T' is re-computed in the same way from the received nonce N, associated data A, and ciphertext C.

3.3 Encryption Part

To encrypt the plaintext, we use a sponge-based construction very similar to IsapRk (see Figure 4). We initialize the internal state with the secret key K_E and a constant IV_3, followed by an application of the c-round permutation p^c. The k-bit nonce N is absorbed using a rate of r_2 bits and the b-round permutation p^b. Then, we squeeze a keystream of the same length as the plaintext P using a rate of r_3 bits and the c-round permutation p^c. The ciphertext C is computed as the XOR of the plaintext P and the keystream.

For decryption, the same keystream is computed from the nonce N and XORed to the ciphertext C to obtain the plaintext P. The detailed procedures for encryption and decryption are also given in Algorithm 4 in the appendix.

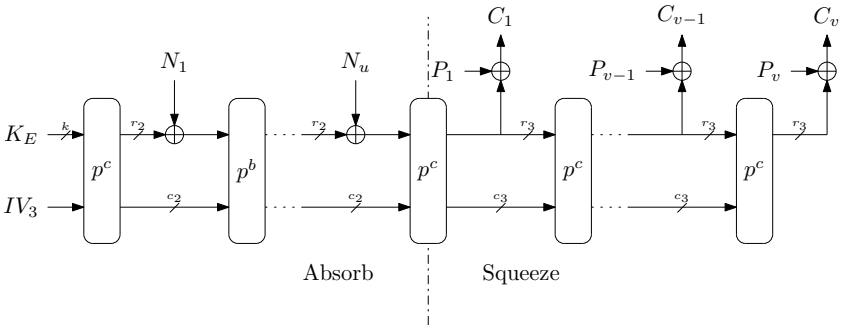


Figure 4: IsapEnc used for encryption.

3.4 Instantiations and Parameter Values

We propose to instantiate the required permutations p^a, p^b, and p^c from the 400-bit permutations Keccak-p[400,n_r], which are the application of the last n_r rounds of Keccak-f[400] [Nat15]. Hence, the only difference between p^a, p^b, and p^c is the different number of rounds a, b, and c that is used. A detailed specification of Keccak-p[400,n_r], including the state layout and specification of the inner and outer state parts, can be found in the submission document of the CAESAR candidate Keyak [BDP+14b].

Table 1 summarizes the recommended parameter sets for Isap. The first, Isap-128, is based on a conservative choice of the relevant parameters based on our design rationale and security analysis given in Section 4 and Section 5. Additionally, we also specify a more aggressive choice of parameters in Isap-128a to encourage further cryptanalysis as well as side-channel analysis. Both algorithms are designed to achieve 128 bits cryptographic security and practical security against side-channel attacks assuming an SPA-secure implementation.

The constant initial values IV_1, IV_2, IV_3, which serve as domain separation between the different algorithms, are specified in Table 2. They are defined as the concatenated bit

Table 1: Recommended parameter configurations for ISAP.

Name	Security level	Bit size of			Rounds		
	k	r_1	r_2	r_3	a	b	c
ISAP-128	128	144	1	144	20	12	12
ISAP-128a	128	144	1	144	16	1	8

values of the used parameter set, plus a different constant for each value, where each entry occupies 1 byte of space. The initial values are then padded with zeros until they reach the length of the permutation minus k bits. In the case of ISAP-128 and ISAP-128a, the IVs have a length of 272 bits, which is more than needed for the desired security level of 128 bits.

Table 2: Initial values for ISAP.

$\text{ISAP}_{a,b,c}^{r_1,r_2,r_3}$-$k$	IV_1	$1\|\|a\|\|b\|\|c\|\|r_1\|\|r_2\|\|r_3\|\|k\|\|0^*$
	IV_2	$2\|\|a\|\|b\|\|c\|\|r_1\|\|r_2\|\|r_3\|\|k\|\|0^*$
	IV_3	$3\|\|a\|\|b\|\|c\|\|r_1\|\|r_2\|\|r_3\|\|k\|\|0^*$
ISAP-128	IV_1	`0x01140c0c90019080`*
	IV_2	`0x02140c0c90019080`*
	IV_3	`0x03140c0c90019080`*
ISAP-128a	IV_1	`0x0110010690019080`*
	IV_2	`0x0210010690019080`*
	IV_3	`0x0310010690019080`*

4 Design Rationale

The main goal of ISAP is to provide security against passive side-channel attacks by design, while still providing good performance and a low hardware footprint. While mechanisms to counteract side-channel attacks and in particular DPA within the cipher itself (e.g., masking) lead to significant overheads and increase with the protection order, approaches based on fresh re-keying lead to much lower overheads. However, state-of-the-art schemes based on re-keying lack security against DPA in scenarios that require multiple decryption of the same input (with the same session key). ISAP is designed to be secure also in such scenarios.

4.1 An Authenticated Encryption Mode Secure Against DPA

For discussing the security of our scheme against differential power analysis (DPA), we prefer to give a more general, high-level view on our mode in Algorithm 2 to better extract the underlying idea. In contrast to the condensed and interwoven descriptions of IsapMac, IsapRk, and IsapEnc, the description in Algorithm 2 clearly shows the fresh re-keying roots of our scheme. Here, we essentially use the same assumptions and requirements as other re-keying schemes. Namely, we assume g_1, g_2 to be (DPA and SPA) secure re-keying functions and assume the implementations of ENC, DEC, and MAC to be secure against SPA attacks when processing arbitrarily long messages. However, there are no requirements on the implementation of the hash function H, since it processes only publicly known data.

To achieve security against DPA, our authenticated encryption mode in Algorithm 2 incorporates the re-keying approach discussed in Section 2 in an efficient Encrypt-then-MAC scheme. While simple re-keying of both a MAC and an encryption scheme can

Algorithm 2: Authenticated encryption and decryption procedures.

Auth. Encryption $\mathcal{E}(K, N, A, P)$	Auth. Decryption $\mathcal{D}(K, N, A, C, T)$
Input: key $K = K_A \| K_E$, $K_A \in \{0,1\}^k$, $K_E \in \{0,1\}^k$, Nonce $N \in \{0,1\}^k$, associated data $A \in \{0,1\}^*$, plaintext $P \in \{0,1\}^*$	**Input:** key $K = K_A \| K_E$, $K_A \in \{0,1\}^k$, $K_E \in \{0,1\}^k$, Nonce $N \in \{0,1\}^k$, associated data $A \in \{0,1\}^*$, ciphertext $C \in \{0,1\}^*$, Tag $T \in \{0,1\}^k$
Output: ciphertext $C \in \{0,1\}^*$, tag $T \in \{0,1\}^k$	**Output:** plaintext $P \in \{0,1\}^*$, or $\perp$
Encryption $K_E^* = g_1(N, K_E)$ $C = ENC_{N,K_E^*}(P)$ **Authentication** $y = H(N, A, C)$ $K_A^* = g_2(y, K_A)$ $T = MAC_{K_A^*}(y)$ **return** C, T	**Verification** $y = H(N, A, C)$ $K_A^* = g_2(y, K_A)$ $T' = MAC_{K_A^*}(y)$ **if** $T \neq T'$ **return** $\perp$ **Decryption** $K_E^* = g_1(N, K_E)$ $P = DEC_{N,K_E^*}(C)$ **return** P

only provide security for the encryption process, our scheme achieves side-channel security for multiple decryption as well. Namely, the verification guarantees the security of the decryption part in case of maliciously modified ciphertexts, while the *MAC* is protected by making its session key depend on the authenticated message itself. In the following, we give a detailed discussion on the DPA security of the two parts encryption/decryption and authentication/verification.

Encryption/Decryption. The encryption and decryption part is an instance of fresh-rekeying such as in [MSGR10, MPR+11]. Such schemes for fresh re-keying combine an SPA-secure encryption scheme *ENC* with a (DPA and SPA) secure re-keying function $g_1 : (K_E, N) \mapsto K_E^*$. As the nonce N that is used to derive the session key K_E^* must not be repeated, fresh session keys are guaranteed and DPA on the encryption scheme *ENC* is effectively prevented.

However, for decryption, there is the threat that an adversary could exploit multiple decryptions with the same session key K_E^* and induce a DPA setting within the decryption *DEC* by using different data, since multiple calls of *DEC* with the same nonce N are allowed. To prevent such a DPA scenario in our mode, verification is performed prior to decryption. Decrypting two different messages (associated data and ciphertext) with the same K_E^* indicates either a collision of g_1 for fixed K_E (depends on concrete instance of g_1, but usually negligible probability), or two ciphertexts that have been encrypted using the same nonce N. Since we require unique nonces, the latter implies that either the ciphertexts are identical, or one ciphertext has been forged. If a cryptographically secure MAC is used, the probability of a successful forgery is negligible and thus the tag verification will fail for one of the ciphertexts with overwhelming probability.

Authenticated ciphers require that no decrypted plaintext is released if tag verification fails. To ensure protection against DPA attacks, we go one step further and require a failed verification to abort the authenticated decryption process, so that the decryption part *DEC* never starts. This ensures that the same session key K_E^* is never used to decrypt distinct ciphertexts with *DEC*. Therefore, the verification is responsible for precluding DPA attacks on the decryption.

Authentication/Verification. The authentication/verification shown in Algorithm 2 is based on a hash-then-MAC paradigm. Here, a session key K_A^* is first derived via a

secure re-keying function g_2 from the hash value y that is computed from the nonce N, associated data A, and ciphertext C using a cryptographic hash function H. Then, a message authentication code (MAC) is used to compute the tag T from the hash value y and the session key K_A^*. This is similar to the construction of Pereira et al. [PSV15], who designed a leakage-resilient MAC based on the hash-then-MAC paradigm as well. However, the main difference to our approach is that in [PSV15], a random nonce N is used to derive the session key in the re-keying function. This, however, cannot provide protection against DPA for multiple verifications. Contrary to that, we use the hash of the message $y = H(N, A, C)$ to derive the session key K_A^* in order to securely allow multiple verifications while still providing protection against DPA.

In more detail, the MAC in Algorithm 2 computes the tag T using a different session key K_A^* for every distinct message (N, A, and C), because distinct messages result in distinct hash values in the absence of collisions. Hence, DPA on the MAC is prevented during the generation of the tag T as the same session key K_A^* is never used to authenticate distinct messages.

While the scheme by Pereira et al. [PSV15] also provides side-channel security during tag generation by the use of a unique nonce input N to the re-keying function, tag verification imposes different challenges. In fact, during tag verification one cannot rely on the uniqueness of the nonce anymore, because an attacker can usually modify the message (N, A, and C) to provoke multiple verifications with different data under the same nonce N and thus allowing for a DPA scenario. However, the MAC in Algorithm 2 prevents such a DPA scenario on the session key K_A^*, since K_A^* is bound to the data it processes. Namely, as y depends on the message (N, A, and C), the MAC session key $K_A^* = g(y, K_A)$ changes whenever the data changes. Adversaries cannot predictably influence y due to the use of a cryptographic hash function H. This guarantees that the key K_A^* is unique for every new message as long as there is neither a collision in the hash function H nor in the re-keying function g_2. Thus, DPA on the session key K_A^* is effectively prevented during verification.

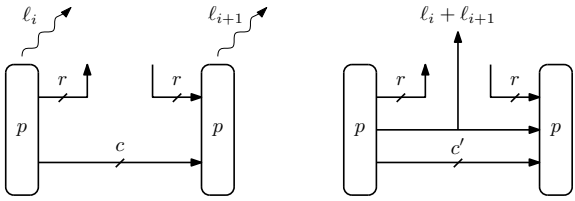
Note however that collisions in the re-keying function g_2 or the hash function H may result in the same session key K_A^* being used in MAC computations of different messages, thus allowing for a DPA. Yet, collisions in g_2 depend on the secret key K_A^* and therefore inputs causing collisions in g_2 cannot be calculated off-line. In contrast, collisions in the hash value y are directly observable and can be calculated off-line. The complexity of calculating collisions off-line is determined by the size of the hash. The generic complexity of finding a collision for an m-bit hash function is $2^{m/2}$. Hence, the size of the hash needs to be chosen depending on the potential threat of such an event, which depends on the concrete choice of functions for MAC and g_2.

4.2 Sponges and Side-Channels Leakage

While the mode of Subsection 4.1 ensures protection of the encryption, ENC, decryption DEC, and message authentication code MAC against DPA, the primitives implementing ENC, DEC, and MAC still have to withstand SPA attacks. Moreover, SPA protection is also mandatory for the implementations of g_1 and g_2, in addition to the requirement that they provide protection against DPA. Besides dedicated countermeasures like, e.g., shuffling, the order of the executed instructions, and already the choice of the used algorithms for encryption/decryption and MAC, play an important role for the resistance of the design against SPA.

Our choice for sponge-based designs is motivated by their suitability to model SPA leakage. Namely, the sponge parameters provide a convenient tool to argue on the side-channel security of keyed sponge constructions given bounded side-channel leakage of the single permutation.

For illustration, we model the leakage from a permutation p by allowing an adversary to learn a certain amount of the state between subsequent permutation calls as depicted in



(a) Leaking permutations (b) Sponge model

Figure 5: Leakage of information in sponge-based constructions.

Figure 5. Hereby, we use ℓ to denote the amount of information (in bits) that an attacker can learn about the state from the collected side-channel information. We do not care how and where the leakage is created within p, but let the adversary account the learned information to either the input or the output state of p. Therefore, given two consecutive permutations p with leakages ℓ_i and ℓ_{i+1}, respectively, the maximum an adversary might learn about the state is $\ell_i + \ell_{i+1}$. This means that if each leakage ℓ_i, ℓ_{i+1} is bounded by λ bits and the adversary can optimally combine these two leakages, the adversary will learn at most 2λ bits of the state between the respective two permutation calls.

The basic idea now is to use the sponge parameters to express a construction's capability to cope with the leakage generated by the permutation. In particular, the sponge parameters are adjusted according to the amount of information an adversary learned about the secret state. This means that if the adversary learns 2λ bits of the internal, secret state, the leaked bits can be considered as an increase of the rate, i.e., $r' = r + 2\lambda$, which results in a smaller capacity $c' = c - 2\lambda$ and thus reduced security. However, a reduced security level corresponding to a capacity of $c - 2\lambda$ bits is still guaranteed by the cryptographic properties of the permutation and the associated constrained-input constrained-output (CICO) problem [BDPV11a]. Sponge-based constructions can thus be considered to have bounded security loss for bounded leakage of the permutation.

Clearly, the challenge in practice is to build an implementation that bounds the leakage of p. Especially if many different types of devices have to use the same cryptographic algorithm it might be infeasible to make any realistic assumptions about the leakage of p. Nevertheless, the advantage of the sponge-based construction is that besides standard SPA countermeasures, like hiding and masking, the capacity is an additional and very natural security parameter that helps to increase the ability of a design to withstand side-channel attacks in practice.

While the above modelling and arguing about the leakage is quite useful, it points out a problem with the absorption of the key. If a key is directly absorbed, the upcoming permutation call directly leaks information about the key bits via side-channels. This has a direct effect on the security of the scheme if the used key length matches the security level. Hence, we propose to store the expanded key, after the application of p^c, for IsapRk and IsapEnc.

Besides giving a useful tool to model and argue about the SPA resistance, sponge-based constructions provide other significant advantages:

- The sponge construction is well-studied and has been analyzed and proven secure for different applications in a large amount of publications [JLM14, ADMV15, BDPV11b].
- It allows to implement a wide range of primitives (hash, MAC, cipher).
- Elegant and simple design, obvious state size, no key schedule, key is injected once.
- Little implementation overhead for decryption, since no inverse building blocks (permutation) are needed.

4.3 Design of IsapMac

To get more insight into the design rationals behind IsapMac, we first take a look at a direct instantiation of the authentication/verification described in Algorithm 2. Figure 6 sketches such an instantiation using a sponge-based hash function and suffix MAC. In contrast to the description in Algorithm 2, the MAC is computed directly using the data instead of the hash value. This leads to a construction where the data is processed twice.

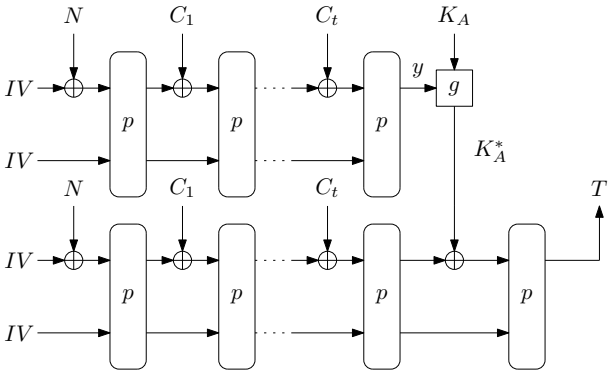


Figure 6: Sketch of authentication/verification using sponge-based hash and suffix MAC.

However, it is possible to omit the hash function and process the key in a manner that resembles the duplex construction [BDPV11b]. As shown in Figure 7, the outer part of the state is used to derive a session key K_A^*, which is then absorbed. This principle is further tweaked (e.g., by implicitly assuming that the employed re-keying function is $g(K_A, y) \oplus y$ to eliminate the XOR used to absorb K_A^*) which leads to IsapMac as presented in Subsection 3.2 (Figure 2). An alternative way of interpreting IsapMac is to see it as sponge-based suffix-MAC that uses a secure re-keying function g to absorb the secret key K_A instead of an XOR. Due to the simplicity of this description, we have chosen to stick to it throughout the paper.

Bertoni et al. [BDPV11a] showed that one can always turn a sponge into a MAC by either putting the key before (prefix-MAC) or after the message (suffix-MAC), as this always gives a pseudo-random function as long as the sponge itself behaves like a random oracle. Compared to a "standard" sponge-based suffix-MAC, IsapMac uses a secure re-keying function g to absorb the secret key K_A. While there are several options for g, e.g., the polynomial multiplication in [MSGR10], we use the function IsapRk as g.

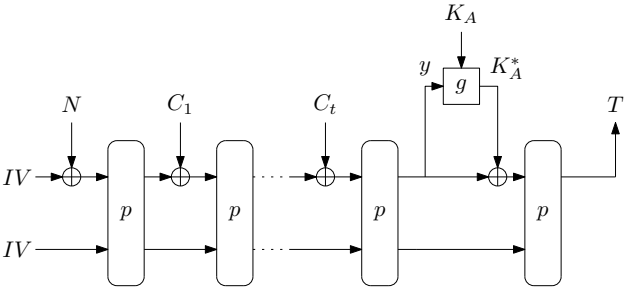


Figure 7: Sketch of authentication/verification just using a sponge-based suffix MAC.

Table 3: Complexity for receiving a v-collision for a 128-bit session key k_2.

v	2	3	4	5	$\cdots$	34
complexity	$2^{64.5}$	$2^{86.2}$	$2^{97.1}$	$2^{103.8}$	$\ldots$	2^{128}

Although IsapRk is not a permutation for a fixed key as, e.g., a polynomial multiplication, we do not expect any negative consequences on the security when absorbing the secret key via a function that ideally behaves like a pseudo-random function.

Instead of using a distinct padding or frame bits for domain separation between associated data and ciphertext, we follow the approach of Ascon [DEMS14] and XOR a single '1' to the inner part of the state. Although this reduces the capacity by one bit in the worst case, the practical security loss is considered to be negligible.

IsapMac prevents DPA on the tag computation in two ways. First, and as shown in Figure 2, the MAC session key K_A^* is derived from the hash value y and the MAC master key K_A via a secure re-keying function g, thus prohibiting DPA on K_A. Second, the design prevents DPA on the MAC session key K_A^* by binding it to the data being processed, thus leading to different MAC session keys K_A^* for different data.

As already mentioned before, a collision in y allows for two side-channel measurements of the MAC using different data but the same MAC session key K_A^*. This holds true for IsapMac as well. Yet, to perform a successful DPA, usually more than two traces will be needed to recover one fixed session key K_A^*. Such a setting occurs with hash multi-collisions. The generic complexity for finding a v-collision is $\sqrt[v]{v! \cdot 2^{m(v-1)}}$ [STKT06]. Luckily, the complexity is quite high already for small values of v as shown in Table 3 for a 128-bit value y.

However, we want to stress that even though a DPA attack exploiting multi-collisions might be able to recover the session key K_A^* of IsapMac, this does not imply a key recovery attack on the master key K_A, since our used re-keying function g (IsapRk) is hard to invert.

4.4 Design of IsapRk

The re-keying function IsapRk used in IsapMac is a sponge-based design as depicted in Figure 3. When setting the rate r_2 to 1, the design is related to the classical GGM construction [GGM86] and can be seen as their sponge-based equivalent, similar to [TS14]. The basic idea in IsapRk is to make DPA infeasible by reducing the input data complexity accordingly. For this purpose, a secret state is constantly updated with small portions of of public data by repeating two phases, (1) modifying the secret state according to the public data, and (2) updating the state such that predictions on the future state based on the absorbed public data become infeasible.

Sponge-based constructions are an ideal choice to implement this basic idea as the rate directly influences the input data complexity for each permutation. IsapRk follows this approach and first initializes the internal state by applying the initial permutation p^c to the padded master key K_A. Then, IsapRk repeatedly injects r_2 nonce bits into the state, each separated by a permutation call p^b. After full absorption of the nonce and finalization using p^c, the session key K_A^* is output. This working principle is similar to sponge instances of a prefix-MAC. While for general MAC computations the absorption rate can be as big as the state size [BDPV12], IsapRk uses a small absorption rate $r_2 = 1$ to limit the data complexity exploitable in a DPA.

In terms of DPA security, a permutation p^b will produce the leakages for two different public inputs, thus IsapRk is 2-limiting per permutation call. This results in IsapRk being a secure re-keying function (regarding DPA) under the assumption that the combined leakage resulting from the processing of two different public inputs is bounded such that

DPA on the secret state is infeasible. This is a common assumption also used in recent block-cipher based instantiations of the GGM construction by Faust et al. [FPS12] or the 2PRG primitive by Standaert et al. [SPY+10]. The reason for using a different permutation p^c at the beginning of IsapRk lies in the fact that some of the concrete instances of IsapRk use a small number of rounds for p^b compared to p^c and we want to ensure good diffusion of the key bits across the whole state before the first non-secret bits are absorbed.

4.5 Design of IsapEnc

The encryption algorithm IsapEnc is an instance of fresh re-keying [MSGR10, MPR+11] that combines the secure re-keying function IsapRk in the initialization phase with a sponge-based stream cipher in the processing phase. However, for the analysis it is more natural to see it as an extension of IsapRk with a longer squeezing phase to produce a keystream of arbitrary length.

As the initialization part is equivalent to IsapRk, it is secure against passive side-channel attacks in consideration of the same aspects, i.e., a small rate $r_2 = 1$ to inject the nonce N with low data complexity. To obtain cryptographic security on the processing part of IsapEnc, the nonce N must not be repeated for different plaintexts. This guarantees that the key stream is unpredictable and unique for different encryptions. As a consequence, DPA on the encryption itself is prevented as well. Moreover, as a part of the authenticated encryption scheme Isap, IsapEnc remains secure against DPA also for multiple decryption of the same data, since it is guaranteed that this data is always decrypted under the same nonce. As mentioned before, current schemes lack this functionality and become vulnerable to DPA if an attacker tampers with the ciphertext or nonce. In IsapEnc, such attack becomes infeasible by using the generic composition Encrypt-then-MAC, i.e., performing verification prior to decryption. Namely, the authentication part aborts the process if tag verification fails, which ensures that the same key is never used to decrypt distinct ciphertexts. Hence, the authentication part precludes DPA attacks on the decryption part.

4.6 Choice of the Permutation

In the case of sponge-based constructions, minimal suitable bit-sizes for permutations are tightly coupled with the aimed security level. Both instances Isap-128 and Isap-128a target 128-bit security. Hence, the capacity of IsapMac should be at least 256 bits, since it is a sponge-based suffix MAC and thus, we have to rely on the results of Bertoni et al. [BDPV08]. If we want to output the tag with one permutation call, while still retaining 256 bits for the capacity, this implies a minimal permutation size of 384 bits. Since Isap-128 and Isap-128a are also aimed for lightweight and low-cost applications, while high performance applications are not the main target, we do not want to increase the rate much and hence want to stay close to 384 bits. However, there is a lack of well-analyzed 384-bit permutations. Thus, we opted to use the well established and analyzed Keccak-p[400,n_r] permutations [Nat15].

Parameters for IsapMac. Since we aim for 128-bit security, we use IsapMac for both instances with a capacity c_1 of 256 bits, while allowing the remaining 144 bits as rate r_1. For the conservative choice Isap-128, we choose p^a to be the permutation Keccak-f[400] (Keccak-p[400,20]) that has 20 rounds as specified in the the Keccak SHA-3 submission (Version 3.0) [BDPV11c]. Since Keccak is the winner of the SHA-3 competition, its variants have been well analyzed. However, current attacks are far away from threatening full-round versions of Keccak. Therefore, we use for our aggressive variant Isap-128a the initial Keccak-p[400,16] with 16 rounds as proposed in the Keccak sponge function family main document (Version 1.2) [BDPV09].

Parameters for IsapRk and IsapEnc. Both IsapRk and IsapEnc are keyed sponge-based constructions with clearly separated absorbing and squeezing phases. According to recent results [BDPV12, GPT15, MRV15], we could set the capacity during the absorbing phase to $c_2 = 0$ and the capacity during the squeezing phase to a minimum of $c_3 = 128$ bits. However, we also have to bear side-channel attacks in mind. Hence, we set the rate to $r_1 = 1$, making the scheme essentially 2-limiting per permutation call p^b, while setting the rate $r_3 = 144$ bits to match the block size of IsapMac. In terms of our arguments of Subsection 4.2, this means that an attacker has to learn about 136 bits of information during invocations of p^b and about 64 bits of information via side-channels during the invocation of p^c, before the attacker is able to invert the sponge with a complexity less than 2^{128} to recover the secret key.

For the number of rounds for Isap-128, the CAESAR candidate Keyak serves as orientation. Hence, we use Keccak-p[400,12] for p^b and p^c. As for Keyak, we expect 12 rounds to be enough to create an unpredictable key-stream during the squeezing phase. Moreover, 12 rounds provide a clear separation between the single-bit injections during the absorption, so that partially known/leaked information about the internal secret state cannot be combined over one permutation call.

The CAESAR candidate Ketje serves as inspiration for the aggressive version Isap-128a. Similar to Ketje, only one round separates the absorption of the one bit elements using Keccak-p[400,1] for p^b. Note that here the side-channel leakage between single permutation calls can clearly be combined. For the squeezing phase, we orient the number of rounds on the "stride" permutation call of Ketje Sr, which has 6 rounds. However, in contrast to Ketje Sr, we have a higher rate of 144 bits during the squeezing phase. Hence, we have decided to add an additional security margin of 2 rounds and use the 8 round permutation Keccak-p[400,8] for p^c.

5 Security Analysis

Due to the prominence of Keccak [BDPV11c] as winner of the SHA-3 competition [Nat12], and Keyak [BDP+14b] and Ketje [BDP+14a] as submissions to CAESAR [CAE14], a plethora of cryptanalytic results for keyed and unkeyed sponge and duplex constructions using round reduced versions of the Keccak-f permutations, as well as on the permutations exist. While arguably the majority of the analyses focuses on the 1600-bit variant of the Keccak-f permutation, the similarity in structure of the permutation usually allows to apply the same techniques on smaller permutation variants. A good overview on existing analysis results on Keccak can be found in [JN15]. In this section, we recapitulate the from our point of view most relevant attacks on Keccak and discuss the applicability to our schemes. Finally, we conclude this section with a note on the side-channel security of Isap.

5.1 Permutation

Zero-sum distinguishers [AM09, BC10] are the permutation distinguishers penetrating the highest number of rounds. They exploit the low algebraic degree of the Keccak-f permutations creating sets of inputs and outputs, which sum to zero. Guo et al. [GLS16] present zero-sum distinguishers for 12 rounds of Keccak-f[1600] with a complexity of 2^{65} using a 3-round linear structure in the middle of the permutation, while achieving 2^{82} using a 2-round linear structure. They also claim for the 12-round 400-bit permutation Keccak-p[400,12] zero-sum distinguishers with a complexity 2^{82} using a 2-round linear structure, while 3-round structures seem to be inapplicable. However, to mount an attack using zero-sum distinguishers on sponges, an attacker would have to be able to choose inputs in the middle of the permutation. Thus, no attacks on Keyak and Ketje with

the 12-round Keccak-p permutations are known that exploit zero-sum distinguishers. Therefore, we conclude that the same is true for Isap-128, which also uses 12 rounds for IsapEnc and IsapRk.

5.2 IsapRk and IsapEnc

IsapRk and IsapEnc are sponge-based constructions where the secret key is injected during the beginning of the absorption phase, similar to a Keccak prefix-MAC, Keyak, or Ketje. We refer to such constructions as keyed sponges. The attacks penetrating the highest number of rounds for keyed sponges exploit the low algebraic degree of the Keccak-f permutations. This includes the cube-like attacks by Dinur et al. [DMP+15], who present amongst others a keystream prediction for a Keccak-based stream cipher which uses 9 rounds of the 1600-bit permutation to achieve 512-bit security with time complexity 2^{256}. Huang et al. [HWX+17] present conditional cube attacks, including a key-recovery attack on 8 rounds of Keyak with a time complexity of 2^{74}.

In the case of Isap-128, two factors prohibit those attacks. First of all, the permutation has 12 rounds, whereas the attacks are only capable of penetrating at most 9 rounds. Second, the nonce N or the hash value y are absorbed bitwise separated by 12 rounds of the permutation, which significantly reduces the ability of an attacker to exploit cubes in the first place. For Isap-128a, the number of rounds between the bitwise injections of the nonce N or the hash value y is reduced to one. Still, this means having at least 128 rounds from the point where the key is introduced up to the point when a part of the state is leaked. Hence, we expect that conditional cube and cube-like attacks do not work on Isap-128a.

Another important attack vector are linear and differential attacks. These are especially relevant in the case of Isap-128a, where only the 1-round permutation is used for absorption and the 8-round permutation is used for squeezing the sponge. While having, e.g., colliding differential trails during absorption would also imply problems for Ketje, the situation changes for the squeezing phase. Due to the increased rate used in Isap-128a compared to Ketje, an attacker has more freedom. For this reason, we have increased the number of rounds to 8 for p^c.

5.3 IsapMac

Since IsapMac is a suffix-MAC, attacks when unkeyed sponges are used as hash functions are also of concern. For instance, collision attacks on the hashing part of IsapMac have the potential to allow for forgeries. For Keccak, collision attacks for up to 5 rounds were proposed by Dinur et al. [DDS13]. Recently, the 5-round challenges for 1600-bit and 800-bit permutations of the Keccak crunchy crypto collision contest [BDPV14] have been solved, while the 5-round challenge for the 400-bit permutation is still open. Regarding pre-image attacks, attacks for up to 4 rounds for variants of Keccak exist [MPS13, GLS16]. Taking these results together with the result for keyed sponges of Subsection 5.2, we conclude that having 20 rounds in the case of Isap-128 and even 16 rounds in the case of Isap-128a provide a sufficient security margin for IsapMac.

5.4 On the Side-Channel Security of Isap

While Isap has been designed to be secure against DPA attacks, care has do be taken regarding SPA attacks. Although the single components IsapMac, IsapRk and IsapEnc of Isap have been designed keeping their resistance against SPA attacks in mind, additional countermeasures on implementation level for all components might be needed. In particular for the decryption, where several measurements for the same data are possible, dedicated countermeasures against SPA attacks are crucial.

As already pointed out by Medwed et al. [MSJ12], the concrete security of a construction against side-channel attacks highly depends on the way it is implemented and on the platform on which it is executed. For instance, they show that an implementation of the GGM construction using AES-128 on an 8-bit microcontroller can be broken by using template attacks. By making assumptions on the implementation, e.g., parallel execution of the S-boxes, Medwed et al. [MSJ12] and follow-up work [MSNF16] are able to provide security guarantees with respect to side-channel attacks for their constructions. In contrast, in this work we do not make any assumption on the way Isap is implemented and on the countermeasures used to protect the implementations. Clearly, an 8-bit microcontroller implementation needs more sophisticated SPA countermeasures than a parallel implementation of the round function. We consider the evaluation of the SPA resistance of various implementation strategies for Isap to be an interesting topic for further research.

6 Implementation

We implemented our authenticated encryption scheme Isap in the two configurations Isap-128 and Isap-128a as presented in Table 1. The actual implementation of both configurations is the same except for the number of rounds. The implementations employ a single instance of the 400-bit Keccak permutation that performs one round per cycle. The number of rounds performed is chosen at runtime depending on the executed algorithm, i.e., IsapEnc, IsapMac, or IsapRk. The synthesis results using a 130 nm UMC technology are shown in Table 4. The choice of 130 nm UMC technology is motivated by the tools which are available to us.

Table 4: Implementation of the AE modes (130 nm).

Function	Area [kGE]	Frequency [MHz]	Initialization [cycles]	Initialization [μs]	Runtime per Block [cycles]	Runtime per Block [μs]
Isap-128	14.0	169	3 401	20.1	36	0.20
Isap-128a	14.0	169	564	3.3	28	0.16

Area. As Isap-128 and Isap-128a use the same implementation design, they each consume 14.0 kGE of chip area. Most of the chip area is due to the Keccak core, which consumes 8.3 kGE. The remaining logic is required for multiplexing and a temporary state register to hold the hash value within IsapMac when performing the secure re-keying function IsapRk. A sole implementation of the secure re-keying function IsapRk yields roughly the same size as the Keccak core itself and is thus slightly smaller than other re-keying functions like a masked polynomial multiplication [MSGR10] or an implementation of the GGM tree using an AES core computing 1 round per cycle [SPY+10].

Runtime. The measured runtime is broken down into two parts: the time for performing initialization, and the time for encrypting and authenticating a 144-bit message block. The runtime of performing initialization is dominated by performing the re-keying operations in both IsapEnc and IsapMac and is independent of the length of the message. Its impact on runtime thus vanishes for long messages. The runtime for processing a single 144-bit block is also independent of the length of the message, but strongly influences the overall runtime for long messages.

Compared to the conservative parameterization Isap-128, the more aggressive parameters of Isap-128a yield a speed-up of 83 % for initialization and 22 % for the processing of a message block. While the very high speed-up during initialization is highly beneficial

for short messages, the speed-up observed for encryption and authentication of a 144-bit message block dominates for long messages.

Comparison. Isap is an efficient authenticated encryption scheme with low hardware footprint that prevents DPA by design. While Isap is based on a standard implementation of the 400-bit Keccak permutation and thus only adds a little hardware overhead, a first-order secure threshold implementation increases the area by a factor of 3–4 [BDN+13]. Similar for AES the area for first-order secure masked implementations [DRB+16, GMK17] increases accordingly. When higher-order DPA security is required, the hardware overhead of masking rises even more [GMK17]. Consequently, the implementation cost of standard authenticated encryption modes for AES such as AES-CCM and AES-GCM secured via masking rises accordingly.

7 Conclusion and Open Questions

While current authenticated encryption schemes such as the CAESAR candidates Ascon, Ketje/Keyak, PRIMATES, and SCREAM are designed to reduce the overhead of side-channel countermeasures like masking on an implementation level, we explored in this work how side-channel attacks can be tackled on an algorithmic level, while still fulfilling the functional requirements of the CAESAR call. Probably the most notable resulting restriction of this is that it is not possible to make any assumptions on the choice of the nonce, besides the fact that the nonce has to be unique per encryption (e.g. it must be possible to implement the nonce as simple counter on encryption side). Hence, the decrypting/verification unit has no influence on the choice of the nonce and thus has to allow multiple decryptions/verifications of (different) ciphertexts with the same nonce.

As a result, we proposed Isap, an authenticated encryption scheme that incorporates ideas from fresh re-keying to withstand DPA attacks. In contrast to existing fresh re-keying schemes, Isap protects the decryption/verification unit against DPA attacks, although the decryption/verification unit does not contribute to the nonce that is used for encryption. This feature does not only reduce communication overhead, but it enables several use cases that are not feasible with current re-keying schemes such as simply storing encrypted data and decrypting it later multiple times. The results of our hardware implementation show that the concrete instances Isap-128 and Isap-128a can be implemented in a straightforward manner with an area of only 14 kGE, while offering security against DPA attacks even for multiple decryption. Therefore, we think that Isap is a valuable addition to the existing pool of symmetric authenticated encryptions schemes and hope that its novel underlying ideas and concepts will stimulate discussion and trigger future work in this direction.

Acknowledgments

The authors would like to thank Mario Werner for many helpful discussions and providing his hardware description of Keccak.

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644052 (HECTOR) and agreement No 681402 (SOPHIA).

Furthermore, this work has been supported in part by the Austrian Research Promotion Agency (FFG) under grant number 845589 (SCALAS) and by the Austrian Science Fund (project P26494-N15).

References

[ABB+14] Elena Andreeva, Begül Bilgin, Andrey Bogdanov, Atul Luykx, Florian Mendel, Bart Mennink, Nicky Mouha, Qingju Wang, and Kan Yasuda. PRIMATEs. Submission to the CAESAR competition: http://competitions.cr.yp.to, 2014.

[ADMV15] Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche. Security of keyed sponge constructions using a modular proof approach. In Gregor Leander, editor, *FSE 2015*, volume 9054 of *LNCS*, pages 364–384. Springer, 2015.

[AM09] Jean-Philippe Aumasson and Willi Meier. Zero-sum distinguishers for reduced Keccak-f and for the core functions of Luffa and Hamsi. https://131002.net/data/papers/AM09.pdf, 2009.

[BC10] Christina Boura and Anne Canteaut. A zero-sum property for the KECCAK-f permutation with 18 rounds. In *ISIT 2010*, pages 2488–2492. IEEE, 2010.

[BCF+15] Sonia Belaïd, Jean-Sébastien Coron, Pierre-Alain Fouque, Benoît Gérard, Jean-Gabriel Kammerer, and Emmanuel Prouff. Improved side-channel analysis of finite-field multiplication. In Tim Güneysu and Helena Handschuh, editors, *CHES 2015*, volume 9293 of *LNCS*, pages 395–415. Springer, 2015.

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

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

[BDN+13] Begül Bilgin, Joan Daemen, Ventzislav Nikov, Svetla Nikova, Vincent Rijmen, and Gilles Van Assche. Efficient and first-order DPA resistant implementations of Keccak. In Aurélien Francillon and Pankaj Rohatgi, editors, *CARDIS 2013*, volume 8419 of *LNCS*, pages 187–199. Springer, 2013.

[BDP+14a] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Ketje. Submission to the CAESAR competition: http://competitions.cr.yp.to, 2014.

[BDP+14b] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Keyak. Submission to the CAESAR competition: http://competitions.cr.yp.to, 2014.

[BDPV08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197. Springer, 2008.

[BDPV09] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Keccak sponge function family main document (Version 1.2). http://keccak.noekeon.org/Keccak-main-1.2.pdf, 2009.

[BDPV11a] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Cryptographic sponge functions (Version 0.1). http://sponge.noekeon.org/, 2011.

[BDPV11b] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the sponge: Single-pass authenticated encryption and other applications. In Ali Miri and Serge Vaudenay, editors, *SAC 2011*, volume 7118 of *LNCS*, pages 320–337. Springer, 2011.

[BDPV11c] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. The Keccak SHA-3 submission (Version 3.0). http://keccak.noekeon.org/Keccak-submission-3.pdf, 2011.

[BDPV12] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Permutation-based Encryption, Authentication and Authenticated Encryption. DIAC Workshop Record (http://www.hyperelliptic.org/djb/diac/record.pdf), 2012.

[BDPV14] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak crunchy crypto collision and pre-image contest. http://keccak.noekeon.org/crunchy_contest.html, 2014.

[BFG14] Sonia Belaïd, Pierre-Alain Fouque, and Benoît Gérard. Side-channel analysis of multiplications in GF(2^{128}) – Application to AES-GCM. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014*, volume 8874 of *LNCS*, pages 306–325. Springer, 2014.

[BKP+16] Francesco Berti, François Koeune, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Leakage-resilient and misuse-resistant authenticated encryption. Cryptology ePrint Archive, Report 2016/996, 2016. http://eprint.iacr.org/2016/996.

[CAE14] CAESAR committee. CAESAR: Competition for authenticated encryption: Security, applicability, and robustness. http://competitions.cr.yp.to/, 2014.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES 2002*, volume 2523 of *LNCS*, pages 13–28. Springer, 2002.

[DDS13] Itai Dinur, Orr Dunkelman, and Adi Shamir. Collision attacks on up to 5 rounds of SHA-3 using generalized internal differentials. In Shiho Moriai, editor, *FSE 2013*, volume 8424 of *LNCS*, pages 219–240. Springer, 2013.

[DEMS14] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon. Submission to the CAESAR competition: http://competitions.cr.yp.to, 2014.

[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 Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016*, volume 9815 of *LNCS*, pages 272–301. Springer, 2016.

[DKM+15] Christoph Dobraunig, François Koeune, Stefan Mangard, Florian Mendel, and François-Xavier Standaert. Towards fresh and hybrid re-keying schemes with beyond birthday security. In Naofumi Homma and Marcel Medwed, editors, *CARDIS 2015*, volume 9514 of *LNCS*, pages 225–241. Springer, 2015.

[DMP+15] Itai Dinur, Pawel Morawiecki, Josef Pieprzyk, Marian Srebrny, and Michal Straus. Cube attacks and cube-attack-like cryptanalysis on the round-reduced Keccak sponge function. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 733–761. Springer, 2015.

[DRB+16] Thomas De Cnudde, Oscar Reparaz, Begül Bilgin, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Masking AES with $d + 1$ shares in hardware. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *CHES 2016*, volume 9813 of *LNCS*, pages 194–212. Springer, 2016.

[FPS12] Sebastian Faust, Krzysztof Pietrzak, and Joachim Schipper. Practical leakage-resilient symmetric cryptography. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 213–232. Springer, 2012.

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

[GJ16] Qian Guo and Thomas Johansson. A new birthday-type algorithm for attacking the fresh re-keying countermeasure. Cryptology ePrint Archive, Report 2016/225, 2016. http://eprint.iacr.org/2016/225.

[GLS+14] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, Kerem Varici, François Durvaux, Lubos Gaspar, and Stéphanie Kerckhoff. SCREAM. Submission to the CAESAR competition: http://competitions.cr.yp.to, 2014.

[GLS16] Jian Guo, Meicheng Liu, and Ling Song. Linear structures: Applications to cryptanalysis of round-reduced Keccak. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016*, volume 10031 of *LNCS*, pages 249–274, 2016.

[GMK17] Hannes Gross, Stefan Mangard, and Thomas Korak. An efficient side-channel protected aes implementation with arbitrary protection order. In Helena Handschuh, editor, *CT-RSA 2017*, volume 10159 of *LNCS*, pages 95–112. Springer, 2017.

[GPT15] Peter Gazi, Krzysztof Pietrzak, and Stefano Tessaro. The exact PRF security of truncation: Tight bounds for keyed sponges and truncated CBC. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015*, volume 9215 of *LNCS*, pages 368–387. Springer, 2015.

[HWX+17] Senyang Huang, Xiaoyun Wang, Guangwu Xu, Meiqin Wang, and Jingyuan Zhao. Conditional cube attack on reduced-round Keccak sponge function. In *EUROCRYPT 2017*, 2017. (to appear).

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 463–481. Springer, 2003.

[JLM14] Philipp Jovanovic, Atul Luykx, and Bart Mennink. Beyond $2^{c/2}$ security in sponge-based authenticated encryption modes. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014*, volume 8873 of *LNCS*, pages 85–104. Springer, 2014.

[JN15] Jérémy Jean and Ivica Nikolic. Internal differential boomerangs: Practical analysis of the round-reduced Keccak-f permutation. In Gregor Leander, editor, *FSE 2015*, volume 9054 of *LNCS*, pages 537–556. Springer, 2015.

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

[Koc03] Paul Kocher. Leak Resistant Cryptographic Indexed Key Update, US Patent 6539092, 2003.

[MBKP11] Amir Moradi, Alessandro Barenghi, Timo Kasper, and Christof Paar. On the vulnerability of FPGA bitstream encryption against power analysis attacks: extracting keys from Xilinx Virtex-II FPGAs. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *CCS 2011*, pages 111–124. ACM, 2011.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks – Revealing the secrets of smart cards.* Springer, 2007.

[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 Emmanuel Prouff, editor, *CARDIS 2011*, volume 7079 of *LNCS*, pages 115–132. Springer, 2011.

[MPS13] Pawel Morawiecki, Josef Pieprzyk, and Marian Srebrny. Rotational cryptanalysis of round-reduced Keccak. In Shiho Moriai, editor, *FSE 2013*, volume 8424 of *LNCS*, pages 241–262. Springer, 2013.

[MRV15] Bart Mennink, Reza Reyhanitabar, and Damian Vizár. Security of full-state keyed sponge and duplex: Applications to authenticated encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015*, volume 9453 of *LNCS*, pages 465–489. Springer, 2015.

[MS16] Amir Moradi and Tobias Schneider. Improved side-channel analysis attacks on Xilinx bitstream encryption of 5, 6, and 7 series. In François-Xavier Standaert and Elisabeth Oswald, editors, *COSADE 2016*, volume 9689 of *LNCS*, pages 71–87. Springer, 2016.

[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 Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT 2010*, volume 6055 of *LNCS*, 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 Emmanuel Prouff and Patrick Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, 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, *ASIACRYPT 2016*, volume 10031 of *LNCS*, pages 602–623, 2016.

[Nat12] National Institute of Standards and Technology. SHA-3 competition. http://csrc.nist.gov/groups/ST/hash/sha-3/index.html, 2007–2012.

[Nat15] 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, August 2015.

[OC14] Colin O'Flynn and Zhizhang (David) Chen. ChipWhisperer: An open-source platform for hardware embedded security research. In Emmanuel Prouff, editor, *COSADE 2014*, volume 8622 of *LNCS*, pages 243–260. Springer, 2014.

[Pie09] Krzysztof Pietrzak. A leakage-resilient mode of operation. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 462–482. Springer, 2009.

[PM16] Peter Pessl and Stefan Mangard. Enhancing side-channel analysis of binary-field multiplication with bit reliability. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 255–270. Springer, 2016.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 142–159. Springer, 2013.

[PSV15] Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 96–108. ACM, 2015.

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In Isabelle Attali and Thomas P. Jensen, editors, *E-smart 2001*, volume 2140 of *LNCS*, pages 200–210. Springer, 2001.

[ROSW16] Eyal Ronen, Colin O'Flynn, Adi Shamir, and Achi-Or Weingarten. IoT goes nuclear: Creating a ZigBee chain reaction. Cryptology ePrint Archive, Report 2016/1047, 2016. http://eprint.iacr.org/2016/1047.

[SPY+10] François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. Leakage resilient cryptography in practice. In Ahmad-Reza Sadeghi and David Naccache, editors, *Towards Hardware-Intrinsic Security – Foundations and Practice*, Information Security and Cryptography, pages 99–134. Springer, 2010.

[STKT06] Kazuhiro Suzuki, Dongvu Tonien, Kaoru Kurosawa, and Koji Toyota. Birthday paradox for multi-collisions. In Min Surp Rhee and Byoungcheon Lee, editors, *ICISC 2006*, volume 4296 of *LNCS*, pages 29–40. Springer, 2006.

[TS14] Mostafa M. I. Taha and Patrick Schaumont. Side-channel countermeasure for SHA-3 at almost-zero area overhead. In *HOST 2014*, pages 93–96. IEEE Computer Society, 2014.

[TS15] Mostafa M. I. Taha and Patrick Schaumont. Key updating for leakage resiliency with application to AES modes of operation. *IEEE Trans. Information Forensics and Security*, 10(3):519–528, 2015.

[UWM17] Thomas Unterluggauer, Mario Werner, and Stefan Mangard. Side-channel plaintext-recovery attacks on leakage-resilient encryption. In *DATE 2017*, 2017. (to appear).

A Algorithms

Algorithm 3: Suffix MAC IsapMac and re-keying function IsapRk.

$\textsc{IsapMac}^{r_1,r_2}_{a,b,c}$-$k$ $(K_A,\ N,\ A,\ C)$	$\textsc{IsapRk}^{r_2}_{b,c}$-$k(K_A, y)$
Input: key $K_A \in \{0,1\}^k$, nonce $N \in \{0,1\}^k$, associated data $A \in \{0,1\}^*$, ciphertext $C \in \{0,1\}^*$ **Output:** tag $T \in \{0,1\}^k$	**Input:** key $K_A \in \{0,1\}^k$, $y \in \{0,1\}^*$ **Output:** sessionkey $K_A^* \in \{0,1\}^k$

$\ell = \lvert A\rvert \mod r_1$
$A_1 \dots A_s \leftarrow r_1$-bit blocks of $A \,\|\, 1 \,\|\, 0^{r_1-1-\ell}$
$\ell = \lvert C\rvert \mod r_1$
$C_1 \dots C_t \leftarrow r_1$-bit blocks of $C \,\|\, 1 \,\|\, 0^{r_1-1-\ell}$
$S \leftarrow N \,\|\, IV_1$
$S \leftarrow p^a(S)$
Absorbing Associated Data
for $i = 1, \dots, s$ **do**
$S \leftarrow p^a((S_{r_1} \oplus A_i) \,\|\, S_{c_1})$
$S \leftarrow S \oplus (0^{r_1+c_1-1} \,\|\, 1)$
Absorbing Ciphertext
for $i = 1, \dots, t$ **do**
$S \leftarrow p^a((S_{r_1} \oplus C_i) \,\|\, S_{c_1})$
Squeezing Tag
$K_A^* \leftarrow \textsc{IsapRk}^{r_2}_{b,c}$-$k(K_A, \lceil S \rceil^k)$
$S \leftarrow p^a(K_A^* \,\|\, \lfloor S \rfloor_{r_1+c_1-k})$
$T \leftarrow \lceil S \rceil^k$
return T
	$\ell = \lvert y\rvert \mod r_2$
	if $\ell = 0$ **then**
	$y_1 \dots y_w \leftarrow r_2$-bit blocks of y
	else
	$y_1 \dots y_w \leftarrow r_2$-bit blocks of $y \,\|\, 0^{r_2-\ell}$
	$S \leftarrow K_A \,\|\, IV_2$
	Absorb
	$S \leftarrow p^c(S)$
	$S \leftarrow (S_{r_2} \oplus y_1) \,\|\, S_{c_2}$
	for $i = 2, \dots, w$ **do**
	$S \leftarrow p^b(S)$
	$S \leftarrow (S_{r_2} \oplus y_i) \,\|\, S_{c_2}$
	Squeeze
	$S \leftarrow p^c(S)$
	$K_A^* \leftarrow \lceil S \rceil^k$
	return K_A^*

Algorithm 4: Encryption and decryption functions.

$\textsc{IsapEnc}_{b,c}^{r_2,r_3}$-$k(K_E, N, P)$	$\textsc{IsapDec}_{b,c}^{r_2,r_3}$-$k(K_E, N, C)$
Input: key $K_E \in \{0,1\}^k$, nonce $N \in \{0,1\}^k$, plaintext $P \in \{0,1\}^*$	**Input:** key $K_E \in \{0,1\}^k$, nonce $N \in \{0,1\}^k$, ciphertext $C \in \{0,1\}^*$
Output: ciphertext $C \in \{0,1\}^*$	**Output:** plaintext $P \in \{0,1\}^*$

$\ell = \lvert N \rvert \mod r_2$
if $\ell = 0$ **then**
$N_1 \dots N_u \leftarrow r_2$-bit blocks of N
else
$N_1 \dots N_u \leftarrow r_2$-bit blocks of $N \,\|\, 0^{r_2-\ell}$
$\ell = \lvert P \rvert \mod r_3$
if $\ell = 0$ **then**
$P_1 \dots P_v \leftarrow r_3$-bit blocks of P
else
$P_1 \dots P_v \leftarrow r_3$-bit blocks of $P \,\|\, 0^{r_3-\ell}$
$S \leftarrow K_E \,\|\, IV_3$
Absorb
$S \leftarrow p^c(S)$
$S \leftarrow (S_{r_2} \oplus N_1) \,\|\, S_{c_2}$
for $i = 2, \dots, u$ **do**
$S \leftarrow p^b(S)$
$S \leftarrow (S_{r_2} \oplus N_i) \,\|\, S_{c_2}$
Squeeze
for $i = 1, \dots, v$ **do**
$S \leftarrow p^c(S)$
$C_i \leftarrow S_{r_3} \oplus P_i$
if $\ell > 0$ **then** $C_v \leftarrow \lceil C_v \rceil^\ell$
return $C_1 \,\|\, \dots \,\|\, C_v$
	$\ell = \lvert N \rvert \mod r_2$
	if $\ell = 0$ **then**
	$N_1 \dots N_u \leftarrow r_2$-bit blocks of N
	else
	$N_1 \dots N_u \leftarrow r_2$-bit blocks of $N \,\|\, 0^{r_2-\ell}$
	$\ell = \lvert C \rvert \mod r_3$
	if $\ell = 0$ **then**
	$C_1 \dots C_v \leftarrow r_3$-bit blocks of C
	else
	$C_1 \dots C_v \leftarrow r_3$-bit blocks of $C \,\|\, 0^{r_3-\ell}$
	$S \leftarrow K_E \,\|\, IV_3$
	Absorb
	$S \leftarrow p^c(S)$
	$S \leftarrow (S_{r_2} \oplus N_1) \,\|\, S_{c_2}$
	for $i = 2, \dots, u$ **do**
	$S \leftarrow p^b(S)$
	$S \leftarrow (S_{r_2} \oplus N_i) \,\|\, S_{c_2}$
	Squeeze
	for $i = 1, \dots, v$ **do**
	$S \leftarrow p^c(S)$
	$P_i \leftarrow S_{r_3} \oplus C_i$
	if $\ell > 0$ **then** $P_v \leftarrow \lceil P_v \rceil^\ell$
	return $P_1 \,\|\, \dots \,\|\, P_v$

Towards Fresh and Hybrid Re-Keying Schemes with Beyond Birthday Security

Publication Data

Christoph Dobraunig, François Koeune, Stefan Mangard, Florian Mendel, and François-Xavier Standaert. "Towards Fresh and Hybrid Re-Keying Schemes with Beyond Birthday Security". In: *Smart Card Research and Advanced Applications, CARDIS 2015*. Ed. by Naofumi Homma and Marcel Medwed. Vol. 9514. LNCS. Springer, 2016, pp. 225–241. URL: https://doi.org/10.1007/978-3-319-31271-2_14

The appended paper is an author-created version.

Contributions

- **Technical:** Contributed to the design of the re-keying schemes presented in Section 3, the observations (attack and fix) on the hybrid re-keying scheme presented in Section 4 and the observations presented in Section 5. No contributions to all proofs given in this paper.
- **Writing:** Contributions to the writing of Sections 2.3, 4, and 5. Minor contributions to the writing of Section 3.2.

Towards Fresh and Hybrid Re-Keying Schemes with Beyond Birthday Security

Christoph Dobraunig[1], François Koeune[2], Stefan Mangard[1], Florian Mendel[1], and François-Xavier Standaert[2].

[1] IAIK, Graz University of Technology, Austria.
[2] Université catholique de Louvain – ICTEAM – Crypto Group, Belgium.
christoph.dobraunig@iaik.tugraz.at, francois.koeune@uclouvain.be, stefan.mangard@tugraz.at, florian.mendel@iaik.tugraz.at,fstandae@uclouvain.be

Abstract. Fresh re-keying is a type of protocol which aims at splitting the task of protecting an encryption/authentication scheme against side-channel attacks in two parts. One part, a re-keying function, has to satisfy a minimum set of properties (such as good diffusion), and is based on an algebraic structure that is easy to protect against side-channel attacks with countermeasures such as masking. The other part, a block cipher, brings resistance against mathematical cryptanalysis, and only has to be secure against single-measurement attacks. Since fresh re-keying schemes are cheap and stateless, they are convenient to use in practice and do not require any synchronization between communication parties. However, it has been shown that their first instantiation (from Africacrypt 2010) only provides birthday security because of a (mathematical only) collision-based key recovery attack recently put forward by Dobraunig et al. (CARDIS 2014). In this paper, we provide two provably secure (in the ideal cipher model) solutions to avoid such collision attacks. The first one is based on classical block ciphers, but does not achieve beyond-birthday CPA security (i.e. it only provably prevents the CARDIS 2014 key recovery attack) and requires an additional block cipher execution in the protocol. The second one is based on tweakable block ciphers and provides tight CPA security while also being more efficient. As a complement, we also show that our reasoning extends to hybrid schemes, where the communication party to protect against side-channel attacks is stateful. We illustrate this claim by describing a collision attack against an example of a hybrid scheme patented by Kocher, and presenting a tweak leading to beyond birthday security. We conclude the paper by discussing the use of fresh/hybrid re-keying for encryption and authentication, together with a cautionary note on their side-channel resistance.

1 Introduction

Designing sound and efficient countermeasures against side-channel attacks is a challenging problem. This is especially true in the context of applications with

The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-31271-2_14

strong cost or energy constraints (e.g. RFIDs, sensor networks, pay-TV, automotive, ...). In such cases, and despite the fact that the devices are likely to be operated in a hostile environment, the direct protection of (e.g.) standard block ciphers such as the AES may be too expensive. As an illustration, the implementation of the well-known masking countermeasure for such block ciphers implies performance overheads that are (at least) quadratic in the security order [11]. Consequently, a new research path has emerged, trying to reduce the adversary's capabilities thanks to re-keying. Leakage-resilient cryptography is the most investigated representative of this trend (see, e.g. [10, 18]). But unfortunately, the concrete guarantees provided by such constructions highly depend on the primitives. For stateful stream ciphers, leakage-resilience indeed delivers strong security levels at low cost. By contrast, for stateless PRFs and PRPs, the situation is less conclusive (essentially due to the fact that the latter primitives bound the number of plaintexts that an adversary can observe, rather than the number of measurements, and hence allow averaging to get noise-free measurements) [3]. Since stateless primitives are essential ingredients for the initialization of an encryption scheme, or for authentication, we are therefore left with the problem of finding good protection mechanisms in this case.

The fresh re-keying scheme proposed in [16] is a typical attempt in this direction. Here, the authors start from the observation that requiring both physical and mathematical security from a single primitive may be too challenging. Therefore, they suggest an alternative solution, where a stateless re-keying function that only has to fulfill a limited number of mathematical properties and is easy to mask is combined with a mathematically strong block cipher. In this context, hardware engineers essentially have to ensure resistance against multi-trace side-channel attacks (aka DPA resistance) for the re-keying function, and resistance against single-trace side-channel attacks (aka SPA resistance) for the block cipher – the latter being an arguably easier task. While such a construction was indeed interesting from a side-channel attack point-of-view, a recent analysis by Dobraunig et al. showed that such a fresh re-keying scheme only provides birthday security against a (mathematical only) chosen-plaintext collision-based key recovery attack [9]. In this paper, we are therefore interested in improved re-keying mechanisms that provide beyond birthday security.

Our contributions. We start by describing two new re-keying schemes – one fresh and one hybrid – with beyond birthday security against the CARDIS 2014 attack. By hybrid, we mean that one communicating party acts like in a stateful scheme, i.e. the fresh key is based on a secret internal state that is continuously updated, while the other communicating party acts stateless, i.e. the session key is always regenerated from the main secret, based on an index value communicated by the first party. In this way, the first party can be protected against side-channel analysis (as in fresh re-keying), without requiring the synchronization burden of fully stateful schemes (e.g. based on a leakage-resilient PRG). For the first scheme, we rely on a provably secure re-keying proposed by Abdalla and Bellare [1], which allows us to prove the security of our (block cipher or hash function based) re-keying in the ideal cipher model, although the bound is quite

loose and does not provide beyond-birthday CPA security (i.e. it only provably prevents the CARDIS 2014 attack). For the second one, we take advantage of tweakable block ciphers to design a very efficient solution, which additionally brings CPA security and benefits from a tight security bound, assuming the existence of an ideal tweakable block cipher. We also suggest concrete instantiations for the building blocks of these schemes, including a couple of new and very efficient tweakable block ciphers based on the TWEAKEY framework [12], proposed in the context of the ongoing CAESAR competition (e.g. Deoxys, Jotlik, KIASU, and Scream) [8].

We complement these new designs with three additional contributions. First, we put forward that a similar reasoning applies to a hybrid re-keying scheme patented by Kocher [13]. That is, such a scheme is also vulnerable to collision attacks (hence only provides birthday security), and can be fixed by taking advantage of tweakable block ciphers. Second, we discuss the use of fresh/hybrid re-keying schemes in concrete applications, and underline important differences between encryption and authentication in this respect. Eventually, we conclude the paper by recalling the side-channel security guarantees of all the proposed re-keying schemes, including their grey areas regarding the interaction between the re-keying function and its underlying block cipher.

Note that resistance against fault attacks is not discussed in this paper, although all the proposed solutions inherit from the good properties of the original Africacrypt re-keying in this respect, and therefore can probably be used to rule out *differential* fault analysis (such as [5] and following works). As in [16], simple fault attacks (e.g. reducing the number of rounds) are considered out of scope, and have to be prevented by other means.

2 Background

2.1 The Africacrypt 2010 fresh re-keying scheme

The Africarypt 2010 fresh re-keying scheme [16] is pictured in Fig. 1. It is built from a block cipher BC and a re-keying function g, and essentially works in two steps. First, a session key k^* is produced by running the re-keying function on the master key k and a random nonce r (selected uniformly at random by the chip needing to be protected). Second this fresh key k^* is used to encrypt a (single) plaintext x with the block cipher. Note that this scheme is trivially tweaked into a hybrid re-keying by replacing the random nonce r by a counter.

2.2 Properties of the g function

Medwed et al. [16] use g to relax the side-channel protection requirements for a block cipher. Informally, the idea is that g will be in charge of generating one-time session keys in a way resistant against side-channel attacks, whereas the block cipher will provide resistance against classical cryptanalysis, but without the need to worry about DPA, as each key is used only once. Since we will re-use

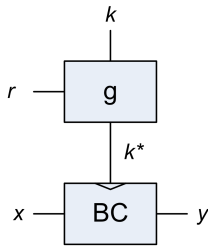


Fig. 1. Africacrypt 2010 fresh re-keying.

the same "separation of duties" strategy and the same g function in the present paper, it is worth recalling the requirements for this function:

1. **Diffusion.** One bit of k^* shall depend on many bits of k.
2. **Stateless Communication.** The parties shall not have an inner state, which has to be kept synchronous.
3. **No additional key material.** k and k^* should have the same size.
4. **Little hardware overhead and side-channel security.** The overhead caused by the use of g (implemented in a secure way) should be small compared to fully protecting the underlying block cipher against side-channel attacks. In other words, the structure of g should make it significantly easier to protect against these attacks (e.g. via masking).
5. **Regularity.** g should have high regularity to facilitate its implementation in a full-custom design (or additional protection mechanisms).

2.3 The CARDIS 2014 collision attack

In this section we describe the attack presented at CARDIS 2014 [9] against the Africacrypt 2010 scheme of Section 2.1. We assume that the generated session keys are used to key a single block cipher encryption. This encryption is used for example in a challenge–response protocol, where the attacker is able to select the challenge x and sees the response $y_i = \mathsf{BC}_{k_i^*}(x)$. The attack can be split into two steps. The first one is the recovery of one session key k_i^*, the second one is the recovery of the master key k out of this knowledge.

The first step is independent of the generation of the session key and targets a single block encryption. In this step the attacker precalculates a list, where he stores pairs of responses (ciphertexts) y_i's and keys k_i^*'s. Those y_i's are encryptions of always the same challenge (plaintext) X for different keys k_i^*. Note that for the creation of the list, all k_i^*'s are chosen by the attacker. Next, in the online phase, the attacker queries an oracle (his target) for multiple encryptions y_i's of the same plaintext X. Since this oracle uses a fresh re-keying scheme, X is encrypted with different keys k_i^*'s and therefore, y_i varies. If such a y_i matches with a y_i in the precalculated list, the corresponding session key is recovered with high probability. For this attack, the best overall complexity of $2 \cdot 2^{n/2}$ is

obtained if a list with $2^{n/2}$ entries is used and $2^{n/2}$ online queries are made (for n-bit session keys).

The second step of the attack depends on the concrete re-keying scheme. In the case of the Africacrypt 2010 proposal, g is a multiplication in a polynomial ring. Since we know one session key k^*, and the corresponding nonce r is invertible with a high probability, we can calculate $k = r^{-1} \cdot k^*$.

3 How to do it right?

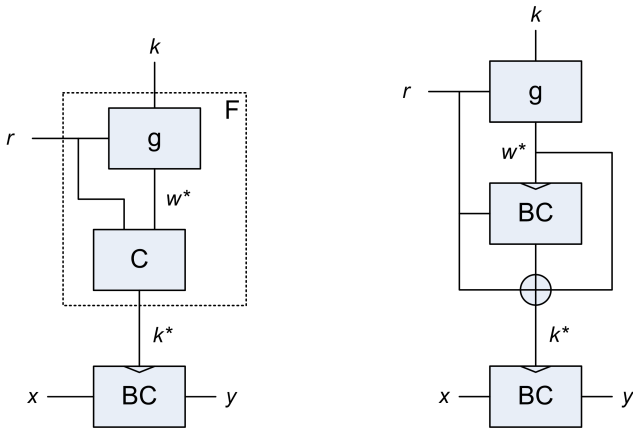
A natural approach to prevent the CARDIS 2014 attack would be to change the instantiation of the g function and to make it non-invertible. For example, one could use a cryptographic hash function for this purpose. Unfortunately, cryptographic hash functions are not easy to protect against side-channel analysis. In order to circumvent this problem, and as already mentioned, we will use the same "separation of duties" strategy as in the Africacrypt re-keying. That is, we will try to separate the burden of side-channel protection from protection against classical cryptanalysis, but this time including collision attacks in our concerns. For this purpose, we present a fresh/hybrid scheme based on a pseudo-random function (PRF) in Section 3.1, and propose an instantiation thereof that is provably secure in the ideal cipher model. We then propose a more efficient solution based on tweakable block ciphers in Section 3.2.

The scenario we focus on in this paper is the case where one communicating party (e.g. the tag) needs re-keying as an easy and cheap protection against side-channel attacks, whereas the other (e.g. the reader) is less cost-sensitive and can be protected through other mechanisms. The re-keying nonce r will thus be randomly chosen by the cheap device and transmitted to the other party. In [15], Mewed et al. considered the scenario where multiple parties must be protected by re-keying and must thus all participate in the selection of r. Their techniques can be straightforwardly applied to our schemes.

3.1 Fresh/hybrid re-keying from the Abdalla-Bellare re-keying

In [1], Abdalla and Bellare proved the security of a PRF-based re-keying scheme. They further suggest to instantiate their PRF with a hash function. Interestingly, such a solution is quite directly applicable in our context. We just need to prove that the combination of g with a well-chosen compression function C is a PRF (represented by the dotted line in Fig. 2a). As previously, the function g shall carry the main burden regarding side-channel protection, whereas the compression function shall prevent collision attacks. Fig. 2a can thus be seen as an extension of the Africacrypt 2010 scheme. Note that here again, the scheme will be stateless if r is a random nonce, and hybrid if it is a counter.

Concretely, and since re-keying schemes are typically combined with block ciphers, we are naturally interested in block cipher-based compression functions. For this purpose, we will analyze one particular construction for C, referred to as the compression function 6 in [6], which Black et al. proved to be pre-image and



(a) General concept (b) Block cipher based

Fig. 2. Fresh/hybrid Abdalla–Bellare re-keying.

collision resistant in the ideal cipher model. Such a solution is pictured in Fig. 2b. However, we note that our construction would keep its security properties with any reasonable instantiation of the PRF in Fig. 2a.

We now prove that, in the ideal cipher model, the generation of k^* is a PRF provided g meets a simple requirement, namely that, for any fixed value K of its first parameter, the function $\mathsf{g}(K,\cdot)$ is one-to-one and, for any fixed value R of its second parameter, the function $\mathsf{g}(\cdot,R)$ is one-to-one (in other words, $\mathsf{g}(K,\cdot)$ and $\mathsf{g}(\cdot,R)$ are permutations of the nonce and key space, respectively). Note that the following result is independent on whether r is based on fresh nonces or a counter.

Theorem 1. *Let us define* $\mathsf{F}(k,r) := \mathsf{BC}_{\mathsf{g}(k,r)}(r) \oplus r \oplus \mathsf{g}(k,r)$. *If* $\mathsf{g}(K,\cdot)$ *and* $\mathsf{g}(\cdot,R)$ *are one-to-one for all values of* K, R, *then the construction* F *is a PRF in the ideal cipher model.*

Proof (sketch). Consider an adversary $\mathcal{A}$ trying to distinguish $\mathsf{F}(k,\cdot)$ from a random function. $\mathcal{A}$ receives access to an oracle $\mathcal{O}$ corresponding to the function to test and to an oracle $\mathcal{O}'$ corresponding to the block cipher. When $\mathcal{A}$ starts, $\mathcal{O}$ tosses a coin to decide how it will behave. Depending on the result, on input r_i, $\mathcal{O}$ will:

- either output a fresh, random y_i;
- or output $y_i = \mathsf{BC}_{\mathsf{g}(k,r_i)}(r_i) \oplus r_i \oplus \mathsf{g}(k,r_i)$, for a fixed value of k. In this case, $\mathcal{O}$ in turn obtains the values $\mathsf{BC}_{\mathsf{g}(k,r_i)}(r_i)$ by querying $\mathcal{O}'$.[1]

[1] As usual, we assume that both $\mathcal{O}$ and $\mathcal{O}'$ act consistently: when receiving an input corresponding to a previous query, they simply replay the previous output (when $\mathcal{O}$ implements F, its consistency is a direct consequence of the consistency of $\mathcal{O}'$).

In addition, $\mathcal{A}$ can directly query $\mathcal{O}'$ with input (k'_i, r'_i) to obtain $\mathsf{BC}_{k'_i}(r'_i)$ or $\mathsf{BC}^{-1}_{k'_i}(r'_i)$.

Consider the case where $\mathcal{O}$ implements the BC-based construction. When $\mathcal{A}$ ends, $\mathcal{O}'$ will thus have received two sets of (possibly intertwined) queries:

- Queries $\mathsf{BC}_{\mathsf{g}(k,r_i)}(r_i)$, through queries to $\mathcal{O}$: we will denote these queries and the corresponding answers as $(r_i, y_i)_{1 \leq i \leq v}$.
- Queries $\mathsf{BC}_{k'_i}(r'_i)$ or $\mathsf{BC}^{-1}_{k'_i}(r'_i)$, through direct calls: we will denote these queries and the corresponding answers as $(k'_i, r'_i, b'_i, y'_i)_{1 \leq i \leq v'}$, where b'_i is a bit equal to 0 (resp. 1) if the request is an encryption (resp. decryption) query.

The central point of the proof is that two queries to $\mathcal{O}'$ yield randomly and independently chosen answers provided the corresponding keys are different. We will now show that this is indeed the case, except with negligible probability.

1. If $r_i = r_j$, then it is easy to see that $y_i = y_j$ (the key is the same, but so is the plaintext too, and $\mathcal{O}'$ receives twice the same query).
2. If $r_i \neq r_j$, then, since $\mathsf{g}(k, \cdot)$ is one-to-one, $\mathsf{g}(k, r_i) \neq \mathsf{g}(k, r_j)$, as requested.
3. Since $\mathsf{g}(\cdot, r_i)$ is one-to-one, k is unknown to $\mathcal{A}$, and $\mathcal{A}$ issues only a polynomial number of requests, the probability to have $k'_j = \mathsf{g}(k, r_i)$ for some (i, j) is negligible (no value of r_i allows reducing the destination space of $\mathsf{g}(k, r_i)$).

Summarizing, if $\mathcal{O}$ implements F, then the computation of all (fresh) output values involves an XOR with $\mathsf{BC}_{\mathsf{g}(k,r_i)}(r_i)$, which, except with negligible probability, are chosen randomly and independently by $\mathcal{O}'$. On the other hand, if $\mathcal{O}$ implements a real random function, then all (fresh) output values are chosen randomly and independently. So, in all cases, all answers received by $\mathcal{A}$ are chosen randomly and independently, except with negligible probability, and none of them allows distinguishing F from a random function.

□

Remarks:

1. The intuition behind this proof is that, without knowledge of k, $\mathcal{A}$ cannot query $\mathcal{O}'$ with keys corresponding to one of the values w^* actually used in the scheme, so that its access to $\mathcal{O}'$ does not help $\mathcal{A}$. Note that the possibility to query $\mathcal{O}'$ with different, but related, keys is not ruled out by the structure of the function g (there could for example be a known difference between $\mathsf{g}(k, r_1)$ and $\mathsf{g}(k, r_2)$, no matter the value of k). However, this is not a problem in the ideal cipher model, where related-key attacks do not apply.
2. $\mathcal{A}$ can of course issue direct queries $(\mathsf{BC}_{k'_i}(r'_{i_1}), \mathsf{BC}_{k'_i}(r'_{i_2}))$, which will not yield independent answers. However, these will obviously not reveal any information on F, since, as shown above, they are unrelated to any query made to F.

3. It is worth noting that the properties we require from g are also in line with a work by Bellare and Kohno. In [4], they propose a formal treatment of related-key attacks by providing the adversary with the ability to issue related-key queries such as $E_{\phi(K)}(m)$, i.e. obtain encryptions with a function ϕ of the (unknown) target key, for a carefully defined set Φ of allowed functions. Bellare and Kohno provide some conditions on the set Φ allowing proving resistance against related-key attacks. Interestingly, our construction can be related to theirs by defining $\phi_i(k) = \mathsf{g}(k, r_i)$, and, with the aforementioned conditions on g, match very well the bounds of [4, Def. 2, Def. 3, Lemma 1]. As our construction is slightly different, this paper provides independent proofs. Nevertheless, the fact that our re-keying function matches independently-defined conditions to avoid related-key attacks is a probable witness of the consistency of our approach.

Being able to prove that the re-keying scheme is a PRF is already of interest regarding the CARDIS 2014 collision attack. As a matter of fact, it guarantees that an attacker cannot distinguish the output of F from a random sequence, which of course also implies that he cannot recover the key k that generated this output. As a consequence, this construction is provably resistant against the collision-based key recovery attack in [9].

In addition, Abdalla and Bellare proved in [1, Theorems 1 and 3] that, if F is a PRF, $\mathcal{SE}$ is an encryption scheme and $\overline{\mathcal{SE}}$ is the associated F-based re-keyed encryption scheme, then the advantage of an adversary trying to break $\overline{\mathcal{SE}}$ can be related to that of adversaries trying to break F and $\mathcal{SE}$ as follows:

$$\mathsf{Adv}^{\mathsf{ind-cpa}}_{\overline{\mathcal{SE}}}(t, lm) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{F}}(t, m) + m \cdot \mathsf{Adv}^{\mathsf{ind-cpa}}_{\mathcal{SE}}(t, l),$$

where t is the adversary's maximum running time, m is the maximum number of keys generated, and l is the maximum number of encryptions performed with each key (so $l = 1$ if we use each fresh key only once).

Unfortunately, this theoretical bound does not provide beyond birthday CPA security. As a matter of fact, an adversary trying $2^{n/2}$ keys against one single block will succeed with probability $2^{-n/2}$, so $\mathsf{Adv}^{\mathsf{ind-cpa}}_{\mathcal{SE}}(2^{n/2}, 1) \geq 2^{-n/2}$. As a consequence, the above bound for an adversary issuing $m = 2^{n/2}$ queries and having computing time $t = 2^{n/2}$ yields:

$$\mathsf{Adv}^{\mathsf{ind-cpa}}_{\overline{\mathcal{SE}}}(2^{n/2}, 2^{n/2}) \leq 1.$$

Interestingly, this bound is tight, since it corresponds to an attack similar to the CARDIS 2014 one, but breaking the CPA game rather than recovering the key. Combined with the observation that the scheme of Figure 2b is also more expensive than the original fresh re-keying scheme, this motivates us to investigate how to overcome these drawbacks, using tweakable block ciphers.

3.2 More efficient solution based on a tweakable block cipher

The scheme presented in Section 3.1 has quite a large performance overhead because of the additional compression/block cipher call needed for a single encryp-

tion. A more efficient construction is to replace this combination by a tweakable block cipher TBC, as shown in Fig. 3. Tweakable block ciphers were introduced by Liskov et al. in [14] as a generalized version of block ciphers. In addition to the secret key, a tweakable block cipher accepts a second parameter (that can be public) called the tweak. Intuitively, "*each fixed setting of the tweak gives rise to a different, apparently independent, family of standard block cipher encryption operators*" and a tweakable block cipher should remain secure even facing an adversary who has control of the tweak.

In our context, the publicly known nonce (or counter) r is used as tweak, and k^* as secret key.

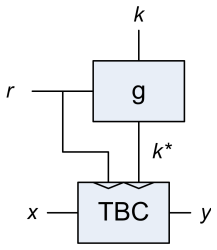


Fig. 3. Fresh/hybrid re–keying with a tweakable block cipher.

This construction again follows the same separation of duties idea as the ones of Fig. 2a and 2b. Namely, the function g is responsible for side-channel protection, whereas the tweakable block cipher prevents the CARDIS 2014 collision attack. Therefore, the requirements for the function g stay the same.

Let us first argue why this construction defeats the aforementioned attacks. We then show it is in fact provably secure in the ideal cipher model.

First recall that the attacks on the re-keying scheme of Africacrypt 2010 exploit the fact that the same plaintext is encrypted with different session keys by the same block cipher. Let us now take into account the fact that $\mathsf{g}(k, \cdot)$ is a permutation. Thus, we get always a different session key k^* for different nonces r. Let us also assume that we have a perfect tweakable block cipher. This means that for every different value of the tweak, we have different and independent block cipher instances. So, basically, we now just use different block ciphers with different keys, and none of them is used with multiple keys, which makes the CARDIS 2014 attack impossible to apply. Taking another viewpoint, an attacker trying to perform the first step of the attack and precalculating a list would now need to do it, not for a set of k_i^*, but for a set of pairs (r_i, k_i^*). This considerably increases the size of the list before the birthday paradox provides a non-negligible chance of success (since this pair has $2n$-bit size).

If we translate this "perfect TBC" assumption in the ideal cipher model, it simply means that each different values of the key *or* the tweak yields a different, independent permutation. We now prove that, in this model, the construction

depicted on Fig. 3 is indeed a TBC (here too, note that the result is independent on whether r is based on fresh nonces or a counter).

Theorem 2. *Let* TBC *be an ideal tweakable block cipher, and let us define* TBC' *as* $\mathsf{TBC}'_k(r,m) = \mathsf{TBC}_{\mathsf{g}(k,r)}(r,m)$. *If* $\mathsf{g}(K,\cdot)$ *and* $\mathsf{g}(\cdot,R)$ *are one-to-one for all values of* K, R, *then* TBC' *is a tweakable block cipher.*

Proof (sketch). Consider a distinguisher $\mathcal{D}$ trying to distinguish TBC'_k from a family of independent random permutations. At the beginning of the experiment, an oracle $\mathcal{O}$ tosses a coin to decide which construction it will implement.

- In the first case, $\mathcal{O}$ chooses a random key k and sets $E(r,m) := \mathsf{TBC}_{\mathsf{g}(k,r)}(r,m)$ and $E^{-1}(r,m) := \mathsf{TBC}^{-1}_{\mathsf{g}(k,r)}(r,m)$. In this case, $\mathcal{O}$ in turn obtains these values by querying an oracle $\mathcal{O}'$ implementing the ideal tweakable block cipher.
- In the second case, $\mathcal{O}$ chooses a family $\Pi(\cdot,\cdot)$ of independent random permutations[2] and sets $E(r,m) := \Pi(r,m)$ and $E^{-1}(r,m) := \Pi^{-1}(r,m)$.

$\mathcal{D}$ can then query $\mathcal{O}$ to obtain $E(r_i,m_i)$ or $E^{-1}(r_i,m_i)$ for values (r_i,m_i) of its choice. In addition, $\mathcal{D}$ can also directly query $\mathcal{O}'$ to obtain $\mathsf{TBC}_{k'_i}(r'_i,m'_i)$ or $\mathsf{TBC}^{-1}_{k'_i}(r'_i,m'_i)$ for values (k'_i,r'_i,m'_i) of its choice. The goal of $\mathcal{D}$ is to discover which construction $\mathcal{O}$ implements. We will denote the maximum number of (direct or indirect) queries $\mathcal{D}$ makes to $\mathcal{O}'$ as l.

Consider the case where $\mathcal{O}$ implements the TBC-based construction. When $\mathcal{D}$ ends, $\mathcal{O}'$ will thus have received two sets of (possibly intertwined) queries:

- Queries $\mathsf{TBC}_{\mathsf{g}(k,r_i)}(r_i,m_i)$ or $\mathsf{TBC}^{-1}_{\mathsf{g}(k,r_i)}(r_i,m_i)$, through queries to $\mathcal{O}$: we will denote these queries and the corresponding answers as $(r_i,m_i,b_i,y_i)_{1\leq i\leq v}$, where b_i is a bit equal to 0 (resp. 1) if the request is an encryption (resp. decryption) query.
- Queries $\mathsf{TBC}_{k'_i}(r'_i,m'_i)$ or $\mathsf{TBC}^{-1}_{k'_i}(r'_i,m'_i)$, through direct calls: we will denote these queries and the corresponding answers as $(k'_i,r'_i,m'_i,b'_i,y'_i)_{1\leq i\leq v'}$, where b'_i is a bit equal to 0 (resp. 1) if the request is an encryption (resp. decryption) query.

Observe that:

1. If $r_i \neq r_j$ (resp. $r'_i \neq r'_j$ and/or $k'_i \neq k'_j$), then y_i and y_j (resp. y'_i and y'_j) are chosen randomly and independently by $\mathcal{O}'$.
2. The same is true when $r_i \neq r'_j$: $\mathcal{O}'$ is queried on different tweak values and provides independent random answers.
3. If $r_i = r_j$, then y_i and y_j are chosen randomly and independently, except that the permutation rule (i.e. different inputs yield different outputs) and consistency rule (i.e. $E(E^{-1}(r,m)) = m$) will be respected. Since Π is also a permutation, none of these limitations helps $\mathcal{D}$ guessing the construction implemented by $\mathcal{O}$.

[2] That is, for each T, $\Pi(T,\cdot)$ is a random permutation of the message space.

4. The same argument applies if $(k'_i, r'_i) = (k'_j, r'_j)$.
5. Finally, if $r_i = r'_j$, then, since k is unknown and $\mathsf{g}(\cdot, r_i)$ is one-to-one, the probability to have $k'_j = g(k, r_i)$ is only $\frac{1}{2^n}$. In all other cases, $\mathcal{O}'$ is queried on different key values and provides independent random answers. Since $\mathcal{O}'$ received a maximum of l queries, the global probability is bounded by $\frac{l}{2^n}$.

It is easy to see that, in the case where $\mathcal{O}$ implements a family of independent random permutations, $\mathcal{O}$ will bear exactly the same behavior, except in one case. This only exception is that there is no corresponding to case 5 above ($k'_j = \mathsf{g}(k, r_i)$). This difference of behavior would help discovering the behaviour of $\mathcal{O}$, but, as argued above, only occurs with probability $\frac{l}{2^n}$. In all other cases, answers received by $\mathcal{D}$ are always random, independent values consistent with a permutation.

The distinguishing advantage is thus bounded by:

$$\Pr[D^{\Pi}(t, l) = 1] - \Pr[D^{\mathsf{TBC}'}(t, l) = 1] \leq \frac{l}{2^n}.$$

□

Having proved that our construction is a TBC, we can for example easily prove that encryption based on it is secure against chosen-plaintext attacks.

Theorem 3. *Let* TBC *be a tweakable block cipher and define* $\Pi =< G, E, D >$*, where* $E_k(m)$ *is performed by choosing a random* r *and returning* $E_k(m) := (r, \mathsf{TBC}_k(r, m))$*. In the ideal tweakable cipher model,* Π *provides indistinguishable encryption against a chosen-plaintext adversary.*

Proof (sketch). Consider an adversary $\mathcal{A}$ attacking Π. During the oracle query phases (both before and after the challenge phase), $\mathcal{A}$ can query an encryption oracle $\mathcal{O}$ to obtain the encryption $(r'_i, \mathsf{TBC}_k(r'_i, m'_i))$ of messages he chooses[3], under an unknown key k chosen uniformly at random. During the challenge phase, $\mathcal{A}$ outputs two messages m_0, m_1 and receives $c = (r, \mathsf{TBC}_k(r, m_b))$ from $\mathcal{O}$. His goal is to discover b. In both cases, $\mathcal{O}$ answers by querying a TBC oracle $\mathcal{O}'$. $\mathcal{A}$ can also directly query $\mathcal{O}'$ with values (m''_i, k''_i, r''_i) of its choice and obtain $E'_{k''_i}(r''_i, m''_i)$ or $E'^{-1}_{k''_i}(r''_i, m''_i)$. We will denote by l the maximum number of queries (both to $\mathcal{O}$ and $\mathcal{O}'$) issued by $\mathcal{A}$.

Observe that:

- As k was chosen uniformly at random, the probability to have $k = k''_i$ for some i is bounded by $\frac{l}{2^n}$. In all other cases, answers to direct queries to $\mathcal{O}'$ are independent from b (queries on different key values).
- Similarly, the probability to have $r = r'_i$ for some i is bounded by $\frac{l}{2^n}$. In all other cases, answers to queries to $\mathcal{O}$ are independent from b (queries on different tweak values).

[3] The tweak r'_i, however, is randomly chosen by $\mathcal{O}$.

As a consequence,

$$\mathsf{Adv}_{\Pi}^{\mathsf{ind-cpa}}(l) \leq \frac{1}{2} + \frac{2l}{2^n}.$$

□

Remarks:

- The proof also holds in the stateful case. The only difference is that the case $r = r_i$ can then never happen, resulting in a slightly better bound, namely $\frac{1}{2} + \frac{l}{2^n}$.
- CPA security obviously assumes that the adversary cannot control the random nonce r used for encryption. By contrast, it is worth noting that the construction of Theorem. 1 did not need to prevent this control of r by the adversary in order to achieve a PRF, and is thus slightly more general.
- Interestingly, we see that the use of a TBC brings the same advantage over block cipher-based constructions (i.e. natural beyond-birthday security) as in the context of authenticated encryption [14].

3.3 Concrete instantiations

Instantiating the previous fresh re-keying schemes essentially requires to specify a block cipher BC, a re-keying function g, and possibly a tweakable block cipher TBC. For the block cipher, a natural choice is the AES. For the re-keying function, Medwed et al. [16] proposed this polynomial multiplication in $\mathbb{F}_{2^8}[y]$ modulo $p(y) = y^{16} + 1$:

$$g \,:\, (\mathbb{F}_{2^8}[y]/p(y))^2 \to \mathbb{F}_{2^8}[y]/p(y), \quad (k, r) \mapsto k \cdot r.$$

A polynomial multiplication globally fits our goals. Unfortunately, the choice $p(y) = y^{16} + 1$ is not suitable for our purpose[4]. As a matter of fact, this polynomial is not irreducible, which implies that the requirement that $\mathsf{g}(K, \cdot), \mathsf{g}(\cdot, R)$ are one-to-one is not strictly satisfied. To meet this requirement, an irreducible polynomial (e.g. $p(y) = y^{16} + y^3 + y +$ "14", using the Rijndael notation for $\mathbb{F}_{2^8}$ elements) should be used instead. This impacts performance a bit, namely making shuffling and implementation in protected logic-styles slightly more expensive than in [16], but this impact is limited, and DPA-protection of this g function remains cheap. Note that the analysis of the side-channel behaviour of the polynomial multiplication has only been done for random nonces r by Medwed et al. [16]. Thus, the polynomial multiplication should only be used in this scenario. Finding instances of g that behave well in the hybrid case is an interesting scope for further research. For the tweakable block cipher, we suggest to use the efficient instances listed in introduction (i.e. Deoxys, Jotlik, KIASU, and Scream). AES-based solutions can also be exploited, by considering the block cipher-based constructions in [14]. Mennink also proposes optimal techniques to build a tweakable block cipher from a block cipher in [17] .

[4] The authors thank Marcel Medwed for pointing this out during this paper's presentation.

4 Application to a hybrid scheme by Kocher

We now investigate another alternative to hybrid re-keying that was patented by Kocher in [13]. We first recall that this scheme can be compromised using the CARDIS 2014 collision attack. We then describe how to fix it with a tweakable block cipher.

4.1 The CARDIS 2014 attack against Kocher's hybrid re-keying

Description of the scheme. In Kocher's re-keying, and in contrast with the schemes of Section 2.1, the session key is not derived from a static secret master key with the help of randomly generated nonces. Instead, the secret itself is updated, changed, and used as session key. The update is based on the tree structure that is depicted in Fig. 4.

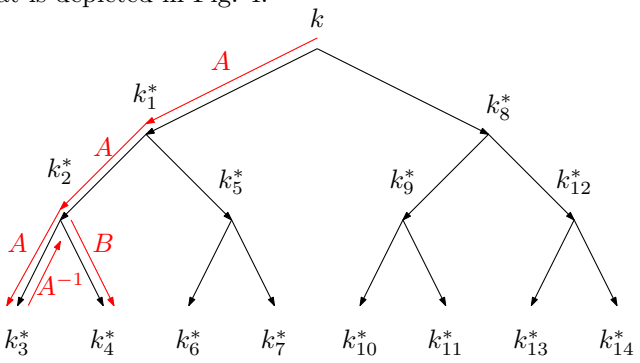


Fig. 4. Hybrid re-keying patented by Kocher [13].

The root of the tree is the secret master key k and the other vertices represent session keys k_i^*. To traverse through the tree, the functions A, B, A^{-1}, and B^{-1} are used, where A^{-1}, and B^{-1} are the inverse functions of A, and B. For instance if we want to go from k to k_1^* in Fig. 4, we calculate $k_1^* = A(k)$. If we want to go from k to k_8^*, we calculate $k_8^* = B(k)$. The number of usable session keys k_i^* is determined by the depth of the tree, which has to be fixed in advance.

We assume for this scheme a similar use-case as supposed for the Africacrypt 2010 scheme. Thus, one party (e.g. the tag) needs easy and cheap protection against side-channel attacks, whereas the other party (e.g. the reader) has to be protected by other mechanisms. To realize this, the tag strictly follows the tree, which means that it uses the session keys in the strict order given in Fig. 4 $(k_1^*, k_2^*, ..k_n^*)$. The tag tells the other party (the reader) the index i of the currently used session key k_i^*, and the reader calculates the session key k_i^* starting from the root k.

Note that other variants of this scheme are possible. For instance, session keys corresponding to internal vertices can be used three times (every time a vertex is visited). By doing so, the number of transitions for the tag can be limited to

one. In other words, this reuse of session keys allows the tag to perform only one of the operations A, B, A^{-1}, or B^{-1} per change of the session key.

Collision attack. As already hinted by Dobraunig et al. [9], the CARDIS 2014 collision attack also applies to re-keying schemes like Kocher's one. For simplicity, we make the same assumptions as in Section 2.3 (i.e. each session key is used in a single block cipher execution, and the attacker can choose the plaintext which is encrypted). In addition, we assume that the concrete instance of Kocher's scheme only uses every session key once.

The first step of the CARDIS 2014 attack described in Section 2.3 is to recover one session key: this step goes exactly as before, i.e. building (offline) a database of encryptions of the same plaintext with various keys, then relying on the birthday paradox to obtain collisions with (online) encryptions of the same plaintext. Next, and as far as the second step is concerned, if we additionally assume that the used operations (permutations) A, B, A^{-1}, and B^{-1} are publicly known, one recovered session key k_i^* and the corresponding index i are enough to recover the master key k. Note that keeping the A, B, A^{-1}, and B^{-1} permutations secret would typically mean implementing them as a block cipher with secret (fixed) key, which would then be a potential target to DPA (i.e. lead to a chicken and egg problem, essentially).

4.2 Efficient fix based on a tweakable block cipher

To get rid of the CARDIS 2014 attack, we propose a fix for Kocher's scheme based on a tweakable block cipher similar to Section 3.2. Here, the session key k_i^* – generated by Kocher's scheme (Fig. 4) – is used as a secret key for the tweakable block cipher, and the index i of the key is used as tweak, as illustrated in Fig. 5. The collision attack does not work in this case, for the same reasons as described in Section 3.2 (namely, different tweaks, and so virtually different block ciphers, are used with different keys).

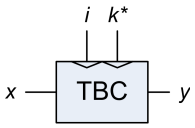


Fig. 5. Hybrid re–keying with a tweakable block cipher.

5 Encryption vs. authentication issues

Our discussion so far dealt with the generic problem of designing a secure and side-channel resistant re-keying scheme. Since they originally borrow from Abdalla and Bellare, our solutions typically lead to secure encryption per se. However, it is worth mentioning that additional security issues might arise when

dealing with other uses of this re-keying scheme. In particular, although we ruled out key recovery attacks by ensuring that the master key cannot be recovered from a session key, it should be pointed out that, in some contexts such as authentication, even the recovery of one single session key might already be a serious threat.

Consider a simple challenge–response protocol carried out between two parties. In this scenario we have a tag that acts as prover and a reader that acts as verifier. So in the first step, the reader sends the challenge x_i to the tag. The tag encrypts the challenge and responds with $y_i = E_{k_i^*}(x_i)$. We assume that a re-keying scheme is used, which changes the key k_i^* for every new call of the encryption. As before, one of the n-bit session keys k^* can be recovered with a complexity of about $2 \cdot 2^{n/2}$, regardless of the re-keying scheme actually used.

Now examine the implications of one session key recovery in this scenario, for actual re-keying schemes. If the Africacrypt 2010 scheme is used to generate the session key k^*, the tag (prover) decides alone of the nonce value r. So an attacker does not necessarily need the master key. It is already enough to have one session key to pass the challenge–response protocol, since the attacker can force the use of a single recovered session key. Concretely, the attacker would first select a plaintext value X and build off-line a DB of encryptions of X with random keys. He would then play the role of a reader and query a genuine tag with challenge X. Finally, having recovered one session key, he would be able to impersonate the tag by always using r as nonce. The same attack would also succeed against the tree-based session key generation scheme of Section 4.1.

This attack cannot be prevented by making $\mathtt{g}$ not invertible. In fact, this attack always applies to every re-keying scheme, as long as a block cipher is re-keyed for every encryption and the prover can determine the session key to be used. This means that this attack might be also applicable to schemes where the CARDIS 2014 attack does not work. Belaïd et al. [2] presented such a scheme, where the re-keying function $\mathtt{g}$ of the Africacrypt 2010 scheme is replaced by a non-invertible function. However, the applicability of the session key replay attack depends on the actual method to determine the nonce, which is not specified by Belaïd et al.

In general, there exist two countermeasures against such a session key replay attack. The first one is to have both parties contributing to the selection process of the session key, as is already done in the CARDIS 2011 scheme [15]. The other one is to prohibit the recovery of the session key in the first place. Interestingly, the constructions using tweakable block ciphers we propose in Sections 3.2 and 4.2 do prevent this recovery. Indeed, as discussed in Section 3.2, changing the value of r will not only result in a different session key k^*, but also in a different tweak value and hence – assuming a perfect tweakable block cipher – virtually in a different block cipher. As a consequence, the collision attack to recover session keys is not applicable any more in this case.

6 Conclusion: side-channel security

We conclude this paper with a cautionary note regarding the exact security improvements brought by fresh re-keying regarding side-channel attacks. For this purpose, the first positive observation is that if the adversary targets the re-keying function and the block cipher independently, the resulting security guarantees are well understood. That is, the implementation will be secure as long as g resists DPA and BC (or TBC) resists SPA. But quite naturally, security arguments and proofs also indicate what are the potential weak points of a construction, and this is clearly the case for fresh re-keying. Looking at Fig. 1, 2a and 2b, this potential weak point is indeed the interaction between the re-keying function and the block cipher. That is, if some leakage about k^* (in Fig. 1) or w^* (in Fig. 2a, 2b) is obtained by the adversary (e.g. when recombining the shares after the masked execution of g), attacks combining mathematical cryptanalysis and leakage (e.g. the algebraic SPA described in [15]) are likely to be very powerful to accumulate partial information on the master key k and finally recover it. This is why fresh re-keying crucially relies on the SPA security of the block cipher, and a secure implementation should recombine the shares of the fresh keys in a sufficiently secure, i.e. typically shuffled, manner (as clearly mentioned in [15] as well). Note that this observation does not annihilate the interest of fresh re-keying which still significantly reduces the adversary's attack paths (compared to the straightforward execution of a block cipher). Interestingly, a very similar situation can be found for the SPRING primitive discussed in [7], which also aims at an informal separation between the parts of the primitive that are easy to mask, and those that are not (and therefore need to be carefully shuffled). Besides, such an issue does not directly apply to Kocher's hybrid scheme which does not make use of a g function. Indeed, combining small leakage on the session keys would require to go through the permutations A and B (and their inverses), which may not be easy if they are implemented with fixed key block ciphers. But this comes at the cost of a slightly more expensive key update mechanism. Besides, and more importantly, it makes any attempt to secure both the encrypting/proving and the decrypting/verifying parties of a protocol much more challenging, since the tree-based construction in Fig. 4 has to be stateless. In other words, it does not benefit from the malleability of the g function which is exploited for this purpose in multi-parties fresh re-keying [15], which gives a typical application of the "no free lunch" theorem.

Acknowledgments. The authors thank Christophe Petit for useful advice. This work has been supported in part by the Austrian Science Fund (project P26494-N15), by the Austrian Research Promotion Agency (FFG) under grant number 845589 (SCALAS), by the Brussels Region Research Funding Agency through the program Secur'IT and by the European Commission through the ERC project 280141 (CRASH) and the COST Action CRYPTACUS. F.-X. Standaert is a research associate of the Belgian Fund for Scientific Research (FNRS-F.R.S.).

References

1. Abdalla, M., Bellare, M.: Increasing the lifetime of a key: A comparative analysis of the security of re-keying techniques. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 546–559. Springer (2000)
2. Belaïd, S., De Santis, F., Heyszl, J., Mangard, S., Medwed, M., Schmidt, J., Standaert, F., Tillich, S.: Towards fresh re-keying with leakage-resilient PRFs: cipher design principles and analysis. J. Cryptographic Engineering 4(3), 157–171 (2014)
3. Belaïd, S., Grosso, V., Standaert, F.: Masking and leakage-resilient primitives: One, the other(s) or both? Cryptography and Communications 7(1), 163–184 (2015)
4. Bellare, M., Kohno, T.: A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 491–506. Springer (2003)
5. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Kaliski Jr., B.S. (ed.) CRYPTO '97. LNCS, vol. 1294, pp. 513–525. Springer (1997)
6. Black, J., Rogaway, P., Shrimpton, T., Stam, M.: An analysis of the blockcipher-based hash functions from PGV. J. Cryptology 23(4), 519–545 (2010)
7. Brenner, H., Gaspar, L., Leurent, G., Rosen, A., Standaert, F.: FPGA implementations of SPRING - and their countermeasures against side-channel attacks. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 414–432. Springer (2014)
8. CAESAR Competition: http://competitions.cr.yp.to/caesar-submissions.html
9. Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F.: On the security of fresh re-keying to counteract side-channel and fault attacks. In: Joye, M., Moradi, A. (eds.) CARDIS 2014. LNCS, vol. 8968, pp. 233–244. Springer (2014)
10. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS 2008. pp. 293–302. IEEE Computer Society (2008)
11. Grosso, V., Standaert, F., Faust, S.: Masking vs. multiparty computation: how large is the gap for AES? J. Cryptographic Engineering 4(1), 47–57 (2014)
12. Jean, J., Nikolic, I., Peyrin, T.: Tweaks and keys for block ciphers: The TWEAKEY framework. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 274–288. Springer (2014)
13. Kocher, P.C.: Leak-resistant cryptographic indexed key update (2003), US Patent 6,539,092
14. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 31–46. Springer (2002)
15. Medwed, M., Petit, C., Regazzoni, F., Renauld, M., Standaert, F.: Fresh re-keying II: securing multiple parties against side-channel and fault attacks. In: Prouff, E. (ed.) CARDIS 2011. LNCS, vol. 7079, pp. 115–132. Springer (2011)
16. Medwed, M., Standaert, F., Großschädl, J., Regazzoni, F.: Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 279–296. Springer (2010)
17. Mennink, B.: Optimally secure tweakable blockciphers. In: Leander, G. (ed.) Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9054, pp. 428–448. Springer (2015), `http://dx.doi.org/10.1007/978-3-662-48116-5_21`
18. Yu, Y., Standaert, F., Pereira, O., Yung, M.: Practical leakage-resilient pseudo-random generators. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) CCS 2010. pp. 141–151. ACM (2010)

Side-Channel Analysis of Keymill

Publication Data

Christoph Dobraunig, Maria Eichlseder, Thomas Korak, and Florian Mendel. "Side-Channel Analysis of Keymill". In: *Constructive Side-Channel Analysis and Secure Design, COSADE 2017*. Ed. by Sylvain Guilley. Vol. 10348. LNCS. Springer, 2017, pp. 138–152. URL: https://doi.org/10.1007/978-3-319-64647-3_9

The appended paper is an author-created version available at https://eprint.iacr.org/2016/793. This version is a minor revision correcting Figure 5.

Contributions

- **Technical:** Contributed to idea and concept of the attack. No contributions to the simulations and the practical evaluation.
- **Writing:** Contributions to the writing of Sections 1, 2, 3, and 5.

Side-Channel Analysis of Keymill

Christoph Dobraunig, Maria Eichlseder, Thomas Korak, and Florian Mendel

Graz University of Technology, Austria
christoph.dobraunig@iaik.tugraz.at

Abstract. One prominent countermeasure against side-channel attacks, especially differential power analysis (DPA), is fresh re-keying. In such schemes, the so-called re-keying function takes the burden of protecting a cryptographic primitive against DPA. To ensure the security of the scheme against side-channel analysis, the re-keying function has to withstand both simple power analysis (SPA) and differential power analysis (DPA). Recently, at SAC 2016, Taha et al. proposed Keymill, a side-channel resilient key generator (or re-keying function), which is claimed to be inherently secure against side-channel attacks. In this work, however, we present a DPA attack on Keymill, which is based on the dynamic power consumption of a digital circuit that is tied to the $0 \rightarrow 1$ and $1 \rightarrow 0$ switches of its logical gates. Hence, the power consumption of the shift-registers used in Keymill depends on the $0 \rightarrow 1$ and $1 \rightarrow 0$ switches of its internal state. This information is sufficient to obtain the internal differential pattern (up to a small number of bits, which have to be brute-forced) of the 4 shift-registers of Keymill after the nonce has been absorbed. This leads to a practical key-recovery attack on Keymill.

Keywords: side-channel analysis · fresh re-keying · differential power analysis

1 Introduction

Side-channel attacks like differential power analysis (DPA) pose a serious threat to devices operating in a hostile environment. Such scenarios quite naturally appear in our current information infrastructure whenever an entity has physical access to a device which uses a cryptographic key that must be kept secret from this entity. Hence, it is necessary to protect such devices against the extraction of the secret key by means of side-channel analysis like SPA and DPA [7]. In particular, for resource-constrained or low-cost devices that are used for the Internet of Things or in RFID applications, the use of protection mechanisms is not straightforward, since applied protection mechanisms have to be cheap and efficient. One protection mechanism that suits such applications very well is fresh re-keying.

Fresh re-keying [9] is an approach for precluding DPA on cryptographic primitives. The resistance against DPA is achieved by a separation-of-duties principle, where a re-keying function takes the burden of protection against DPA away from the cryptographic primitive. In this construction, the re-keying function

This article is a minor revision of an article that appears in the proceedings of COSADE 2017. The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-64647-3_9.

processes a nonce and master key to compute a fresh session key. This session key is then used by the cryptographic primitive. The nonce, or initial value (IV), is generated uniquely for each encryption, and must never be reused for another encryption. The nonce is considered public information and has to be transmitted to (or synchronized with) the decrypting recipient together with the ciphertext. Since the cryptographic primitive is only called once per session key, DPA attacks are naturally prevented, and only dedicated countermeasures against SPA are needed. However, the re-keying function has to provide resistance against SPA and DPA attacks, either by its design, or by application of countermeasures like threshold implementations [10], masking [12], hiding [2], shuffling [6], etc. The intention behind re-keying schemes is that the re-keying function itself can be protected more easily against DPA than the cryptographic scheme, or that it can even be designed to provide inherent security against DPA. Both options profit from the fact that the re-keying function itself does not need to fulfill strong cryptographic requirements [9].

Re-keying functions. Medwed et al. [9] proposed polynomial multiplication as re-keying function, which has further been extended to the multi-user setting [8]. While such a polynomial multiplication lacks inherent protection against DPA, it is easy to mask and additionally allows easy-to-implement countermeasures against SPA, such as shuffling [9]. However, Pessl and Mangard [11] showed at CT-RSA 2016 that this multiplication is vulnerable to side-channel analysis, in particular at the point where its masks have to be combined and the session key is used in the cryptographic scheme. Additionally, the original scheme by Medwed et al. is susceptible to time-memory trade-off attacks [3]. Recently at Crypto 2016, Dziembowski et al. [4] presented a more formal treatment of re-keying functions and proposed two schemes. The first is based on learning parity with leakage, the second on learning with rounding, and both are efficient and easy to mask.

Keymill. In contrast to designs relying on side-channel countermeasures like masking for side-channel protection, Keymill [14] claims to be secure against side-channel analysis inherently by design without requiring any redundant circuit. Having a re-keying function which provides inherent security against side-channel analysis is beneficial with respect to implementation metrics. Since such schemes do not require masking to withstand DPA, no randomness is needed to create and update masks, and masks do not have to be stored and processed in the first place. A comparison of a modular multiplication and Keymill by Taha et al. [14] shows that a hardware implementation of Keymill requires 775 gate equivalents (GE), while an implementation of a modular multiplication with first-order masking requires 7300 GE [9].

To achieve such low implementation costs, Keymill only uses 4 nonlinear feedback shift-registers taken from the stream cipher Achterbahn [5]. The shift-registers are connected via a rotating cross-connect, which shifts the output of each shift-register's nonlinear feedback function into another shift-register. This

cross-connect joins the function outputs with shift-register inputs cyclically per clock. For this construction and also for a toy example consisting of two 8-bit registers involving a similar rotating cross-connect, the authors claim that no DPA attacks are feasible without constructing a hypothesis for the whole key, or equivalently for the whole internal state of the four shift-registers, and thus render DPA attacks infeasible.

Our Contribution. In this work, we present a DPA attack on Keymill. Our attack shows that the claim of Keymill to be inherently secure against side-channel attacks without the need of additional circuits does not hold. The basic idea of the attack is as follows. Instead of making a hypothesis about the exact values of the internal state bits or the secret key, we target the internal difference between neighboring bits of the shift-registers. As observed by Burman et al. [1], and Zadeh and Heys [15], the dynamic power consumption of shift-registers depends on the number of internal differences of neighboring bits. The more internal differences we have, the more power the shift-register consumes. We recover those internal differences bit by bit by comparing the power consumption of a reference nonce (e.g., 0), with power traces of a modified nonce where a single bit has been flipped. Knowing these internal differences allows to recover the full state and consequently the master key by guessing a few additional bits.

Our attack requires the attacker to obtain traces for related (partially chosen) pairs of nonce values, but without violating the single-use requirement for nonces. This scenario is explicitly covered by the security claim of Keymill, although similar to chosen-plaintext attacks, it might not be easy to collect such data in a practical application. We verified the validity and robustness of the attack both for simulated data and for measurements from an FPGA implementation of Keymill.

Outline. In Sect. 2, we give a brief background on fresh re-keying and restate the specification of Keymill. Then, we describe the side-channel attack on Keymill and on a variant of Toy Model II given in the Keymill specification in Sect. 3. Sect. 4 gives experiments for our attack and discusses the influence of different levels of noise. Finally, we conclude in Sect. 5.

2 Background

In this section, we first give a brief introduction to the concept of fresh re-keying, where we restate the requirements on re-keying functions. Then, we briefly summarize the specification of Keymill and finally, discuss time-memory trade-off attacks on such re-keying schemes.

2.1 Fresh Re-Keying

Fresh re-keying has been proposed by Medwed et al. [9] as a countermeasure against side-channel and fault attacks for low-cost devices. A typical scenario

where fresh re-keying can be applied is the communication of an RFID tag with an RFID reader. Typically, RFID tags are low-cost devices that additionally have strict requirements regarding power consumption, not allowing costly protection mechanisms against side-channel and fault attacks of the implemented cryptographic primitives. This stands in contrast to the more expensive RFID readers, where costly protection mechanisms like masking are usually affordable.

Fig. 1 shows the working principle of fresh re-keying in a communication scenario between an RFID reader and an RFID tag. For sending a message, the tag generates a nonce and derives a session key k^* by using a re-keying function g. This session key is then used by the block cipher E to encrypt the message m. The ciphertext c together with the nonce is sent to the reader, where it can be decrypted.

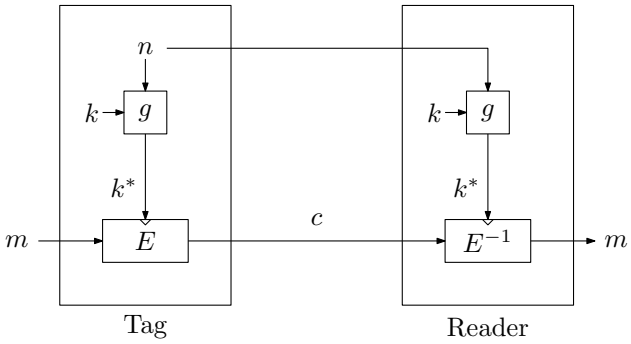


Fig. 1. Fresh re-keying scheme of Medwed et al. [9].

Since the nonce is generated by the tag, the tag can ensure that the block cipher E is always used with a new session key k^*, which will preclude DPA on the block cipher. However, in the case of the reader, having a unique nonce cannot be ensured, because the nonce is received over the communication channel and thus, might be chosen by an attacker. Therefore, the implementation of the block cipher E of the reader has to be protected against DPA by other means. Apart from that, the implementation of g for both entities has to withstand DPA, because here, the master key k is processed with a different nonce. On the designer's side, the challenge is to find a suitable re-keying function g which fulfills the following six properties given by Medwed et al. [9]:

1. Good diffusion of the master key k.
2. No synchronization between parties. Hence, g should be stateless.
3. No need for additional key material.
4. Little hardware overhead. Total costs lower than protecting E alone.
5. Easy protection against side-channel attacks.
6. Regularity.

One option for a re-keying function is the polynomial multiplication in $\mathbb{F}_{2^8}[y]$ modulo $p(y)$ proposed by Medwed et al. [9]:

$$g : (\mathbb{F}_{2^8}[y]/p(y))^2 \to \mathbb{F}_{2^8}[y]/p(y), \quad (k, n) \mapsto k \cdot n.$$

2.2 Brief Description of Keymill

Keymill [14] is a new keystream generator recently proposed by Taha et al. at SAC 2016. In contrast to the fresh re-keying scheme by Medwed et al. discussed in Sect. 2.1, Keymill does not only provide one session key k^*, instead it provides a keystream. As indicated in Fig. 2, this is particularly useful when encrypting longer messages that require several block cipher calls. The nonce n is required to be unique, but is otherwise public.

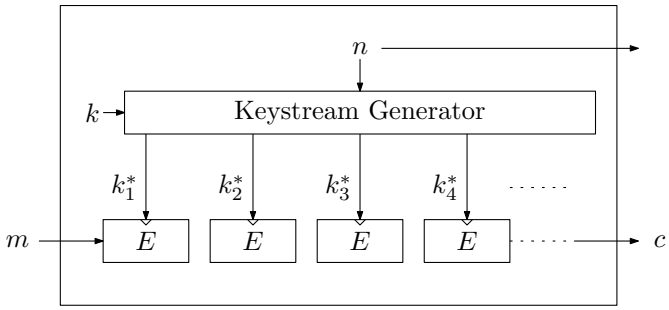


Fig. 2. Re-keying using a keystream generator as shown in [14].

Keymill operates on an internal state of 128 bits, composed of 4 NLFSRs as shown in Fig. 3. Shift-register R_0 has 31 bits, shift-registers R_1 and R_2 have 32 bits, and shift-register R_3 has 33 bits. The feedback functions F_0, F_1, F_2 and F_3 are selected from the set of feedback functions used for the stream cipher Achterbahn [5]:

$$\begin{aligned}
F_0(S) = {} & s_0 + s_2 + s_5 + s_6 + s_{15} + s_{17} + s_{18} + s_{20} + s_{25} + s_8 s_{18} + s_8 s_{20} \\
& + s_{12} s_{21} + s_{14} s_{19} + s_{17} s_{21} + s_{20} s_{22} + s_4 s_{12} s_{22} + s_4 s_{19} s_{22} \\
& + s_7 s_{20} s_{21} + s_8 s_{18} s_{22} + s_8 s_{20} s_{22} + s_{12} s_{19} s_{22} + s_{20} s_{21} s_{22} \\
& + s_4 s_7 s_{12} s_{21} + s_4 s_7 s_{19} s_{21} + s_4 s_{12} s_{21} s_{22} + s_4 s_{19} s_{21} s_{22} \\
& + s_7 s_8 s_{18} s_{21} + s_7 s_8 s_{20} s_{21} + s_7 s_{12} s_{19} s_{21} + s_8 s_{18} s_{21} s_{22} \\
& + s_8 s_{20} s_{21} s_{22} + s_{12} s_{19} s_{21} s_{22} \\
F_1(S) = F_2(S) = {} & s_0 + s_3 + s_{17} + s_{22} + s_{28} + s_2 s_{13} + s_5 s_{19} + s_7 s_{19} \\
& + s_8 s_{12} + s_8 s_{13} + s_{13} s_{15} + s_2 s_{12} s_{13} + s_7 s_8 s_{12} + s_7 s_8 s_{14} \\
& + s_8 s_{12} s_{13} + s_2 s_7 s_{12} s_{13} + s_2 s_7 s_{13} s_{14} + s_4 s_{11} s_{12} s_{24} \\
& + s_7 s_8 s_{12} s_{13} + s_7 s_8 s_{13} s_{14} + s_4 s_7 s_{11} s_{12} s_{24} + s_4 s_7 s_{11} s_{14} s_{24}
\end{aligned}$$

$$
\begin{aligned}
F_3(S) = {} & s_0 + s_2 + s_7 + s_9 + s_{10} + s_{15} + s_{23} + s_{25} + s_{30} + s_8 s_{15} + s_{12} s_{16} \\
& + s_{13} s_{15} + s_{13} s_{25} + s_1 s_8 s_{14} + s_1 s_8 s_{18} + s_8 s_{12} s_{16} + s_8 s_{14} s_{18} \\
& + s_8 s_{15} s_{16} + s_8 s_{15} s_{17} + s_{15} s_{17} s_{24} + s_1 s_8 s_{14} s_{17} + s_1 s_8 s_{17} s_{18} \\
& + s_1 s_{14} s_{17} s_{24} + s_1 s_{17} s_{18} s_{24} + s_8 s_{12} s_{16} s_{17} + s_8 s_{14} s_{17} s_{18} \\
& + s_8 s_{15} s_{16} s_{17} + s_{12} s_{16} s_{17} s_{24} + s_{14} s_{17} s_{18} s_{24} + s_{15} s_{16} s_{17} s_{24}
\end{aligned}
$$

Note that all feedback functions are nonsingular and additionally do not depend on the first bit $s_{\ell-1}$ of each ℓ-bit register, that is, they are of the form

$$F_j(S) = F_j(s_0, \ldots, s_{\ell-1}) = s_0 + F'_j(s_1, \ldots, s_{\ell-2}).$$

The outputs of the feedback functions are then mixed via a rotating cross-connect, depending on the current clock cycle index i:

$$F_j \to R_{j+i \pmod 4} \quad \text{for } j = 0, 1, 2, 3.$$

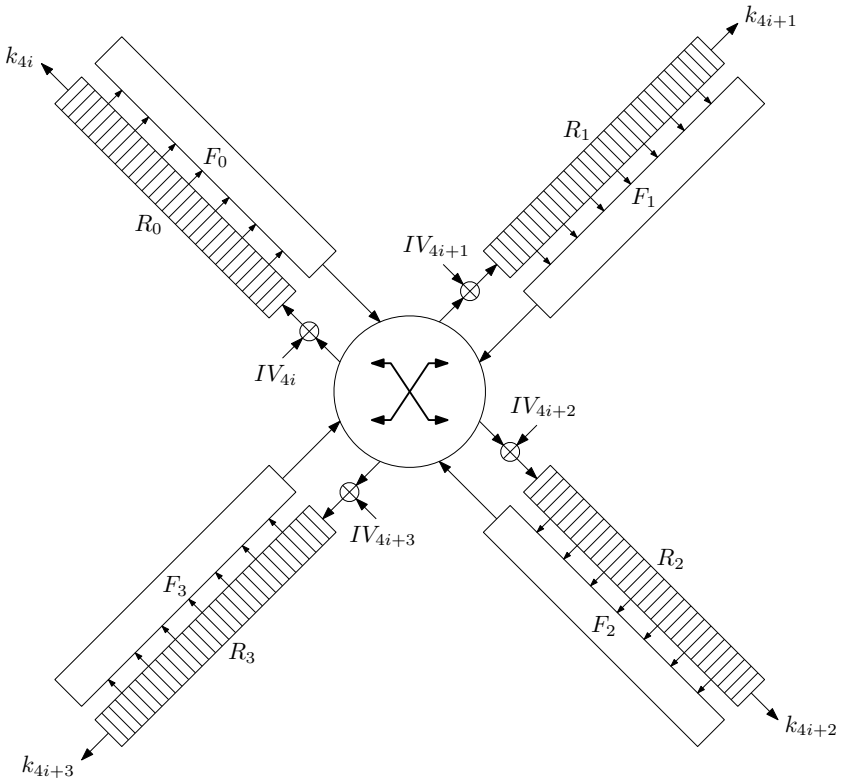


Fig. 3. Structure of Keymill

After loading the 128-bit secret key into the internal state, 4 bits of the 128-bit nonce that can be monitored (or controlled) by the attacker are added

to the feedback functions of the shift-registers in each clock cycle. After absorbing the nonce in 32 clock cycles, the internal state is clocked 33 more times before producing any output. Afterwards 4 bits of output are generated (one from each shift-register) in each clock cycle. We refer to the specification of Keymill [14] for a more detailed description.

The designers claim that this construction "expands the size of any useful key hypothesis to the full entropy" [14]. More specifically, they claim that the SCA-security ("the minimum size of a key hypothesis (in bits) such that the leakage-model using the correct key correlates to the measured leakage significantly higher than the leakage-model using any other key" [14]) is about 128 bits.

2.3 Remark on Time-Memory Trade-Off Attacks

As elaborated in [3], the re-keying scheme proposed by Medwed et al. [9] is susceptible to time-memory trade-off attacks dependent on the used re-keying function. For instance, if a polynomial multiplication is used together with AES-128, the master key can be recovered with a complexity of 2^{65} [3]. Since Keymill has an internal state-size of 128-bits, similar attacks are possible on the scheme shown in Fig. 2.

3 Side-Channel Attack on Keymill

In this section, we will present side-channel attacks on Keymill. First, we discuss the power consumption of shift-registers following the work of Zadeh and Heys [15] and show how this power consumption can be used to recover the differences of neighboring shift-register bits. This and the fact that the first bits of the shift-registers are not used in the feedback functions of Keymill allows us to mount a side-channel attack. For simplicity, we first demonstrate the attack on a variant of Toy Model II given in the Keymill specification [14] and afterwards discuss the application to Keymill.

3.1 Power Consumption of a Shift-Register

In all our attacks, we exploit the dynamic power consumption of the shift-registers at the triggering edge of the clock (i.e., positive edge). More specifically, we observe the dynamic power consumption of the building blocks of the shift-registers, the D-flip-flops. As shown by Zadeh and Heys [15], the dynamic power consumption of a D-flip-flop at the triggering edge depends on whether its state changes or not. If the state of the D-flip-flop changes, more power is consumed than if it remains the same. As an example, Zadeh and Heys [15] analyze a D-flip-flop constructed out of 6 NAND gates. For such a flip-flop, 3 gates change if the flip-flop changes its state, whereas only one gate changes if not.

Next, we have a look at the power consumption of a shift-register. For simplicity, consider a 4-bit shift-register consisting of 4 flip-flops D_0, D_1, D_2, and D_3. In the following, we assume that D_4 is the input of our shift-register, which

is shifted towards D_0. For instance, let us consider the power consumption of the change from state $S_0 = \texttt{0110}_2$ to state $S_1 = \texttt{1101}_2$. For this transition, D_0 changes its state, D_1 keeps its state, D_2 changes its state, and D_3 changes its state. Since the power consumption of the flip-flops is higher if they change their state, the power consumption of the shift-register is correlated with the Hamming weight of $S_0 \oplus S_1$ ($= \texttt{1011}_2$). In this example, 3 flip-flops change their state.

Now, we want to consider a state change from S_0 to S'_1, where we shift in a `0` instead of a `1` as before. So we observe the power consumption for the change from state $S_0 = \texttt{0110}_2$ to state $S'_1 = \texttt{1100}_2$. If this transition happens, only two flip-flops change their state. Thus, we observe for the transition $S_0 \rightarrow S'_1$ a smaller power consumption than for $S_0 \rightarrow S_1$. This allows us to derive information about the difference of the bits stored in D_4 and D_3 of S'_1 and S_1, respectively. In more detail, we know that they are equal for S'_1 and different for S_1. We will use this observation in our side-channel attack on a variant of Toy Model II and Keymill itself in the following sections.

3.2 Attack on Toy Model II

For the sake of simplicity, we first describe the working principle of our attack on a slightly modified variant of Toy Model II given in the Keymill specification [14], which has only two 8-bit shift-registers. In the attack, we assume that similar to Keymill, the output of the first flip-flop of each shift-register is not connected to the feedback function, as shown in Fig. 4. Besides nonsingularity, this is the only assumption on the feedback function that is necessary to mount our attack. We do not rely on any other specific properties of the feedback functions. The shift-register is preinitialized with the secret key. After that, the 16-bit nonce is absorbed, 2 bits per clock cycle. Our goal is to recover all *internal differences* of both shift-registers after the nonce (e.g., $n = \texttt{0000}_{16}$) has been absorbed.

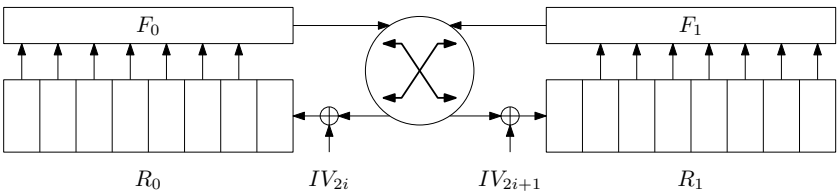


Fig. 4. Structure of modified Toy Model II

First, we collect two power traces, one for a nonce starting with $\texttt{00}_2$ and one for a nonce starting with $\texttt{10}_2$. We look at the power consumption when the first two bits of the nonce are absorbed in the first cycle. Here, we have a difference in n_0 for R_0, but equal values in n_1 for R_1. Since the first flip-flop of each shift-register is not connected to the feedback function, the circuit processes the same

information for both initial values, except for the first flip-flop of the left shift-register R_0. As already discussed in Sect. 3.1, this gives us information about the difference of the first two bits of R_0 after absorbing the first two bits of the nonce. If the power consumption when absorbing 00_2 is higher than in the 10_2 case, we know that the first two bits of R_0 are different after 00_2 is absorbed. If the power consumption is lower, then they are equal.

Next, we use two initial values starting with 00_2 and 01_2. This allows us to learn the internal difference of the first two bits of the shift-register R_1 after 00_2 is absorbed. Then, we use 0000_2 and 0010_2 to learn information of the difference of the first two bits after 0000_2 has been absorbed, still preserving the information of the difference of the now second and third bits of both shift-registers learned in the steps before. By continuing in this way, we can learn the differences of all neighboring bits of R_0 and R_1 after the nonce 0000_{16} has been absorbed.

Now, guessing one bit in each shift-register determines the other 7 bits in each shift-register. Hence, we are left with only 4 possible internal states. From this states on, we can invert Toy Model II step by step until we get 4 key candidates in total. Note that inversion of a fully known state is trivial due to the nonsingularity of the feedback functions, which allows to recover the previous last bit s_0 from the known feedback output and the known values of the other taps. Overall, if we are able to obtain noiseless measurements for about 16 chosen nonces (one per bit of the state), we can recover the entire key k.

3.3 Attack on Keymill

Compared to Toy Model II, Keymill is essentially the same, except everything is larger. As described in Sect. 2.2, we have 4 shift-registers: one 31-bit shift-register, two 32-bit shift-registers, and one 33-bit shift-register. The 128-bit nonce is absorbed in 32 cycles, each cycle taking 4 bits. Furthermore, the 4 feedback functions of Keymill do not consider the outputs of the first flip-flop of each shift-register. As mentioned before, this fact is exploited in our side-channel attack. Again, we want to recover the internal differential pattern of the used shift-registers after a certain nonce, e.g., $n = 0 \cdots 0$ has been absorbed. Please note that the all 0 nonce is just an example taken for simplicity. The attack works for every other choice of the nonce.

The attack proceeds in a similar way as described in Sect. 3.2. First, we record a power trace for a nonce starting with 0000_2 and a second trace for a nonce starting with 1000_2. We compare the power consumption for the two traces at the time the first nibble of the nonce is absorbed. At this time, for both traces, the processed values are equal except for the inputs of shift-register R_0. Since the output of the first flip-flop of R_0 is not fed back into the feedback function, the power consumption differs only because of the state changes of this flip-flop. As discussed in Sect. 3.1, this is sufficient to recover the difference of the first two bits of shift-register R_0. The power traces of nonces starting with 0100_2, 0010_2, and 0001_2 can be used to learn the difference of shift-registers R_1, R_2 and R_3, respectively.

When the second nibble of the nonce is absorbed, those differences are shifted by one position, but are still known, if the first nibble of the nonce starts with 0000_2. Hence, we can use nonces starting with $0000\,0000_2$, $0000\,1000_2$, $0000\,0100_2$, $0000\,0010_2$, and $0000\,0001_2$ and learn the differences of the first two bits of each shift-register, while retaining the knowledge of the differences between the second and the third bits. Proceeding this way, we can learn at most 32 differences of neighboring bits per shift-register.

This means that we can learn all internal differences of all 4 shift-registers, since one shift-register has 31 bits, two have 32 bits and one has 33 bits. So, at most 30, two times 31, and 32 differences have to be learned. Since we know all internal differences of each shift-register, a guess of one state bit in each shift-register determines all others. Thus, guessing 4 bits in total leads to 16 different states we recover. From these states, we can invert Keymill, resulting in 16 possible key candidates in total.

Summarizing, if we can obtain noiseless measurements for about 128 chosen nonces, then we can recover the full internal state and consequently the secret key k. In particular, we recover the internal state bit by bit by making a hypothesis on 1 bit of "equivalent key information", instead of an actual key bit value: The xor difference of two neighboring state bits.

3.4 A Note on Filtering the Noise

The success of our attacks crucially depends on the ability to distinguish power consumption changes for a change of the input values. This means that the noise level has to be small enough to reliably identify these changes. If the attacker is allowed to repeat nonces, averaging the traces and filtering the noise is no problem. Even if the nonce is required to be unique (as usually the case), this can easily be done, since the state of the shift-registers only depends on bits of the nonce that have already been absorbed. Hence, we can use all the remaining nonce bits after the relation we want to recover to average the power consumption for this cycle. For Keymill, we can average over up to 16 power traces even if we recover bit relations in the penultimate nonce-absorbing cycle. Dependent on the noise level, it might happen that the last few internal differences of the state cannot be recovered anymore, since there are too few traces to filter the noise. So these bits might have to be guessed additionally at the end of the attack.

4 Practical Evaluation

In order to show the practicability of the attacks discussed in Sect. 3, we present two experiments. First, we run the attack based on simulated leakage traces to analyze the impact of noise on the success of the attack. For the second evaluation, we use power measurements from an FPGA implementation of Keymill to evaluate the practicability of the attack targeting real hardware.

First, we simulate the described attack targeting the proposed Keymill design as shown in Fig. 3. Therefore, the four registers $R_0 \dots R_3$ and the corresponding

feedback functions $F_0 \dots F_3$, which compose the four NLFSRs, have been modelled in software. At the start of the simulation, the registers are initialized with the secret key. Then, for every clock cycle, the simulation returns the Hamming distance produced by the shift registers. The current Hamming distance depends on the values in the shift register, the results of the feedback functions $F_0 \dots F_3$ and the nonce.

Gaussian noise with zero mean ($\mu_{\text{noise}} = 0$) and varying standard deviation σ_{noise} can be added to the noise-free Hamming-distance measurements ($\text{HD}_{\text{noisefree}}$) in order to simulate measurements captured from real hardware, i.e. HD_{meas} (see Equation 1). In order to minimize the influence of the noise it is possible to repeat the simulation with a similar nonce t times for calculating the mean of the measurements.

$$\text{HD}_{\text{meas}} = \text{HD}_{\text{noisefree}} + \text{noise}, \qquad \text{where noise} \leftarrow \mathcal{N}(0, \sigma_{\text{noise}}). \tag{1}$$

For every setting (specific σ_{noise} and specific t), we performed $N_{\text{full}} = 500$ experiments with randomly chosen initial states of the four shift-registers $R_0 \dots R_3$ to calculate the success rate SR of the attack,

$$\text{SR} = \frac{N_{\text{success}}}{N_{\text{full}}},$$

where N_{success} is the number of successful state recoveries. Fig. 5 depicts the results of this simulation. It is clearly visible that SR decreases with increasing noise. This effect can be compensated by repeating the attack with the same nonce t times and calculate the mean of the measurements. For $t = 1$, the success rate starts to decrease for noise levels above $\sigma_{\text{noise}} = 0.1$. For $t = 50$, the success rate remains 1 up to a noise level of $\sigma_{\text{noise}} = 1.3$.

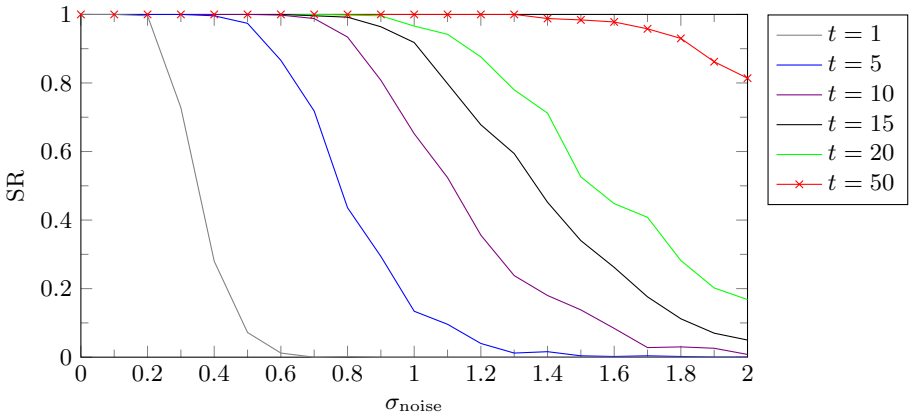


Fig. 5. Success rate (SR) for increasing noise levels (σ_{noise}). For the graphs different numbers (1–50) of Hamming-distance measurements have been used for calculating the mean Hamming distance.

Fig. 6 shows the influence of σ_{noise} on the Hamming-distance measurements (HD_{meas}). For this specific plot, $\text{HD}_{\text{noisefree}} = 64$ has been selected. The '+' markers represent single HD measurements. In the noise-free scenario, i.e. $\sigma_{\text{noise}} = 0$, all HD measurements have the value 64. For a high noise level, i.e. $\sigma_{\text{noise}} = 2$, the HD measurements are in the range between 58 and 70.

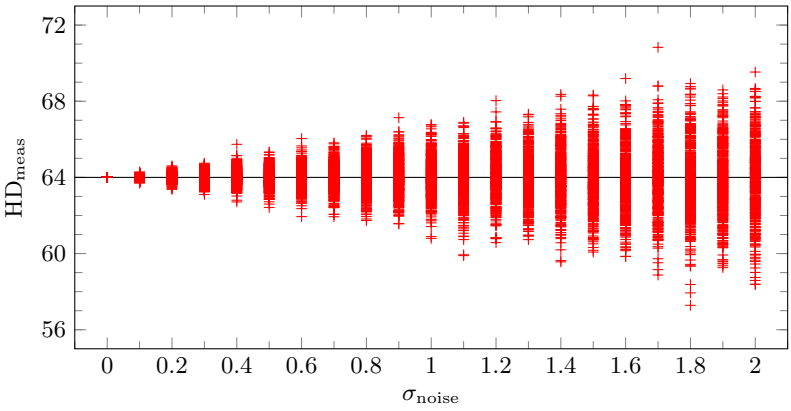


Fig. 6. Hamming distance measurements for increasing σ_{noise}, $\text{HD}_{\text{noisefree}} = 64$.

In a final experiment, Keymill is evaluated on real hardware. We chose the Sakura G board [13], which is the reference platform for side-channel evaluations of cryptographic hardware designs on FPGAs. The main FPGA (Xilinx Spartan-6 LX75) has been configured with the Keymill design and the power consumption during the initialization (i.e. the first 33 clock cycles where the bits of the nonce are shifted into the shift registers, four bits per clock cycle) has been measured with an oscilloscope. For every bit position of the nonce, two trace sets have been recorded, one with the corresponding bit set to '0' and one with the corresponding bit set to '1'. In order to evaluate the number of traces required for reaching a specific success rate, 10 000 traces have been recorded for every nonce. The results of the evaluations are depicted in Fig. 7. It shows that for the given FPGA implementation, at least 220 measurements for every nonce are required for reaching a success rate of 1. In scenarios where repeated measurements of the same nonce are prohibited, iterating over the last 8 bits of the nonce can be done to average the measurements. This leads to 256 traces per fixed 120 bits that can be used to filter the noise.

Comparing the means of the two trace sets allows to distinguish between the Hamming distances. The higher amount of traces required for reaching a success rate of 1 indicates that the noise on real hardware is significantly larger than the noise during previously performed simulations. For the sake of completeness we have performed the simulations for $t = 220$ and larger noise levels. The results show that for $\sigma_{\text{noise}} \geq 3.6$ the success rate starts to decrease for $t =$

220. Experiments on the real hardware reveal that for recovering the whole initial state, approximately $220 \cdot 128 = 28\,160$ measurements are required in total. The applied measurement setup allows us to collect the required amount of measurements for reaching a success rate of 1 within an hour. With some improvements of the setup the measurement time could be reduced to a few minutes, but this was not the goal of this work.

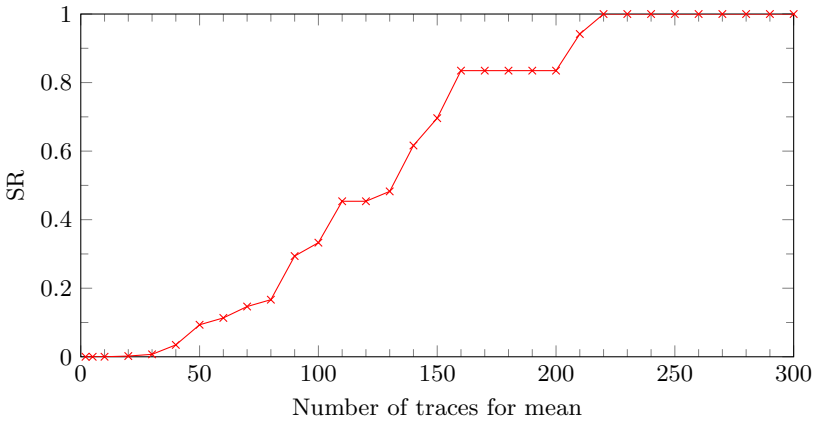


Fig. 7. Evolution of the success rate (SR) for the attack on the FPGA with increasing number of traces for calculating the mean.

5 Conclusion

In this work, we showed that a DPA attack on Keymill is feasible. In contrast to the DPA attacks that are claimed to be thwarted by the specification of Keymill, we do not make hypotheses on the actual values of Keymill's key or internal state. Instead, we first recover the internal differences of neighboring bits step by step from side-channel measurements, and then take advantage of the resulting entropy reduction to recover the actual values. Our attack violates the claim by the designers that Keymill is inherently secure against side-channel attacks by design. Indeed, we show that Keymill needs dedicated countermeasures against DPA attacks exploiting internal differences.

Our attack requires the ability of an attacker to choose the nonces. Therefore, guaranteeing that only random nonces can be used seems to be an efficient countermeasure. Although this prevents a straightforward application of our attack to recover all differences between state-bits, the recovery of just a fraction of the differences of the first few bits still remains possible. Hence, it is part of future work to evaluate if extensions of the presented attack concept are applicable for random nonces.

Acknowledgments. This work has been supported in part by the Austrian Science Fund (project P26494-N15) and by the Austrian Research Promotion Agency (FFG) under grant number 845589 (SCALAS).

References

1. Burman, S., Mukhopadhyay, D., Veezhinathan, K.: LFSR based stream ciphers are vulnerable to power attacks. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 384–392. Springer (2007)
2. Clavier, C., Coron, J.S., Dabbous, N.: Differential power analysis in the presence of hardware countermeasures. In: Koç, Ç.K., Paar, C. (eds.) CHES 2000. LNCS, vol. 1965, pp. 252–263. Springer (2000)
3. Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F.: On the security of fresh re-keying to counteract side-channel and fault attacks. In: Joye, M., Moradi, A. (eds.) CARDIS 2014. LNCS, vol. 8968, pp. 233–244. Springer (2014)
4. Dziembowski, S., Faust, S., Herold, G., Journault, A., Masny, D., Standaert, F.X.: Towards sound fresh re-keying with hard (physical) learning problems. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 272–301. Springer (2016)
5. Gammel, B.M., Göttfert, R., Kniffler, O.: Achterbahn-128/80. eSTREAM, ECRYPT Stream Cipher Project (2006)
6. Herbst, C., Oswald, E., Mangard, S.: An AES smart card implementation resistant to power analysis attacks. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 239–252 (2006)
7. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) CRYPTO '99. LNCS, vol. 1666, pp. 388–397. Springer (1999)
8. Medwed, M., Petit, C., Regazzoni, F., Renauld, M., Standaert, F.X.: Fresh rekeying II: Securing multiple parties against side-channel and fault attacks. In: Prouff, E. (ed.) CARDIS 2011. LNCS, vol. 7079, pp. 115–132. Springer (2011)
9. Medwed, M., Standaert, F.X., Großschädl, J., Regazzoni, F.: Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 279–296. Springer (2010)
10. Nikova, S., Rijmen, V., Schläffer, M.: Secure hardware implementation of nonlinear functions in the presence of glitches. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 218–234. Springer (2008)
11. Pessl, P., Mangard, S.: Enhancing side-channel analysis of binary-field multiplication with bit reliability. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 255–270. Springer (2016)
12. Prouff, E., Rivain, M.: Masking against side-channel attacks: A formal security proof. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 142–159. Springer (2013)
13. Sakura-G – Side-Channel Evaluation Board. `http://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-G.html`, accessed: 2016-11-28
14. Taha, M., Reyhani-Masoleh, A., Schaumont, P.: Keymill: Side-channel resilient key generator. In: Avanzi, R., Heys, H. (eds.) SAC 2016. LNCS, Springer (2016), (to appear). eprint version: `http://eprint.iacr.org/2016/710`
15. Zadeh, A.A., Heys, H.M.: Simple power analysis applied to nonlinear feedback shift registers. IET Information Security 8(3), 188–198 (2014)

Statistical Fault Attacks on Nonce-Based Authenticated Encryption Schemes

Publication Data

Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Victor Lomné, and Florian Mendel. "Statistical Fault Attacks on Nonce-Based Authenticated Encryption Schemes". In: *Advances in Cryptology – ASIACRYPT 2016*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. LNCS. 2016, pp. 369–395. URL: `https://doi.org/10.1007/978-3-662-53887-6_14`

The appended paper is an author-created extended version available at `https://eprint.iacr.org/2016/616`. This extended version has adapted text in Sections 2.1, and 4 and provides a new Appendix B.

Contributions

- **Technical:** Contributed to the concept of the attacks, the simulation of the attacks, and the code for performing the key recovery. No contributions to the practical execution/implementation of the attacks.
- **Writing:** Contributions to the writing of Sections 2.3, and 3. Minor contributions to the writing of Section 1.

Statistical Fault Attacks on Nonce-Based Authenticated Encryption Schemes

Christoph Dobraunig[1], Maria Eichlseder[1], Thomas Korak[1], Victor Lomné[2], and Florian Mendel[1]

[1] Graz University of Technology, Graz, Austria
firstname.lastname@iaik.tugraz.at
[2] ANSSI, Paris, France
victor.lomne@ssi.gouv.fr

Abstract. Since the first demonstration of fault attacks by Boneh et al. on RSA, a multitude of fault attack techniques on various cryptosystems have been proposed. Most of these techniques, like Differential Fault Analysis, Safe Error Attacks, and Collision Fault Analysis, have the requirement to process two inputs that are either identical or related, in order to generate pairs of correct/faulty ciphertexts. However, when targeting authenticated encryption schemes, this is in practice usually precluded by the unique nonce required by most of these schemes.
In this work, we present the first practical fault attacks on several nonce-based authenticated encryption modes for AES. This includes attacks on the ISO/IEC standards GCM, CCM, EAX, and OCB, as well as several second-round candidates of the ongoing CAESAR competition. All attacks are based on the Statistical Fault Attacks by Fuhr et al., which use a biased fault model and just operate on collections of faulty ciphertexts. Hereby, we put effort in reducing the assumptions made regarding the capabilities of an attacker as much as possible. In the attacks, we only assume that we are able to influence some byte (or a larger structure) of the internal AES state before the last application of MixColumns, so that the value of this byte is afterwards non-uniformly distributed.
In order to show the practical relevance of Statistical Fault Attacks and for evaluating our assumptions on the capabilities of an attacker, we perform several fault-injection experiments targeting real hardware. For instance, laser fault injections targeting an AES co-processor of a smart-card microcontroller, which is used to implement modes like GCM or CCM, show that 4 bytes (resp. all 16 bytes) of the last round key can be revealed with a small number of faulty ciphertexts.

Keywords: fault attacks · authenticated encryption · CAESAR · Differential Fault Attacks (DFA) · Statistical Fault Attacks (SFA)

1 Introduction

Fault attacks pose a serious threat for cryptographic implementations. For this kind of attacks, the analyzed device is operated outside its defined operating

conditions, which can lead to erroneous outputs. By analyzing the erroneous output data, secret information can be revealed. In the worst case, a single fault can reveal the entire secret key of a block cipher like AES, which has been shown to be feasible by many researchers in the last decade [7,33]. Popular techniques to inject faults include modifications of the power supply [50] or the clock source [6] by injecting glitches. Other methods, such as laser fault injection [45], have been proven even more powerful, because they additionally allow a precise localization of the fault injection.

While fault attacks on block ciphers and stream ciphers have received a great deal of attention from the scientific community, authenticated ciphers have been arguably less popular targets among researchers. At the same time, they describe an important class of cryptographic algorithms with many applications in information security. Authenticated encryption provides both confidentiality and authentication of data to two parties communicating via an insecure channel. This is essential for many applications such as SSL/TLS, IPSEC, SSH, or hard-disk encryption. In most applications, there is not much value in keeping the data secret without ensuring that it has not been intentionally or unintentionally modified. For this reason, in practical applications, block ciphers like AES are typically used mainly as a building block for an authenticated encryption scheme.

An authenticated encryption scheme is usually modeled as a function with four inputs: a unique nonce N, associated data A, plaintext P, and secret key K. It generates two outputs: the ciphertext C, and the authentication tag T:

$$\mathcal{E}(K, N, A, P) = (C, T).$$

The corresponding decryption algorithm takes the secret key K, nonce N, authenticated data A, ciphertext C, and tag T, and either outputs the plaintext P if the verification tag is correct, or $\perp$ if the verification of the tag failed:

$$\mathcal{D}(K, N, A, C, T) \in \{P, \perp\}.$$

It is usually assumed (and typically essential for the security of the authenticated encryption scheme) that nonces never repeat for encryptions $\mathcal{E}$ under the same key K. We refer to such schemes as nonce-based authenticated encryption. While some schemes claim a certain level of robustness even in misuse settings (such as repeated nonces, or release of unverified plaintext), this does not mean that they are intended to be intentionally misused in practical implementations: repeating nonces always incurs a certain loss of security.

An interesting consequence of the unique nonce in the encryption procedure is the implicitly provided protection against several classes of fault attacks [11,12,49]. In particular, Differential Fault Analysis (DFA) [11] is rendered almost impossible, since an attacker is unable to observe both the correct and the faulty output for the same input, if the attacker cannot fix the value of the nonce. Moreover, in contrast to nonce-based (but unauthenticated) encryption schemes (such as CBC, CTR, etc.), where the decryption procedure (with a fixed nonce) is still susceptible to DFA, this is not the case for nonce-based authenticated encryption schemes that only return the plaintext if the tag is correct.

For this reason, all published fault attacks on authenticated encryption schemes so far are in settings where either the nonce is repeated, or unverified plaintext is released [42, 43].

These observations might lead to the impression that nonce-based authenticated encryption schemes are not susceptible to fault attacks and thus, no dedicated fault attack countermeasures might be necessary to protect the implemented scheme against these attacks. However, in this work, we show that this assumption is not true, and present the first fault attacks on authenticated encryption schemes that are not performed in some kind of misuse scenario. We show that countermeasures against fault attacks are essential for implementations of authenticated encryption schemes operating in hostile environments.

Our Contribution. We present fault attacks for a wide range of authenticated encryption schemes. Our attacks do not require any misuse scenario, such as nonce reuse or release of unverified plaintext. We focus our discussion on various AES-based schemes, including the ISO/IEC standards CCM [48], GCM [32], EAX [9], and OCB [40], as well as several second-round CAESAR [46] candidates. However, our analysis is applicable to a broader range of constructions and is not limited to AES-based schemes.

All our attacks are based on an enhancement of the Statistical Fault Attack (SFA) presented by Fuhr et al. [18], which requires only very limited assumptions about the attacker's capabilities: the ability to induce a fault that leads to a biased (non-uniform) distribution in certain bytes. In case of AES, we assume that the attacker is able to influence some byte (or a larger structure) of the internal state of AES before the last application of MixColumns, so that the value of this byte is non-uniformly distributed. Particularly, we do not have to rely on the exact position of a fault, the number of faults injected during a single encryption, or even the knowledge that a certain fault has happened at all in an individual encryption. All we need to do is to collect ciphertexts and estimate the distribution of a single byte for various key guesses.

In order to evaluate the assumptions on the capabilities of an attacker, we also perform fault-injection experiments targeting three different hardware platforms. In the first setting, clock glitch attacks on a GCM software implementation executed on an 8-bit microcontroller are performed. In addition, we evaluate implementations using AES co-processors on a smartcard chip and a general-purpose microcontroller by means of laser fault injection and clock tampering, respectively. In all three settings, 4 bytes of the last round key of AES could be successfully recovered with 30, 16, and 1 200 faulty ciphertexts, respectively. In all practical scenarios, the attack has to be repeated three more times to recover the full last round key (in case of AES-128).

Outline. The remainder of the paper is organized as follows. In Sect. 2, we give some background on fault attacks in general, recapitulate the work of Fuhr et al. [18] on SFA, and introduce our attack model. In Sect. 3, we show how

SFA can be applied to various AES-based authenticated encryption schemes. Finally, we present practical experiments and verify the practicality of SFA on three different hardware platforms in Sect. 4.

2 Background

In this section, we revisit the Statistical Fault Attacks on AES underlying our attacks. We start with a general overview of different types of fault attacks, and briefly describe the biased fault model in the attack of Fuhr et al. [18]. Finally, we discuss the modified, much more general biased fault model we use in this paper, and how to identify the best key candidates.

2.1 Fault Attacks

The fault attacks of Boneh et al. [13] demonstrate the vulnerability of unprotected implementations of asymmetric cryptographic primitives like RSA. Later, also attacks on Elliptic Curve Cryptography [10] have been shown. Those attacks point out that faults induced during the execution of an cryptographic primitive can be used to extract the secret of a device. In this work, we will use either clock glitches, or a laser to induce the faults, however, numerous other ways exist to perform fault attacks like electro-magnetic pulses, or variation in the supply voltage [7, 31, 45].

Biham and Shamir [11] introduced several fault attacks on symmetric cryptographic primitives, amongst others Differential Fault Analysis (DFA). DFA works by collecting pairs of faulty and fault-free ciphertexts, where the fault has been induced in the last few rounds of the computation and an example of a DFA has been shown on DES [11]. Here, knowledge about the difference induced combined with the knowledge about the differences of the cipheretexts can be used to retrieve information on the secret key. DFA is not limited to DES or Feistel Structures and can be applied to other schemes like AES [37].

In contrast to DFA, Collision Fault Analysis (CFA) [12] exploits faults induced in the first rounds of a cryptographic primitive. For CFA, an attacker encrypts related plaintexts and uses a fault in an attempt to cancel the differences caused by the plaintexts. In this attack, the knowledge of a successful collision can then be used to get information about the secret key. In a Safe Error Attack (SEA) [49], the knowledge whether a fault has an effect on the outcome of a computation or not is exploited.

As we have seen, most fault attacks require the ability of an attacker to perform an encryption twice with the same inputs, or even to be able to choose the relation of inputs. Conditions that are usually hard to fulfill in nonce-based authenticated encryption schemes. However, this is not the case for the Statistical Fault Attack (SFA) [18] that works with random unknown plaintexts. A short introduction to SFA is provided in the next section.

2.2 Statistical Fault Attacks

In 2013, Fuhr et al. proposed a new type of fault attack, called Statistical Fault Attack (SFA) [18]. In contrast to most previous attacks, the adversary only requires a collection of faulty ciphertexts encrypted with the same key. Hence, SFA works with random and unknown plaintexts.

Fault Model. Unlike most traditional fault attacks, SFA requires a slightly different fault model. Assuming that intermediate variables get uniformly distributed towards the last rounds for secure cryptographic primitives like AES, an attacker has to be able to induce faults which change the distribution of some intermediate values to be non-uniform. In particular, Fuhr et al. considered the following three fault models:

(a) the stuck-at-0 fault model with probability 1,
(b) the stuck-at-0 fault model with probability 1/2,
(c) the stuck-at model to an unknown and random value e with probability 1.

Using these non-uniform fault models, Fuhr et al. were able to show several attacks on AES based on simulations. Their attacks target the last 4 rounds with a small number of faulty ciphertexts and practical complexity.

Description of the AES. AES is a byte-oriented block cipher following the wide-trail design strategy. It operates on a state of 4×4 bytes and updates it in 10, 12, or 14 rounds, depending on the key size of 128, 192, or 256 bits. In each round (except the last one with no MixColumns), the following four transformations are applied.

SubBytes (SB): This step is the only non-linear transformation of the cipher. It is a permutation consisting of an S-box S applied to each byte of the state.
ShiftRows (SR): This step is a byte transposition that cyclically shifts each row of the state by different offsets. Row j is shifted right by j byte positions.
MixColumns (MC): This step is a permutation operating on the state column by column. To be more precise, it is a left-multiplication by a 4×4 circular MDS matrix M over $\mathbb{F}_{2^8}$.
AddRoundKey (AK): In this transformation, the state is modified by combining it with a round key with a bitwise `xor` operation.

Attack Procedure and Complexity. While Fuhr et al. proposed several attack variants, we will focus only on the attack that targets the 9^{th} round of AES. When changing the distribution of one byte of AES before the last MixColumns, they showed that with these fault models, 4 bytes of the last round key could be recovered with high probability using the Squared Euclidean Imbalance (SEI) distinguisher with only 6, 14, and 80 faulty ciphertexts, respectively. We briefly recount the attack below, but refer to [18] for a more detailed description.

If we denote our target state before the last MixColumns in the encryption to the i^{th} ciphertext by $\tilde{S}_9^i$, we can express one byte of this state as a function of the ciphertext $\tilde{C}^i$, 4 bytes of the last round key K_{10}, and one byte of $\mathsf{MC}^{-1}(K_9)$, as follows. Our target state is

$$\begin{aligned}\tilde{S}_9^i &= \mathsf{MC}^{-1}(\mathsf{SB}^{-1} \circ \mathsf{SR}^{-1}(\tilde{C}^i \oplus K_{10}) \oplus K_9) \\ &= \mathsf{MC}^{-1}(\mathsf{SB}^{-1} \circ \mathsf{SR}^{-1}(\tilde{C}^i \oplus K_{10})) \oplus \mathsf{MC}^{-1}(K_9).\end{aligned}$$

Each byte of $\tilde{S}_9^i$ can therefore be deduced using one hypothesis on 4 bytes of K_{10} and on one particular byte of $\mathsf{MC}^{-1}(K_9)$. As shown by Fuhr et al., the xor with $\mathsf{MC}^{-1}(K_9)$ does not modify the distance of the biased distribution from uniform. Hence, it can be omitted in the attack. In other words, this allows to mount the attack on a modified $\tilde{S}_9^{i\prime}$:

$$\tilde{S}_9^{i\prime} = \mathsf{MC}^{-1} \circ \mathsf{SB}^{-1} \circ \mathsf{SR}^{-1}(\tilde{C}^i \oplus K_{10}).$$

This allows us to recover 4 bytes of the last round key K_{10} by making 2^{32} hypotheses on their value and predicting one byte of $\tilde{S}_9^{i\prime}$. By repeating the attack 4 times, one can recover the complete last round key K_{10}.

2.3 A Generalized Fault Model

In this work, we want to go beyond specific fault models like in Sect. 2.2. The only assumption we make is that the attacker is able to influence some byte (or a larger structure) of the internal state of AES before the last MixColumns such that this value becomes clearly non-uniformly distributed. We make no assumptions about the details of this non-uniformity, nor do we require that the attacker knows the new distribution. To exploit this type of fault, the attacker will collect faulty (biased) ciphertexts, compute backwards to the target byte for different key guesses, and try to reject wrong key guesses that would result in an approximately uniform measured distribution of the biased target byte. In the remainder of this section, we discuss how to identify the non-uniform distribution for the wrong key guesses.

We do not consider the distribution on bit-level, but for example on byte-level. Exploiting such non-uniform distributions of multi-bit values (more specifically, distributions of several sums of single bits) has already been investigated in the context of multidimensional linear cryptanalysis [21]. However, the distributions in this context are typically very close to uniform, unlike the distributions we expect in the case of SFA. Unfortunately, as noted by Samajder and Sarkar [44], the state-of-the-art framework for multidimensional linear cryptanalysis is not suitable for handling distributions which are significantly different from uniform. On the positive side, testing the closeness of discrete distributions [41] is a well-established field of research. Here, the central challenge is to determine whether two discrete distributions are the same (or close to each other) with the help of as few samples as possible. In our case, we want to determine whether our given samples are distributed uniformly or not.

The algorithms needing the fewest samples to perform this task are based on an idea of Goldreich and Ron [19]. Their algorithm makes use of collisions between sampled values to test for uniformity, since the expected number of collisions is lowest for uniformly distributed samples. Hence, the further a distribution deviates from the uniform distribution, the more collisions and multi-collisions we expect.

Of course, it is possible to directly base the testing of the key hypothesis on uniformity testing. For instance, Batu et al. [8] present a test which requires $O\left(\epsilon^{-4} \cdot \sqrt{2^s} \cdot \log(1/\gamma)\right)$ samples for distributions over 2^s-element sets. Their test accepts with probability $1-\gamma$ if the samples come from a distribution with ℓ_1-norm distance smaller than $\epsilon/\sqrt{3 \cdot 2^s}$ to the uniform distribution. It rejects with probability $1-\gamma$ if the samples come from a distribution which is more than ϵ away from the uniform distribution.

However, for our use-case, an approach that ranks keys according to some metric, like the number of collisions, is more suitable than a binary decision whether the measured distribution is uniform or not. Significantly more samples are needed to clearly separate the distribution for the right key hypothesis from the wrong ones to enforce a binary decision, whereas for the ranking, it is usually sufficient if the right key is ranked somewhere among the top candidates. Since the uniformity tests of Batu et al. [8] and Paninski [36] are actually based on counting collisions, they also provide us with a starting point for a ranking algorithm. This algorithm ranks the key hypothesis according to the number of collisions, and gives multi-collisions a higher weight. In our experiments, this ranking algorithm performs as good as ranking based on the SEI.

Interestingly, the key ranking mechanism based on the SEI used in [18, 38] can also be linked to counting collisions. Let s be the bitsize of our biased intermediate value $S_i = f^{-1}(\hat{K}, \tilde{C}_i)$, computed from the faulty ciphertext $\tilde{C}_i$ under the key hypothesis $\hat{K}$. Assuming that we have N faulty ciphertexts, the SEI d is calculated as

$$d(\hat{K}) = \sum_{\delta=0}^{2^s-1} \left(\frac{\#\{i \mid f^{-1}(\hat{K}, \tilde{C}_i) = \delta\}}{N} - \frac{1}{2^s} \right)^2 .$$

This distinguisher assigns high values to key hypotheses $\hat{K}$ that lead to distributions of intermediate values S_i with many collisions. For instance, consider a sample size of $N = 2^s$ samples. Then, the SEI is essentially counting collisions, since only events that occur exactly once do not increase d. Moreover, since the deviation from uniform is squared, a greater deviation, or in our sense a multi-collision, contributes more to d.

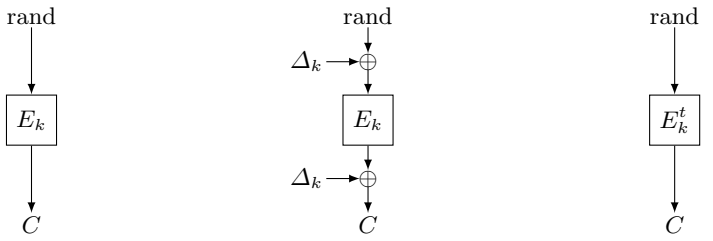
To sum up, it turned out that the SEI cannot be outperformed in practice by a new ranking algorithm based on counting collisions, since the SEI is actually doing that. Hence, we decided to stick to the more common SEI to measure if the distribution of one byte value becomes clearly non-uniformly distributed. So for AES, the 4-byte key guesses of the last round key are ranked according to the resulting SEI of one byte before the last MixColums when decrypting faulty ciphertexts for one round. To be able to observe non-uniformness and to evaluate

the SEI, we require the input to the block cipher to be different for each fault and the block cipher output to be known.

3 Statistical Fault Attacks on Authenticated Encryption

In this section, we evaluate the applicability of the Statistical Fault Attack to several authenticated encryption modes for AES. This includes the widely-used ISO/IEC-standardized modes like CCM [48], EAX [9], GCM [32] and OCB [40], as well as new authenticated encryption modes proposed in the CAESAR initiative [46]. For evaluating the applicability of the fault attacks to these authenticated encryption schemes, we only need very limited assumptions. As already stated in Sect. 2, we assume that the attacker is able to influence some byte (or a larger structure) of the internal state of AES before the last MixColumns operation in a way that this value becomes clearly non-uniformly distributed.

We classify the investigated authenticated encryption modes into three categories, as illustrated in Fig. 1:



(a) Basic Construction **(b)** XEX-like Construction **(c)** Tweakable Block Cipher

Fig. 1. Classification of AES-based authenticated encryption schemes.

Basic Construction. The schemes in this category allow to directly observe the output of the block cipher. This includes schemes based on classical encryption schemes such as CTR [15], CBC [17], CFB [17], etc., but also schemes based on the XE construction [39], which masks the input of the block cipher using secret masks Δ_k. More generally, we assume that the input to the block cipher is a secret random value, but the output is observable to the attacker.

XEX-like Construction. This construction is similar to XE, but unlike XE, both the input and the output of the block cipher are masked using secret, nonce-dependent masks Δ_k. Constructions following the XEX construction [39] include for instance IAPM [28], OCB [40], and several of the CAESAR candidates.

Tweakable Block Cipher. The third category covers schemes that use a dedicated tweakable block cipher, which depends on a (typically nonce-dependent) tweak in addition to the secret key. Since the focus of this work is on

AES-based modes, we will restrict ourselves to constructions using the AES round function and following the TWEAKEY framework [27], such as for instance the CAESAR candidates KIASU [26] and Deoxys [24].

In the remainder of this section, we will discuss the applicability of Statistical Fault Attacks to schemes of these three categories in turn.

3.1 Application to the Basic Construction

In this construction, the output of the block cipher is directly known to the attacker, or can trivially be recovered by, say, xoring observable values with public values or constants. It is easy to see that in this case, the Statistical Fault Attack described in Sect. 2 can be applied in a straight-forward way to recover the secret key k. As an example, we discuss the application of Statistical Fault Attacks on AES in counter (CTR) modes as used in GCM, CCM and EAX (all standardized by ISO/IEC).

Statistical Fault Attack on CCM, EAX and GCM. As a representative example for the three modes, we will discuss the attack on CCM, which is shown in Fig. 2. As its name implies, the CTR-with-CBC-MAC mode (CCM) can be split into an encryption part using AES in counter mode to encrypt the plaintext P and an authentication part using CBC-MAC to authenticate the nonce N, associated data A, and plaintext P, which generates the tag T. For clarity, we have substituted the first part of the CBC-MAC, where the associated data is processed, with its outcome V in Fig. 2. Since the fault attack is solely performed on the encryption part, the following observations also hold for EAX and GCM that both use AES in CTR mode for encryption.

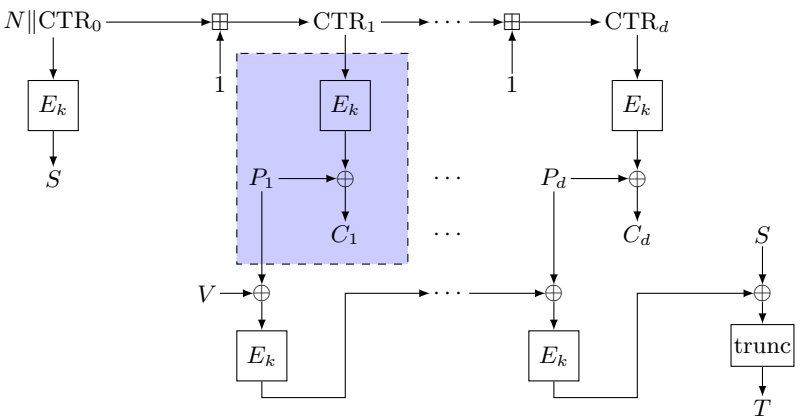


Fig. 2. The counter with CBC-MAC mode.

For the sake of simplicity, we restrict our fault attack to the encryption E_k of the first plaintext block (marked by the dashed rectangle in Fig. 2). Let us recall the conditions of Sect. 2 that are necessary for the Statistical Fault Attack to work:

1. The inputs of the block cipher need to be different for each fault.
2. The block cipher output needs to be known.

Condition 1 is always fulfilled, since it is required that the nonce N changes for each encryption and thus, the input to E_k changes as well. Condition 2 is fulfilled assuming a known plaintext attack, where the plaintext block P_1 is known to the attacker. Then, one can compute the keystream part for encrypting this plaintext block by xoring it with C_1. The resulting keystream is the output of the block cipher E_k. To sum up, we are able to observe outputs of the block cipher E_k for various inputs. Thus, we have the same preconditions as for the fault attack on plain AES described in Sect. 2. Hence, the attack can be applied to CCM (and any other scheme based on CTR mode) in a straight-forward way. We want to stress that the attacker does not require to know the input of the block cipher, it is just necessary that it changes. Therefore, the attack also applies to modes where the value of the counter is unknown, such as EAX.

Statistical Fault Attack on OCB. Although ISO/IEC-standard OCB is based on the XEX construction, we show that it is also vulnerable to the attack on the basic construction. The reason for this is that if the last plaintext block is incomplete, it is instead processed using the XE construction, as shown in Fig. 3. Therefore, the knowledge of this incomplete last plaintext and ciphertext block allows an attacker to compute the output of the block cipher E_k and thus, the Statistical Fault Attack is again applicable.

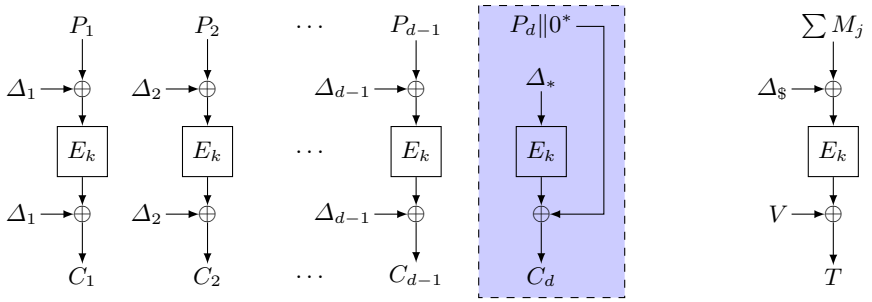


Fig. 3. Encryption in OCB.

Application to Other Modes. Besides CCM, EAX, GCM, and OCB, the fault attack discussed in this section also applies to several other authenticated encryption modes. For instance, to the CAESAR candidates Cloc [22] and Silc [23],

which are based on cipher-feed-back mode (CFB), where the ciphertext is the xor of the output of a block cipher E_k and the plaintext blocks. Another example is AES-OTR [34], which uses a balanced two-round Feistel network for encryption. The round function of this network is AES in an XE mode. Since the balanced Feistel network has only two rounds, knowledge of the plaintext and ciphertext implies knowledge of the block cipher output. Thus, again, the Statistical Fault Attack is directly applicable.

3.2 Application to XEX-like Constructions

In this construction, the output of the block cipher is masked with a secret value Δ_k, which prevents a straightforward application of the basic attack. However, depending on how Δ_k is computed, the Statistical Fault Attack may nevertheless be applicable. In the simplest case, Δ_k is not nonce-dependent. This allows to repeatedly observe ciphertexts masked with a secret, but constant value Δ_k. We demonstrate how to exploit this in an attack on the CAESAR candidate AES-COPA [4].

Statistical Fault Attack on AES-COPA. AES-COPA uses an XEX-like construction for encrypting the plaintext, which is shown in Fig. 4. The input V of the plaintext processing is the result of a PMAC-like processing of the associated data A and the nonce N. Thus, V will change for different nonce values. Each processed ciphertext block requires two invocations of the block cipher E_k. AES-COPA masks both the input of the block cipher processing the plaintext blocks P_j, and the output of the block cipher that generate ciphertext blocks C_j. The masks are based on a secret value $L = E_k(0)$. We focus our attack on the block cipher call that generates C_1, as marked in Fig. 4.

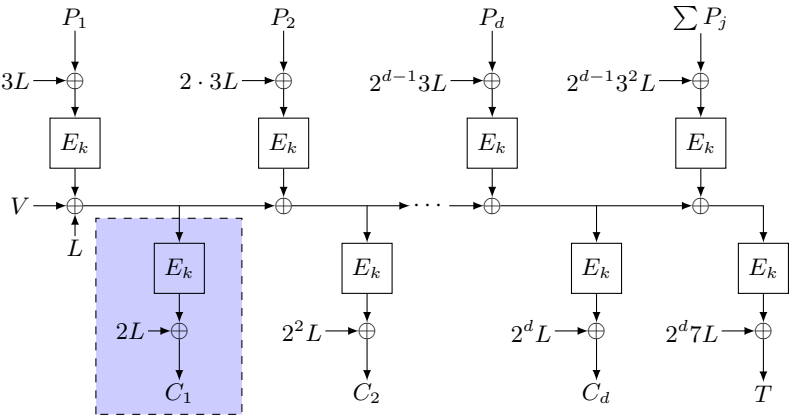


Fig. 4. Plaintext processing of AES-COPA, $L = E_k(0)$.

So far, only one of our two prerequisites for the SFA from Sect. 2 is fulfilled. We can vary the input of the block cipher calls by changing, for example, the nonce, associated data, or plaintext. However, the output of the block cipher is unknown, since it is masked with the secret value $\Delta_k = 2 \cdot E_k(0)$ to get C_1. To overcome this obstacle and since Δ_k solely depends on the secret key k, we consider Δ_k as a part of the key schedule to compute the last round key. Thus, instead of the last round key K_{10} of AES, we get $K'_{10} := K_{10} \oplus (2 \cdot E_k(0))$ as the last round key.

Hence, instead of recovering the last round key K_{10} of AES as in the attacks before, we now can recover K'_{10} by using SFA as described in Sect. 2. For recovering K'_{10}, the complexity and the needed numbers of faults are the same as for the attack on AES itself. However, the knowledge of K'_{10} does not directly lead to a key recovery attack of the master key k. Therefore, we need to perform the Statistical Fault Attack a second time. One option is to target again the first plaintext block and use our knowledge of K'_{10} to now target the AES round key K_9. Alternatively, we repeat the attack for the second plaintext block to recover $K_{10} \oplus (4 \cdot E_k(0))$ and thus get K_{10} by solving the resulting linear system. In both cases, the master key can then easily be recovered from K_9 and K_{10}, respectively.

Application to Other Modes. Besides COPA, other schemes that use a nonce-independent Δ_k and allow the Statistical Fault Attack include ELmD [14] and Shell [47]. In contrast, some schemes, such as IAPM, OCB, or some CAESAR candidates, also include the nonce in the computation of Δ_k. All these schemes have in common that Δ_k changes unpredictably for each block cipher call, which prevents a straight-forward application of Statistical Fault Attacks.

Instead of relying on misuse settings like repeated nonces, we will have a closer look at how these schemes typically compute Δ_k. In many cases, Δ_k can be decomposed into two values: a known, nonce-dependent part δ_N, and a secret, key-dependent part δ_k, which are then for example combined with a linear function to produce Δ_k. In this case, we can adapt our attack as follows, similar to the COPA case. First, we recover the modified last round key $K'_{10} = K_{10} \oplus \delta_k$. Depending on the key schedule and the function δ_k, this may already be sufficient to recover the master key (e.g., if δ_k and the key schedule are linear). Otherwise, we repeat the attack a second time to the round before to recover K_9 as described before.

3.3 Application to Modes Based on Tweakable Block Ciphers

In this construction, the authenticated encryption scheme uses a tweakable block cipher E_k^t instead of a regular block cipher as basic building block. In this case, the Statistical Fault Attack is not generally applicable. However, for some tweakable block ciphers such as the ones presented within the TWEAKEY framework [27], we can adapt our attack. In particular, this is possible if the last subkeys of the tweakable block cipher can be described by the composition of

two values, $\delta_t \oplus \delta_k$. We illustrate the working principle of the attack for the CAESAR candidate Deoxys [24], but the same attack is also applicable to KIASU [26], where the tweak t is only xored to each round-key.

Statistical Fault Attack on Deoxys. Deoxys offers two modes of operation, both using two variants of the underlying tweakable block cipher Deoxys-BC. We focus on Deoxys$^{\neq}$-128-128, which uses Deoxys-BC-256 as underlying tweakable block cipher. As shown in Fig. 5, Deoxys$^{\neq}$ encrypts the individual plaintext blocks P_j in an ΘCB3-like [30] way. This ensures both the variation of the tweakable block cipher inputs, and knowledge of the outputs. However, since the tweak is partly defined by the nonce, we have to determine the influence of this nonce on the last round key that we want to recover using SFA. Thus, we have to have a closer look at the definition of the tweakable block cipher Deoxys-BC-256.

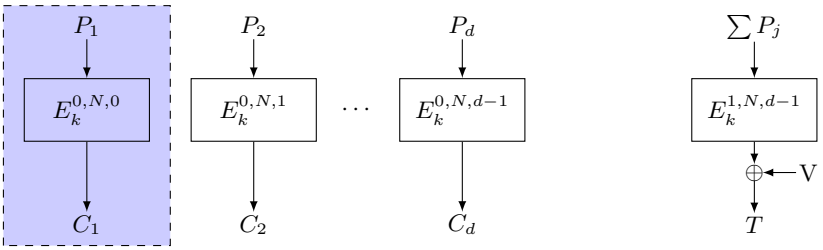


Fig. 5. Plaintext processing for Deoxys$^{\neq}$.

Fig. 6 shows how Deoxys-BC-256 uses the round function f of the AES, but computes different round keys K_i based on the master key k and tweak t. Here, K_i is the xor sum of three values: a key-dependent round key K_i^k, a tweak-dependent round tweak K_i^t, and a round constant c_i. The values are updated using a simple byte permutation h. For instance, $K_0^k = k$, $K_0^t = t$, $K_1^k = 2\,h(k)$, $K_1^t = h(t)$, $K_r^k = 2\,h(2\,h(\ldots 2\,h(k)\ldots))$, and $K_r^t = h(h(\ldots h(t)\ldots))$.

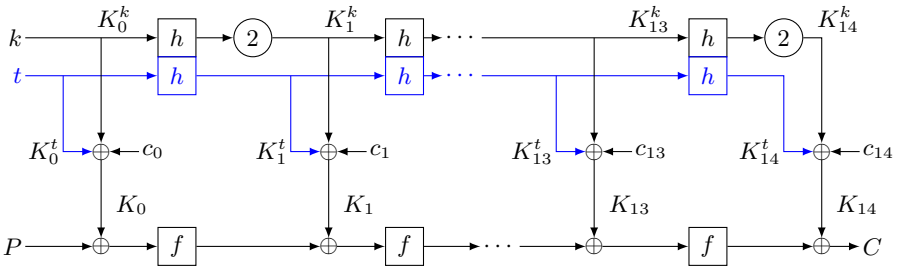


Fig. 6. Block cipher Deoxys-BC-256.

Since the value of the tweak used for encryption is publicly known, the varying part K_i^t of the round keys K_i can be easily calculated. The unknown parts K_i^k of the round key are constant for multiple calls of the block cipher under the same key k. Hence, the last round key K_{14}^k can be recovered with the SFA on AES described in Sect. 2.

3.4 Summary and Discussion of Results

We demonstrated in the previous sections that several authenticated encryption modes for AES are susceptible to Statistical Fault Attacks. A summary of the results is given in Table 1. However, Statistical Fault Attacks are applicable to a broader range of authenticated encryption schemes, and are not limited to AES-based modes. Natural targets for the attack include, for instance, the CAESAR candidates Joltik [25] and Scream [20], which also follow the TWEAKEY framework [27], or Prøst [29], which applies the modes of COPA [5] and OTR [35] to an Even-Mansour block cipher.

Moreover, the attack is not limited to block cipher based constructions. For instance, the APE construction [3] uses a secret key in the finalization for tag generation, making it a natural target for the attack. Also the sponge-based CAESAR candidates Ascon [16] and PRIMATEs [2] both employ a keyed finalization, with similar effects. However, the fact that large parts of the internal state are truncated to generate the authentication tag might complicate the attack.

Table 1. Statistical fault attacks on AES-based authenticated encryption modes in the nonce-respecting setting.

Primitive	Classification	Comments	Reference
CCM	basic	CTR	3.1
GCM	basic	CTR	3.1
EAX	basic	CTR	3.1
OCB	basic	XE (incomplete blocks)	3.1
Cloc/Silc*	basic	CFB	3.1
OTR*	basic	XE	3.1
COPA*	XEX		3.2
ELmD*	XEX		3.2
SHELL*	XEX		3.2
KIASU*	TBC	TWEAKEY	3.3
Deoxys*	TBC	TWEAKEY	3.3

* CAESAR candidates

4 Practical Verification/Implementation of the Attacks

In order to demonstrate the practical relevance of Statistical Fault Attacks and to validate the assumptions from previous sections, we performed three fault-injection experiments targeting real hardware.

An AES-GCM implementation executed on an off-the-shelf microcontroller served as target for the first experiment. In this context we used the ASM AES version from [1] to realize the block cipher. Due to the lack of embedded platforms implementing GCM or CCM completely in hardware, we put the focus of the following analysis on hardware AES co-processors available on a smartcard microcontroller and on a general-purpose microcontroller, respectively. The remaining parts for realizing the authenticated encryption modes are then implemented in software.

In all settings, the fault injections aim to induce a bias on at least one byte of the AES state before the last MixColumns transformation, and allow to reveal 32 bits of the last AES round key. For full key recovery, the attack has to be repeated three more times. The following list provides an overview of the fault-injection methods and the attack results for the three settings:

1. Clock tampering has been used to disturb the execution of the AES software implementation running on an ATxmega 256A3 general-purpose microcontroller. This setting allowed to reveal 4 bytes of the last round key with less than 30 faulted ciphertexts.
2. Laser fault injections on an AES co-processor on a smartcard microcontroller. Our experiments show that less than 16 faulty ciphertexts are sufficient to reveal 4 bytes of the last round key.
3. Clock tampering on a hardware AES co-processor implemented on a general-purpose microcontroller. In this setting, we need approximately 1 200 faulted ciphertexts for recovering 4 bytes of the last round key.

For all attacks, 4 bytes of the last round key can be recovered out of the faulted ciphertexts in less than one hour using an Intel Core i7 3770K. In the following, we give a detailed description and summary of the practical fault-injection attacks. For the extended version of the paper, we have performed additional experiments targeting the AES co-processor of an 8-bit microcontroller using a single clock glitch (Fig. 12) and 20 consecutive clock glitches (Fig. 13) which are given in Appendix B.

4.1 AES Software Implementation on an 8-bit Microcontroller

In the following setting, we used clock glitches to provoke faults during an AES computation implemented in software on an 8-bit microcontroller. In particular, we used the ASM AES version from [1] for realizing the GCM AE mode.

For the clock-glitch experiments, a nominal clock frequency of 24 MHz ($T_{\text{clk}} = 41.7\,\text{ns}$) was used. According to [1], one 128-bit encryption requires 2 555 clock cycles. For simplicity, we used one general-purpose I/O pin of the microcontroller

for indicating the start of the AES encryption. This trigger pin together with the knowledge of the length of the AES encryption procedure allows to find the correct time interval for inserting the clock glitch. Next to that, our results show that faults in consecutive clock cycles also lead to successful key recovery. As a consequence, this behavior allows to relax the precision prerequisite of the trigger information.

With the found parameters, we collected two sets, each containing 80 faulty ciphertexts. For the first set, a single clock glitch was inserted. For the second set, clock glitches in 50 consecutive clock cycles were inserted. Next, we performed SFA attacks using an increasing number of faulty ciphertexts on both sets individually. The results containing the set size N, the SEI value for the correct subkey (SEI_c), and the maximum SEI value of the wrong subkey guesses (SEI_w) were stored in two separate lists (one list for each set) in the format $[N, \mathrm{SEI}_c, \max(\mathrm{SEI}_w)]$. For this attack scenario, we started with $N = 4$ and increased N in every iteration by 4.

Fig. 7 displays the evolution of the SEI values for increasing number of ciphertexts in the single clock glitch setting. Values corresponding to the correct subkey are plotted in red, the maximum SEI values of the wrong subkey guesses are plotted in blue. With 30 faulty ciphertexts, SEI_c exceeds $\max(\mathrm{SEI}_w)$, which allows to reveal the correct subkey value.

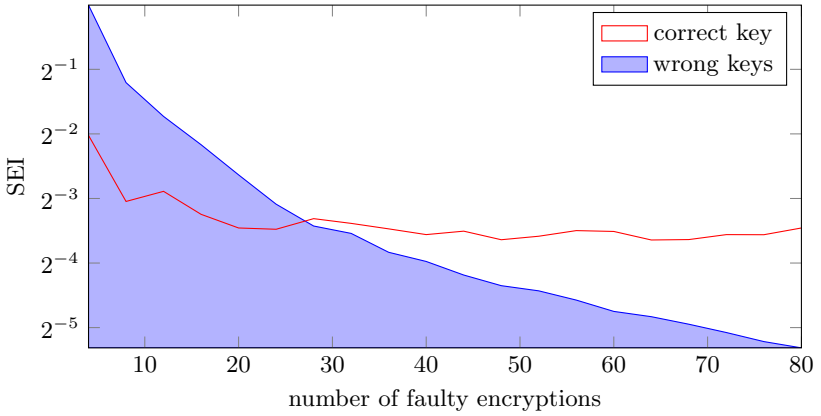


Fig. 7. SEI values for correct key (SEI_c) plotted against best SEI for a wrong key ($\max(\mathrm{SEI}_w)$) for increasing number of faulty encryptions. Setup: AES software implementation, single clock glitch.

Fig. 8 displays the evolution of the SEI values for an increasing number of ciphertexts for the setting with 50 consecutive clock glitches. In this setting, 24 ciphertexts are sufficient for SEI_c to exceed $\max(\mathrm{SEI}_w)$, which allows to reveal the correct subkey value.

Results of the fault attacks targeting the AES software implementations using clock glitches show that with 30 faulty ciphertexts, it is possible to reveal the 32-bit subkey if a single clock glitch is inserted. Furthermore, if the clock glitch is inserted in 50 consecutive clock cycles, approximately 25 faulty ciphertexts are sufficient for subkey recovery. We did not further investigate the approach of inserting the clock glitch in consecutive clock cycles because this is out of scope of the current work. Nevertheless, by carefully trimming the fault injection parameters, the number of faulty ciphertexts for successful subkey recovery could probably be further decreased.

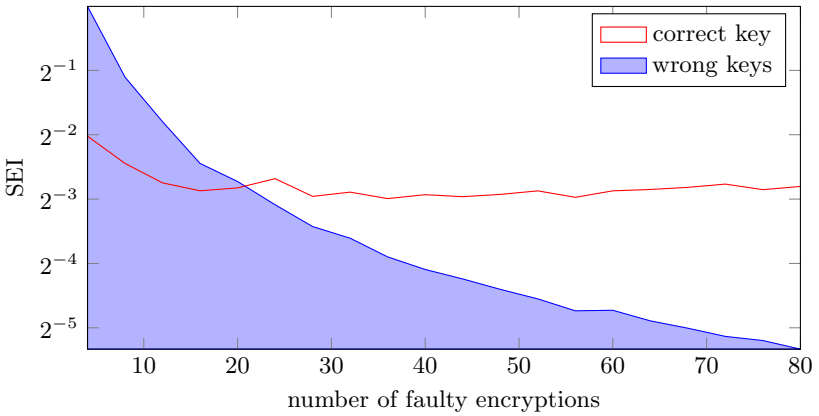


Fig. 8. Evolution of the SEI values with increasing number of faulty encryptions. Setup: AES software implementation, multiple clock glitches.

4.2 AES Hardware Co-Processor of a Smartcard Microcontroller

In this experiment, we used a laser fault injection system to induce faults during encryptions of an AES Hardware co-processor of a smartcard microcontroller. This co-processor can easily be used as building block for realizing authenticated encryption modes like GCM or CCM on the smartcard.

The laser fault injection system consists of an infrared laser diode module and a microscope allowing to focus the laser spot depending on the microscope objective used. Here an objective with a 10× magnification is used. The whole system is mounted on a motorized X-Y-Z stage.

As the smartcard microcontroller runs its own operating system, the only signal available for triggering the laser injection system is the sending of the encryption command through APDU command. Therefore, a temporal delay is added to postpone the laser injection during the AES encryption thanks to a remotely controllable pulse generator. Furthermore, as the smartcard microcontroller runs on its own internal clock network, an inherent temporal jitter

is present due to the asynchronism between the laser injection system and the smartcard microcontroller clock network. These experimental conditions are very close to the ones present in real world scenarios.

By applying a spatial fault injection cartography, we have been able to find a spatial position where only one byte of the AES state is faulted. Furthermore, by trying different delays, we found a spatio-temporal setting where only 4 bytes of the ciphertext were faulted with a high reliability. By studying the indices of the faulted ciphertext bytes, we concluded that we successfully induced a fault on one byte of the AES state just before the last MixColumns. The fact that the hardware AES module can also be used outside of the context of authenticated encryption, i.e., for encrypting single plaintext blocks, simplified this profiling. However, if the stand-alone usage of the AES co-processor is not possible on the attacked platform, the search for the right fault injection parameters becomes more complicated, but is still feasible.

With the found parameters, we collected again 80 faulty ciphertexts. With the collected faulty ciphertexts, the same evaluation as in the previous section was conducted. We started again with an initial attack set size $N = 4$ and increased the size of the attack set by 4 in every iteration. The evolution of the SEI values with increasing set size is depicted in Fig. 9. Values corresponding to the correct subkey are plotted in red, the maximum SEI values of the wrong subkey guesses are plotted in blue.

As depicted on Fig. 9, SEI_c already exceeds $\max(\mathrm{SEI}_w)$ with only $N = 16$ ciphertexts. Therefore, this number of ciphertexts allows to retrieve 4 bytes of the correct last round key. This result validates the practicability of the fault model and even shows that laser-based fault injection systems are well suitable for this kind of attacks.

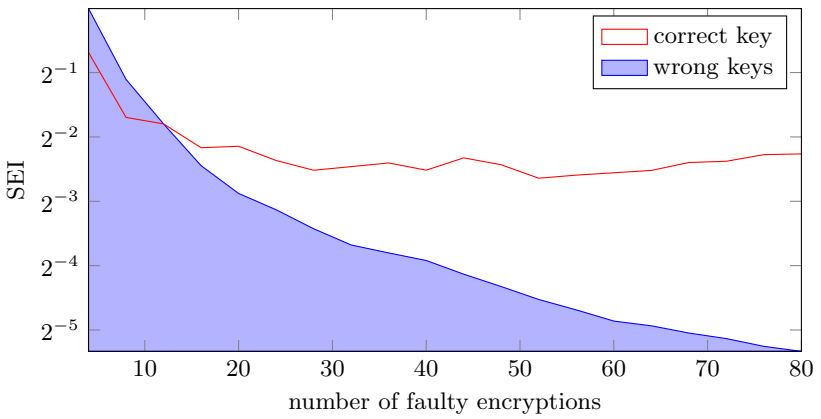


Fig. 9. Evolution of the SEI values with increasing number of faulty encryptions. Setup: AES hardware co-processor of a smartcard microcontroller, laser.

4.3 AES Co-Processor on a General-Purpose Microcontroller

In this setting, we use clock glitches to inject faults during the encryption procedure of an AES co-processor integrated on a general-purpose microcontroller. This co-processor can on the one hand be used as stand-alone block cipher to encrypt plaintext blocks, on the other hand it can be used in the context of AE for realizing a mode of operation like GCM or CCM. The co-processor in stand-alone mode allows profiling the hardware in order to find suitable fault-injection parameters. The target of the fault injection is the output of the byte substitution (SubBytes) in the 9^{th} AES round. The AES co-processor implements the SubBytes function with pure combinational logic. Since one column of the state is processed in a single clock cycle, this allows to create faults in 4 bytes of the state with a single clock glitch.

We define with T_{glitch} the time interval between two subsequent positive clock edges in case of a clock glitch. This value is smaller compared to the nominal clock period T_{clk}, as illustrated in Fig. 10. If T_{glitch} is smaller than the path delay of the combinational SubBytes block, the output value of this block has not settled to its correct, stable value. As a result, a wrong value is sampled by the registers at the output of the block, which leads to faults in the ciphertext.

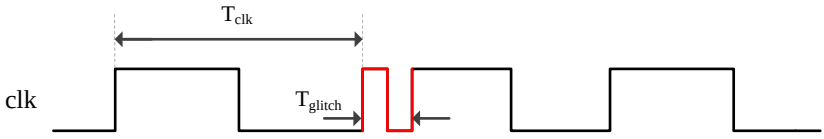


Fig. 10. Clock signal with intentionally inserted additional positive clock edge.

For the clock glitch experiments, we used a nominal clock frequency of 10 MHz ($T_{\text{clk}} = 100$ ns). Preliminary fault experiments allowed to find the correct clock cycle (i.e., the delay between the start of the encryption and the targeted instruction) to disturb the SubBytes operation in the 9^{th} round before the MixColumns step. With $T_{\text{glitch}} = 10.2$ ns, we achieved a fault probability of 99.5 %.

With these parameters, we executed the AES encryption to receive 2 000 faulty ciphertexts. The increased number of ciphertexts was required because preliminary experiments revealed that the bias introduced with the clock glitch was significantly smaller compared to the bias introduced by the laser attack. With the collected faulty ciphertexts, the same evaluation as in the previous section was conducted. Due to a smaller bias, we started with an initial attack set size $N = 32$ and increased the size of the attack set by 32 in every iteration. The evolution of the SEI values with increasing set size is depicted in Fig. 11. Values corresponding to the correct subkey are again plotted in red, the maximum SEI values of the wrong subkey guesses are plotted in blue.

As depicted on Fig. 11, starting at 1 200 ciphertexts, SEI_c exceeds $\max(\text{SEI}_w)$. This allows to reveal the correct subkey in an attack setting. Compared to the

results presented in the previous section, the number of required ciphertexts is nearly 100 times higher, but the number is still practical and this amount of ciphertexts can be collected within minutes. However, the effort for performing clock-glitch attacks compared to laser fault attacks (e.g., preparing the fault-injection environment, finding good fault-injection parameters) is significantly smaller, which has to be taken into account.

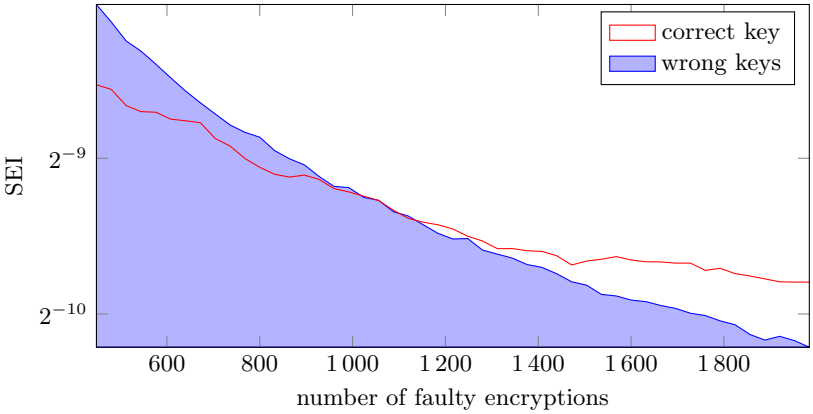


Fig. 11. Evolution of the SEI values with increasing number of faulty encryptions. Setup: AES co-processor on a general-purpose microcontroller, clock glitch.

4.4 Discussion and Remarks

The goal of the attacks presented in this section is a feasibility study proving that the assumed biased fault model is indeed valid on different platforms using different fault-injection mechanisms.

For the software implementation, a general-purpose I/O pin indicating the start of the AES encryption has been used, which allowed a precise fault injection using clock glitches. Real-world scenarios, like the second experiment targeting the smartcard microcontroller, typically do not allow the usage of a trigger pin. In such scenarios, other sources for synchronizing the fault-injection procedure can be applied, like spying the communication or the power profile. This can decrease the precision of the fault injections.

But it is important to note that the outcome of the SFA attack does not strictly rely on a precise fault injection. If only a subset of the received ciphertexts are affected by the expected fault pattern, the remaining ciphertexts (fault-free or fault hitting another location during the cipher rounds) are treated as noise. A more reliable fault injection process however minimizes the number of required ciphertexts for successful key recovery.

Furthermore, when the attacked platform allows the usage of the AES co-processor for stand-alone encryption (e.g., as in the previous experiments), one

can easily perform a profiling step which simplifies the search for appropriate fault injection parameters. Nevertheless, if the AES co-processor can only be used in the context of the authenticated encryption mode, it is still possible to find the appropriate fault injection parameters. Of course, the number of attempts and the search space for the parameters increase, resulting in a more time-consuming setup phase for the fault injection.

With the practical results presented in this section, we showed that implementations of AES-based authenticated encryption modes on different hardware platforms are vulnerable to the proposed fault attacks introduced in this work.

5 Conclusion

In this work, we demonstrate for the first time that a wide range of nonce-based authenticated encryption schemes, including the widely used ISO/IEC standards CCM, GCM, EAX, and OCB, are susceptible to fault attacks. All our attacks need only very limited assumptions about the attacker's capabilities. To confirm these assumptions and to show the practical relevance of the attacks, we perform several fault-injection experiments targeting real hardware. This highlights the need for dedicated fault attack countermeasures for authenticated encryption schemes. Although our analysis focus only on AES-based constructions, we want to note that it is applicable to a broader range of authenticated encryption schemes. This is part of future work.

Acknowledgments

The authors would like to thank the organizers and participants of ASK 2015 that initiated this work and the anonymous reviewers for useful comments.

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644052 (HECTOR).
Furthermore, this work has been supported in part by the Austrian Research Promotion Agency (FFG) under grant number 845589, by the Austrian Science Fund (project P26494-N15) and by the French ANR-14-CE28-0015 project.

A Data of Practical Verification/Implementation

Table 2. Evolution of the SEI values for correct key (SEI_c) and the best wrong key ($\max(SEI_w)$) for increasing number of faulty encryptions N. Setup: AES software implementation, single clock glitch (left) and multiple clock glitches (right).

N	SEI_c	$\max(SEI_w)$
4	0.25	1.00
8	0.12	0.43
12	0.13	0.30
16	0.11	0.22
20	0.09	0.16
24	0.09	0.12
28	0.10	0.09
32	0.10	0.09
36	0.09	0.07
40	0.08	0.06
44	0.09	0.05
48	0.08	0.05
52	0.08	0.05
56	0.09	0.04
60	0.09	0.04
64	0.08	0.04
68	0.08	0.03
72	0.08	0.03
76	0.08	0.03
80	0.09	0.03

N	SEI_c	$\max(SEI_w)$
4	0.25	1.00
8	0.18	0.46
12	0.15	0.29
16	0.14	0.18
20	0.14	0.15
24	0.16	0.12
28	0.13	0.09
32	0.13	0.08
36	0.13	0.07
40	0.13	0.06
44	0.13	0.05
48	0.13	0.05
52	0.14	0.04
56	0.13	0.04
60	0.14	0.04
64	0.14	0.03
68	0.14	0.03
72	0.15	0.03
76	0.14	0.03
80	0.14	0.02

Table 3. Evolution of the SEI values for correct key (SEI_c) and the best wrong key ($\max(SEI_w)$) for increasing number of faulty encryptions N. Setup: AES hardware co-processor of a smartcard microcontroller, laser.

N	SEI_c	$\max(SEI_w)$
4	0.62	1.00
8	0.31	0.46
12	0.29	0.29
16	0.22	0.18
20	0.23	0.14
24	0.19	0.11
28	0.17	0.09
32	0.18	0.08
36	0.19	0.07
40	0.17	0.07
44	0.20	0.06
48	0.19	0.05
52	0.16	0.04
56	0.17	0.04
60	0.17	0.03
64	0.17	0.03
68	0.19	0.03
72	0.19	0.03
76	0.21	0.03
80	0.21	0.02

Table 4. Evolution of the SEI values for correct key (SEI_c) and the best wrong key ($\max(\mathrm{SEI}_w)$) for increasing number of faulty encryptions N. Setup: AES co-processor on a general-purpose microcontroller, clock glitch.

N	SEI_c	$\max(\mathrm{SEI}_w)$	N	SEI_c	$\max(\mathrm{SEI}_w)$
32	0.02930	0.08203	1 024	0.00165	0.00164
64	0.01514	0.03369	1 056	0.00162	0.00162
96	0.01020	0.02040	1 088	0.00155	0.00154
128	0.00769	0.01489	1 120	0.00150	0.00151
160	0.00625	0.01125	1 152	0.00147	0.00145
192	0.00521	0.00971	1 184	0.00145	0.00140
224	0.00474	0.00817	1 216	0.00143	0.00136
256	0.00430	0.00693	1 248	0.00138	0.00137
288	0.00398	0.00620	1 280	0.00135	0.00130
320	0.00355	0.00535	1 312	0.00131	0.00128
352	0.00341	0.00492	1 344	0.00131	0.00125
384	0.00304	0.00448	1 376	0.00130	0.00122
416	0.00284	0.00416	1 408	0.00129	0.00120
448	0.00271	0.00388	1 440	0.00127	0.00117
480	0.00266	0.00359	1 472	0.00122	0.00113
512	0.00247	0.00330	1 504	0.00124	0.00111
544	0.00241	0.00315	1 536	0.00125	0.00107
576	0.00240	0.00297	1 568	0.00126	0.00106
608	0.00233	0.00280	1 600	0.00124	0.00104
640	0.00231	0.00264	1 632	0.00123	0.00103
672	0.00229	0.00250	1 664	0.00123	0.00101
704	0.00213	0.00238	1 696	0.00123	0.00100
736	0.00206	0.00227	1 728	0.00123	0.00098
768	0.00195	0.00219	1 760	0.00119	0.00097
800	0.00188	0.00215	1 792	0.00120	0.00095
832	0.00182	0.00202	1 824	0.00117	0.00093
864	0.00180	0.00195	1 856	0.00116	0.00089
896	0.00181	0.00190	1 888	0.00114	0.00087
928	0.00178	0.00180	1 920	0.00113	0.00088
960	0.00171	0.00173	1 952	0.00113	0.00087
992	0.00168	0.00172	1 984	0.00113	0.00084

B AES Co-processor of an 8-bit Microcontroller

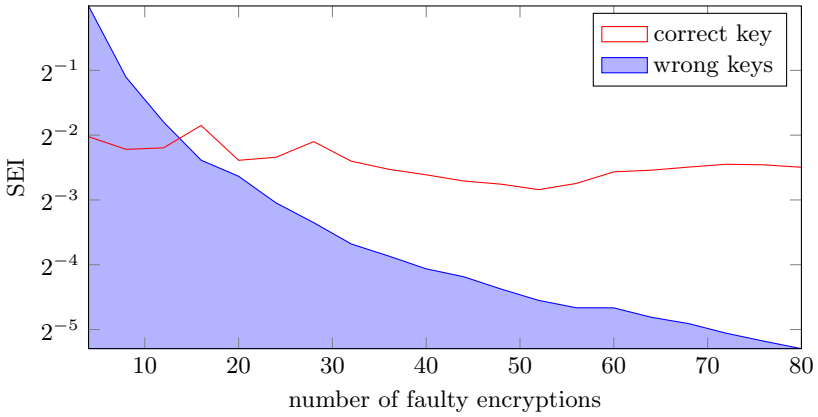


Fig. 12. SEI values for correct key (SEI_c) plotted against best SEI for a wrong key ($\max(SEI_w)$) for increasing number of faulty encryptions. Setup: AES co-processor of an 8-bit microcontroller, single clock glitch.

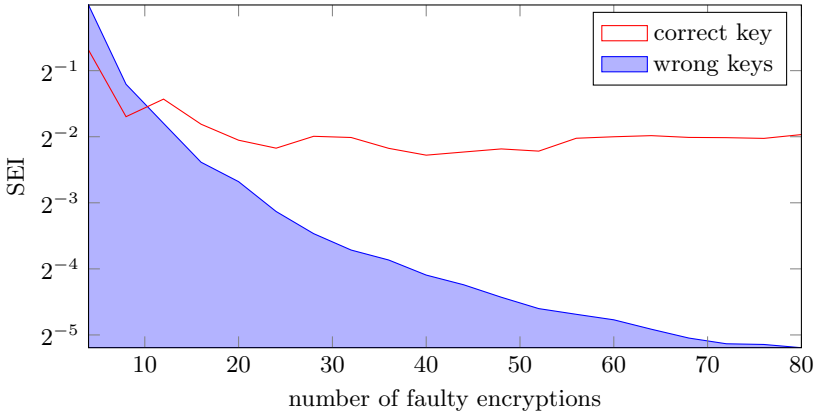


Fig. 13. Evolution of the SEI values with increasing number of faulty encryptions. Setup: AES co-processor of an 8-bit microcontroller, multiple clock glitches.

Table 5. Evolution of the SEI values for correct key (SEI_c) and the best wrong key ($\max(SEI_w)$) for increasing number of faulty encryptions N. Setup: AES co-processor of an 8-bit microcontroller single clock glitch (left) and multiple clock glitches (right).

N	SEI_c	$\max(SEI_w)$
4	0.25	1.00
8	0.21	0.46
12	0.22	0.29
16	0.28	0.19
20	0.19	0.16
24	0.20	0.12
28	0.23	0.10
32	0.19	0.08
36	0.17	0.07
40	0.16	0.06
44	0.15	0.05
48	0.15	0.05
52	0.14	0.04
56	0.15	0.04
60	0.17	0.04
64	0.17	0.04
68	0.18	0.03
72	0.18	0.03
76	0.18	0.03
80	0.18	0.03

N	SEI_c	$\max(SEI_w)$
4	0.62	1.00
8	0.31	0.43
12	0.37	0.29
16	0.29	0.19
20	0.24	0.16
24	0.22	0.11
28	0.25	0.09
32	0.25	0.08
36	0.22	0.07
40	0.21	0.06
44	0.21	0.05
48	0.22	0.05
52	0.22	0.04
56	0.25	0.04
60	0.25	0.04
64	0.25	0.03
68	0.25	0.03
72	0.25	0.03
76	0.25	0.03
80	0.26	0.03

References

1. AVR crypto lib. http://avrcryptolib.das-labor.org, accessed: 2016/01/13
2. Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mendel, F., Mennink, B., Mouha, N., Wang, Q., Yasuda, K.: PRIMATEs. Submission to the CAESAR Competition (Round 2), http://competitions.cr.yp.to/round2/primatesv102.pdf
3. Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: APE: authenticated permutation-based encryption for lightweight cryptography. In: Cid, C., Rechberger, C. (eds.) Fast Software Encryption – FSE 2014. LNCS, vol. 8540, pp. 168–186. Springer (2014)
4. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: AES-COPA. Submission to the CAESAR Competition (Round 2), http://competitions.cr.yp.to/round2/aescopav2.pdf
5. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: Parallelizable and authenticated online ciphers. In: Sako, K., Sarkar, P. (eds.) Advances in Cryptology – ASIACRYPT 2013. LNCS, vol. 8269, pp. 424–443. Springer (2013)
6. Balasch, J., Gierlichs, B., Verbauwhede, I.: An in-depth and black-box characterization of the effects of clock glitches on 8-bit MCUs. In: Fault Diagnosis and Tolerance in Cryptography – FDTC 2011. pp. 105–114. IEEE (2011)
7. Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., Whelan, C.: The sorcerer's apprentice guide to fault attacks. In: Fault Diagnosis and Tolerance in Cryptography – FDTC 2004. pp. 330–342 (2004)
8. Batu, T., Fortnow, L., Rubinfeld, R., Smith, W.D., White, P.: Testing closeness of discrete distributions. J. ACM 60(1), 4 (2013)
9. Bellare, M., Rogaway, P., Wagner, D.: The EAX mode of operation. In: Roy, B.K., Meier, W. (eds.) Fast Software Encryption – FSE 2004. LNCS, vol. 3017, pp. 389–407. Springer (2004)
10. Biehl, I., Meyer, B., Müller, V.: Differential fault attacks on elliptic curve cryptosystems. In: Bellare, M. (ed.) Advances in Cryptology – CRYPTO 2000. LNCS, vol. 1880, pp. 131–146. Springer (2000)
11. Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Kaliski Jr., B.S. (ed.) Advances in Cryptology – CRYPTO '97. LNCS, vol. 1294, pp. 513–525. Springer (1997)
12. Blömer, J., Krummel, V.: Fault based collision attacks on AES. In: Breveglieri, L., Koren, I., Naccache, D., Seifert, J. (eds.) Fault Diagnosis and Tolerance in Cryptography – FDTC 2006. LNCS, vol. 4236, pp. 106–120. Springer (2006)
13. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults (extended abstract). In: Fumy, W. (ed.) Advances in Cryptology – EUROCRYPT '97. LNCS, vol. 1233, pp. 37–51. Springer (1997)
14. Datta, N., Nandi, M.: ELmD. Submission to the CAESAR Competition (Round 2), http://competitions.cr.yp.to/round2/elmdv20.pdf
15. Diffie, W., Hellman, M.E.: Privacy and authentication: An introduction to cryptography. Proceedings of the IEEE 67(3), 397–427 (1979)
16. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon. Submission to the CAESAR Competition (Round 2), http://competitions.cr.yp.to/round2/asconv11.pdf
17. Dworkin, M.: Recommendation for block cipher modes of operation. NIST special publication 800(38A), 1–59 (2001)

18. Fuhr, T., Jaulmes, É., Lomné, V., Thillard, A.: Fault attacks on AES with faulty ciphertexts only. In: Fischer, W., Schmidt, J. (eds.) Fault Diagnosis and Tolerance in Cryptography – FDTC 2013. pp. 108–118. IEEE Computer Society (2013)
19. Goldreich, O., Ron, D.: On testing expansion in bounded-degree graphs. Electronic Colloquium on Computational Complexity (ECCC) 7(20) (2000)
20. Grosso, V., Leurent, G.L., Standaert, F., Varici, K., Journault, A., Durvaux, F., Gaspar, L., Kerckhof, S.: SCREAM. Submission to the CAESAR Competition (Round 2), http://competitions.cr.yp.to/round2/screamv3.pdf
21. Hermelin, M., Cho, J.Y., Nyberg, K.: Multidimensional extension of matsui's algorithm 2. In: Dunkelman, O. (ed.) Fast Software Encryption – FSE 2009. LNCS, vol. 5665, pp. 209–227. Springer (2009)
22. Iwata, T., Minematsu, K., Guo, J., Morioka, S., Kobayashi, E.: CLOC. Submission to the CAESAR Competition (Round 2), http://competitions.cr.yp.to/round2/clocv2.pdf
23. Iwata, T., Minematsu, K., Guo, J., Morioka, S., Kobayashi, E.: SILC. Submission to the CAESAR Competition (Round 2), http://competitions.cr.yp.to/round2/silcv2.pdf
24. Jean, J., Nikolic, I., Peyrin, T.: Deoxys. Submission to the CAESAR Competition (Round 2), http://competitions.cr.yp.to/round2/deoxysv13.pdf
25. Jean, J., Nikolic, I., Peyrin, T.: Joltik. Submission to the CAESAR Competition (Round 2), http://competitions.cr.yp.to/round2/joltikv13.pdf
26. Jean, J., Nikolic, I., Peyrin, T.: KIASU. Submission to the CAESAR Competition (Round 1), http://competitions.cr.yp.to/round1/kiasuv1.pdf
27. Jean, J., Nikolic, I., Peyrin, T.: Tweaks and keys for block ciphers: The TWEAKEY framework. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology – ASIACRYPT 2014. LNCS, vol. 8874, pp. 274–288. Springer (2014)
28. Jutla, C.S.: Encryption modes with almost free message integrity. In: Pfitzmann, B. (ed.) Advances in Cryptology – EUROCRYPT 2001. LNCS, vol. 2045, pp. 529–544. Springer (2001)
29. Kavun, E.B., Lauridsen, M.M., Leander, G., Rechberger, C., Schwabe, P., Yalçin, T.: Prøst. Submission to the CAESAR Competition (Round 1), http://competitions.cr.yp.to/round1/proestv11.pdf
30. Krovetz, T., Rogaway, P.: The software performance of authenticated-encryption modes. In: Joux, A. (ed.) Fast Software Encryption – FSE 2011. LNCS, vol. 6733, pp. 306–327. Springer (2011)
31. Maurine, P.: Techniques for EM fault injection: Equipments and experimental results. In: Bertoni, G., Gierlichs, B. (eds.) Fault Diagnosis and Tolerance in Cryptography – FDTC 2012. pp. 3–4. IEEE Computer Society (2012)
32. McGrew, D.A., Viega, J.: The security and performance of the Galois/Counter Mode (GCM) of operation. In: Canteaut, A., Viswanathan, K. (eds.) Progress in Cryptology – INDOCRYPT 2004. LNCS, vol. 3348, pp. 343–355. Springer (2004)
33. Michael, T., Mukhopadhyay, D., Ali, S.: Differential Fault Analysis of the Advanced Encryption Standard using a single fault. In: Information Security Theory and Practice, pp. 224–233. Springer (2011)
34. Minematsu, K.: AES-OTR. Submission to the CAESAR Competition (Round 2), http://competitions.cr.yp.to/round2/aesotrv2.pdf
35. Minematsu, K.: Parallelizable rate-1 authenticated encryption from pseudorandom functions. In: Nguyen, P.Q., Oswald, E. (eds.) Advances in Cryptology – EUROCRYPT 2014. LNCS, vol. 8441, pp. 275–292. Springer (2014)
36. Paninski, L.: A coincidence-based test for uniformity given very sparsely sampled discrete data. IEEE Transactions on Information Theory 54(10), 4750–4755 (2008)

37. Piret, G., Quisquater, J.: A differential fault attack technique against SPN structures, with application to the AES and KHAZAD. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2003. LNCS, vol. 2779, pp. 77–88. Springer (2003)
38. Rivain, M.: Differential fault analysis on DES middle rounds. In: Clavier, C., Gaj, K. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2009. LNCS, vol. 5747, pp. 457–469. Springer (2009)
39. Rogaway, P.: Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In: Lee, P.J. (ed.) Advances in Cryptology – ASIACRYPT 2004. LNCS, vol. 3329, pp. 16–31. Springer (2004)
40. Rogaway, P., Bellare, M., Black, J.: OCB: A block-cipher mode of operation for efficient authenticated encryption. ACM Trans. Inf. Syst. Secur. 6(3), 365–403 (2003)
41. Rubinfeld, R.: Taming big probability distributions. ACM Crossroads 19(1), 24–28 (2012)
42. Saha, D., Chowdhury, D.R.: Scope: On the side channel vulnerability of releasing unverified plaintexts. In: Dunkelman, O., Keliher, L. (eds.) Selected Areas in Cryptography – SAC 2015. LNCS, Springer (2015), in press
43. Saha, D., Kuila, S., Chowdhury, D.R.: Escape: Diagonal fault analysis of APE. In: Meier, W., Mukhopadhyay, D. (eds.) Progress in Cryptology – INDOCRYPT 2014. LNCS, vol. 8885, pp. 197–216. Springer (2014)
44. Samajder, S., Sarkar, P.: Another look at normal approximations in cryptanalysis. Cryptology ePrint Archive, Report 2015/679 (2015), `http://ia.cr/2015/679`
45. Skorobogatov, S.P., Anderson, R.J.: Optical fault induction attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2002. LNCS, vol. 2523, pp. 2–12. Springer (2002)
46. The CAESAR committee: CAESAR: Competition for authenticated encryption: Security, applicability, and robustness (2014), `http://competitions.cr.yp.to/caesar.html`
47. Wang, L.: Shell. Submission to the CAESAR Competition (Round 2), `http://competitions.cr.yp.to/round2/shellv20.pdf`
48. Whiting, D., Ferguson, N., Housley, R.: Counter with CBC-MAC (CCM). RFC 3610 (2003)
49. Yen, S., Joye, M.: Checking before output may not be enough against fault-based cryptanalysis. IEEE Trans. Computers 49(9), 967–970 (2000), `http://dx.doi.org/10.1109/12.869328`
50. Zussa, L., Dutertre, J.M., Clediere, J., Tria, A.: Power supply glitch induced faults on FPGA: An in-depth analysis of the injection mechanism. In: On-Line Testing Symposium – IOLTS 2013. pp. 110–115. IEEE (2013)